

RALINK TECHNOLOGY, CORP.

RT3X5X/5350 LINUX STATION RELEASE NOTES AND USER'S GUIDE

Copyright © 2011 Ralink Technology, Corp.

All Rights Reserved.

This document is property of Ralink Technology Corporation. Transmittal, receipt, or possession of this document does not express, license, or imply any rights to use, sell, design, or manufacture from this information or the software documented herein. No reproduction, publication, or disclosure of this information, in whole or in part, shall be allowed, unless the prior written consent of Ralink Technology Corporation is obtained.

NOTE: THIS DOCUMENT CONTAINS SENSITIVE INFORMATION AND HAS RESTRICTED DISTRIBUTION.

Proprietary Notice and Liability Disclaimer

The confidential Information, technology or any Intellectual Property embodied therein, including without limitation, specifications, product features, data, source code, object code, computer programs, drawings, schematics, know-how, notes, models, reports, contracts, schedules and samples, constitute the Proprietary Information of Ralink (hereinafter "Proprietary Information")

All the Proprietary Information is provided "AS IS". No Warranty of any kind, whether express or implied, is given hereunder with regards to any Proprietary Information or the use, performance or function thereof. Ralink hereby disclaims any warranties, including but not limited warranties of non-infringement, merchantability, completeness, accuracy, fitness for any particular purpose, functionality and any warranty related to course of performance or dealing of Proprietary Information. In no event shall Ralink be liable for any special, indirect or consequential damages associated with or arising from use of the Proprietary Information in any way, including any loss of use, data or profits.

Ralink retains all right, title or interest in any Proprietary Information or any Intellectual Property embodied therein. The Proprietary Information shall not in whole or in part be reversed, decompiled or disassembled, nor reproduced or sublicensed or disclosed to any third party without Ralink's prior written consent.

Ralink reserves the right, at its own discretion, to update or revise the Proprietary Information from time to time, of which Ralink is not obligated to inform or send notice. Please check back if you have any question. Information or items marked as "not yet supported" shall not be relied on, nor taken as any warranty or permission of use.

Ralink Technology Corporation (Taiwan)

5F, No.5, Tai-Yuen Street,
Chupei City
HsinChu Hsien 302, Taiwan, ROC
Tel +886-3-560-0868
Fax +886-3-560-0818

Sales Taiwan: Sales@ralinktech.com.tw

Technical Support Taiwan: FAE@ralinktech.com.tw

<http://www.ralinktech.com/>

Contents

Contents	3	2.2.12	NetworkType	16	
1.1	RT 305X Version History	7	2.2.13	AuthMode	16
1.1.1	Version 2.6.0.0	7	2.2.14	EncrypType	17
1.1.2	Version 2.5.0.0	7	2.2.15	DefaultKeyID	17
1.1.3	Version 2.4.1.0	7	2.2.16	WEP KeyType	17
1.1.4	Version 2.3.0.0	7	2.2.17	WEP Hex Key	17
1.1.5	Version 2.2.0.0	8	2.2.18	WEP Key String	17
1.1.6	Version 2.0.0.0	8	2.2.19	WPAPSK	18
1.1.7	Version 1.9.0.0	8	2.2.20	WmmCapable	18
1.1.8	Version 1.8.0.0	8	2.2.21	IEEE80211H	18
1.1.9	Version 1.7.0.0	9	2.2.22	PSMode	18
1.1.10	Version 1.6.0.0	9	2.2.23	FastRoaming	18
1.1.11	Version 1.5.0.0	10	2.2.24	RoamThreshold	18
1.1.12	Version 1.4.0.0	10	2.2.25	TGnWifiTest	18
1.1.13	Version 1.3.0.0	10	2.2.26	WirelessEvent	18
1.1.14	Version 1.2.0.0	10	2.2.27	CarrierDetect	19
1.1.15	Version 1.1.0.0	11	2.2.28	HT_RDG	19
1.1.16	Version 1.0.0.0	11	2.2.29	HT_EXTCHA	19
1.2	RT 3352 Version History	11	2.2.30	HT_OpMode	19
1.2.1	Version 2.6.0.0	11	2.2.31	HT_MpduDensity	19
1.2.2	Version 2.5.0.0	11	2.2.32	HT_BW	19
1.3	RT535X Version History	12	2.2.33	HT_AutoBA	19
1.3.1	Version 2.6.0.0	12	2.2.34	HT_AMSDU	19
1.3.2	Version 2.5.0.0	12	2.2.35	HT_BAWinSize	19
2	Configuration	12	2.2.36	HT_GI	20
2.1	Configuration File		2.2.37	HT_MCS	20
RT2860STA.dat	13	2.2.38	HT_MIMOPSEnable	20	
2.2	Configuration file use	14	2.2.39	HT_MIMOPSMODE	20
2.2.1	CountryRegion	14	2.2.40	HT_DisallowTKIP	20
2.2.2	CountryRegionForABand	15	2.2.41	HT_RxStream	20
2.2.3	SSID	15	2.2.42	HT_TxStream	20
2.2.4	WirelessMode	15	2.2.43	HT_LinkAdapt	21
2.2.5	Channel	16	2.2.44	HT_HTC	21
2.2.6	BGProtection	16	2.2.45	HT_DisableReordering	21
2.2.7	TxPreamble	16	2.2.46	BeaconLostTime	21
2.2.8	RTSThreshold	16	2.2.47	AutoRoaming	21
2.2.9	FragThreshold	16	2.2.48	MacAddress	21
2.2.10	TxBurst	16	2.2.49	TDLSCapable	22
2.2.11	PktAggregate	16	2.2.50	AutoConnect	22
			2.2.51	HT_40MHZ_INTOLERANT	22
			2.2.52	AntGain	22
			2.2.53	BandedgeDelta	22
			2.2.54	HwAntDiv=Value (RT5350 only)	22
			2.2.55	P2P_GOIntent	23

2.2.56	P2P_DevName	23	3.1.48	BeaconLostTime=value ..	31
2.2.57	P2P_ListChannel	23	3.1.49	AutoRoaming=value	31
2.2.58	P2P_OpChannel	23	3.1.50	SiteSurvey=value	32
3	Wireless Tools	23	3.1.51	TdlsCapable=value	32
3.1	lwpri ra0 set use	23	3.1.52	TdlsSetup=value	32
3.1.1	DriverVersion	23	3.1.53	AutoReconnect=value	32
3.1.2	CountryRegion	23	3.1.54	AdhocN=value	32
3.1.3	CountryRegionABand	24	3.1.55	AntGain	32
3.1.4	SSID	24	3.1.56	HwAntDiv=Value (RT5350 only)	33
3.1.5	WirelessMode	25	3.2	lwpri ra0 show use	33
3.1.6	TxBurst:	25	3.2.1	connStatus	33
3.1.7	PktAggregate:	25	3.2.2	driverVer	33
3.1.8	TxPreamble:	25	3.2.3	bainfo	33
3.1.9	TxPower:	25	3.2.4	rxbulk	33
3.1.10	Channel	26	3.2.5	txbulk	33
3.1.11	BGProtection:	26	3.2.6	AutoReconnect	33
3.1.12	RTSThreshold:	26	3.2.7	WPAPSK	33
3.1.13	FragThreshold:	26	3.2.8	PMK	33
3.1.14	NetworkType:	26	3.3	lwpri ra0 use	35
3.1.15	AuthMode:	26	3.3.1	radio_off	35
3.1.16	EncrypType:	26	3.3.2	radio_on	35
3.1.17	DefaultKeyID:	27	3.4	lwpri Examples	36
3.1.18	Key1	27	3.4.1	Infrastructure	36
3.1.19	Key2	27	3.4.2	Ad-Hoc	37
3.1.20	Key3	27	3.4.3	Get site survey	37
3.1.21	Key4	27	3.4.4	Get Statistics	37
3.1.22	WPAPSK	27	3.4.5	ANY SSID	37
3.1.23	WmmCapable	27	3.5	iwlist	37
3.1.24	IEEE80211H	28	3.6	iwconfig	37
3.1.25	PSMode	28	4	WPS – Wi-Fi Protected Setup	39
3.1.26	ResetCounter	28	4.1	lwpri use	39
3.1.27	Debug	28	4.1.1	wsc_conf_mode	39
3.1.28	CarrierDetect	28	4.1.2	wsc_mode	39
3.1.29	HtRdg	28	4.1.3	wsc_pin	40
3.1.30	HtExtcha	29	4.1.4	wsc_ssid	40
3.1.31	HtOpMode	29	4.1.5	wsc_bssid	40
3.1.32	HtMpduDensity	29	4.1.6	wsc_start	40
3.1.33	HtBw	29	4.1.7	wsc_stop	40
3.1.34	HtAutoBa	29	4.1.8	wsc_gen_pincode	40
3.1.35	HtAmsdu	29	4.1.9	wsc_cred_count	40
3.1.36	HtBaWinSize	29	4.1.10	wsc_cred_ssid	40
3.1.37	HtGi	30	4.1.11	wsc_cred_auth	41
3.1.38	HtMcs	30	4.1.12	wsc_cred_encr	41
3.1.39	HtProtect	30	4.1.13	wsc_cred_keyidx	41
3.1.40	HtMimoPs	30			
3.1.41	FixedTxMode	30			
3.1.42	LongRetry	30			
3.1.43	ShortRetry	31			
3.1.44	HtTxStream=value	31			
3.1.45	HtRxStream=value	31			
3.1.46	HtDisallowTKIP=value	31			
3.1.47	HtBaDecline	31			

4.1.14	wsc_cred_key	41	6.1.1	ATE	58
4.1.15	wsc_cred_mac	42	6.1.2	ATEDA	58
4.1.16	wsc_conn_by_idx	42	6.1.3	ATESA	58
4.1.17	wsc_auto_conn	42	6.1.4	ATEBSSID	58
4.1.18	wsc_ap_band	42	6.1.5	ATECHANNEL	58
4.1.19	Wsc4digitPinCode	42	6.1.6	ATETXPOW0	59
4.1.20	WscV2Support (WPS2.0)	43	6.1.7	ATETXPOW1	59
4.2	WPS STA as an Enrollee or Registrar	43	6.1.8	ATETXFREQOFFSET	59
4.2.1	Enrollee Mode	43	6.1.9	ATETXLEN	59
4.2.2	Registrar Mode	44	6.1.10	ATETXCNT	59
4.3	WPS IOCTL use	45	6.1.11	ATETXMODE (Refer to TxMode)	59
4.3.1	iwpriv commands without argument	45	6.1.12	ATETXBW (Refer to TxMode)	60
4.3.2	iwpriv commands with one INT argument	45	6.1.13	ATETXGI (Refer to TxMode)	60
4.3.3	iwpriv commands with string argument	45	6.1.14	ATETXMCS (Refer to TxMode)	60
4.4	WPS IOCTL Sample Program ...	47	6.1.15	ATETXANT	60
5	Wifi direct - P2P command	49	6.1.16	ATERXANT	60
5.1	Iwpriv use	49	6.1.17	ATERXFER	60
5.1.1	P2pOpMode	49	6.1.18	ATESHOW	61
5.1.2	p2pLisCh	49	6.1.19	ATEHELP	61
5.1.3	p2pOpCh	49	6.1.20	ResetCounter	61
5.1.4	p2pGoInt	50	6.1.21	ATERRF	61
5.1.5	p2pDevName	50	6.1.22	ATELDE2P	61
5.1.6	p2pWscMode	50	6.1.23	ATETSSICBAEX (For RT5350 only)	61
5.1.7	p2pWscConf	50	6.1.24	ATEIPG	61
5.1.8	p2pScan	50	6.1.25	ATEPAYLOAD	62
5.1.9	p2pInv	51	6.1.26	ATETSSICBA	62
5.1.10	p2pDevDisc	51	6.2	Tx Mode, MCS, BW and GI Selection Table	62
5.1.11	p2pLink	51	6.2.1	MODE = 0, Legacy CCK ...	62
5.1.12	p2pCfg	51	6.2.2	MODE = 1, Legacy OFDM62	
5.1.13	p2pTab	51	6.2.3	MODE = 2, HT Mixed Mode	62
5.1.14	p2pProv	51	6.2.4	MODE = 3, HT Greenfield	62
5.1.15	p2pStat	51	6.3	Examples	63
5.1.16	p2pReset	52	6.3.1	Check EVM & Power	63
5.1.17	p2pPerTab	52	6.3.2	Check Carrier	64
5.1.18	p2pDefConfMthd	52	6.3.3	Check specturm mask ...	64
5.1.19	p2pLinkDown	52	6.3.4	Frequency offset tuning ..	64
5.1.20	p2pSigmaEnable	52	6.3.5	Rx	64
5.1.21	Other P2P command	52	6.3.6	Show all ate parameters ..	65
5.1.22	P2P example:	53	6.3.7	Online help	65
6	ATE Test Command Format	57	6.3.8	Display Rx Packet Count and RSSI	65
6.1	iwpriv ra0 set [parameters]=[val]	58	6.3.9	Internal ALC calibration (For RT5350 only)	66

6.3.10	Internal ALC function testing in ATE mode (For RT5350 only)	66	7.2.1	Set Data, Parameters is Same as iwpriv	78
6.4	iwpriv ra0 bbp [parameters]=[Value]	67	7.2.2	Get Data, Parameters is Same as iwpriv	79
6.4.1	BBPID	67	7.2.3	Set Raw Data with Flags	80
6.4.2	BBPID=Value	67	7.2.4	Get Raw Data with Flags	88
6.5	iwpriv ra0 mac [parameters]=[val]	67	7.2.5	Set Raw Data with Flags	97
6.5.1	MAC_OFFSET	67	8	IOCTL instructions	99
6.5.2	MAC_OFFSET=Value	67	8.1	Get Data	99
6.6	iwpriv ra0 e2p [parameters]=[val]	67	8.1.1	GET station connection status:	99
6.6.1	E2P_ADDR	68	8.1.2	GET station statistics information:	99
6.6.2	E2P_ADDR=Value	68	8.1.3	GET AP list table:	99
6.7	Example	68	8.1.4	GET scan table:	99
6.7.1	Hardware access	68	8.1.5	GET station's MAC:	99
6.7.2	Statistic counter operation	68	8.1.6	GET station connection status:	100
6.7.3	Suggestion:	68	8.1.7	GET AP's BSSID	100
6.8	ated	70	8.1.8	GET SSID	100
6.8.1	Introduction	70	8.1.9	GET station's last TX related information:	100
6.8.2	Environment setup	70	8.1.10	GET station's last RX related information:	100
6.8.3	How to use ated for ATE purpose	70	8.1.11	GET station's wireless mode:	100
7	IOCTL	71	8.1.12	GET Bss type:	101
7.1	Parameters for iwconfig	71	8.1.13	GET Authentication Mode:	101
7.2	Parameters for iwpriv	78	8.1.14	GET Encryption Type:	102
			8.1.15	GET RSSI 0 (unit: db)	102
			8.1.16	GET RSSI 1 (unit: db)	102
			8.1.17	GET RSSI 2 (unit: db)	102
			8.1.18	GET Driver wireless extension version	102
			8.2	How to display rate, BW:	104

1.1 RT 305X Version History

1.1.1 Version 2.6.0.0

1. Support WPS 2.0 STA.
2. Support P2P on STA.
3. Fix issue for dma_addr_t problem.
4. Pass Carrier Detection on internal pretest.
5. Prevent MiniportMMRequest into infinite loop.
6. Update ATE code.
7. Fix issue: STA didn't use legacy rate when STA is Ad-hoc creator and in OPEN-WEP or WPAPSK-TKIP.
8. Fix issue: Beacon lost when STA tries to do WPS with WPS AP or STA connects to hidden AP.
9. Fix issue: Shorten WPS procedure.
10. Fix issue: STA supports the WPA authentication which group cipher is WEP.

1.1.2 Version 2.5.0.0

11. Add mac80211 iw utility other commands support.
12. WMM ACM: see history of acm_comm.c.
13. Fix issue: interrupt() issue in fedora core 7.
14. Fix issue in 64bit CPU:
15. CFG80211: Patch for Linux 2.6.32.
16. Add an ATE command for AUTOALC.
17. Add an ATE command for interpacket GAP.
18. Add an ATE command for ATE payload pattern.
19. Fix issue: UAPSD SP can not be closed correctly for mix-mode power save.
20. Fix issue: The extended rate of probe-req is disappeared in AP-Client mode.
21. Fix issue: kernel panic when forward VLAN bc/mc packet to wireless LAN.
22. Fix issue: get wrong VLAN priority from VLAN tag in RTMPCheckEtherType().
23. Fix issue: no 11b basic rate in beacon when change WirelessMode to 2.4G, then 5G, then 2.4G
24. Fix issue: no 11b-only mode can be set in AP mode.
25. MBSS: add function to set different phy mode for different BSS.
26. QLOAD: Fix issue for big endian or value of QLOAD element in beacon will be wrong.
27. TX Block: Fix no packet can be sent when TX Fail count > a threshold for non-WDS interface.
28. Fix issue: WPS process failed with WPS Client that sends dis-assoc close to WSC_DONE.

1.1.3 Version 2.4.1.0

1. Fix issue: Radio On/Off can not work.

1.1.4 Version 2.3.0.0

1. WMM ACM AP/STA support. (Pass WiFi Test Plan v0.30)
2. Add HAS_STATS_COUNT compile option.
3. Provide some 802.1x parameter support (Quiet-period, idle-timeout, NAD-ID).
4. Fix issue: The Tx data rate can't be fixed in B/G PHY mode.
5. QLOAD: Add status information display function. "qload show" & QloadClr.
6. QLOAD: Add channel busy alarm and a command thread in PCI/USB driver.
7. It supports the MAC assignment from configuration file(.dat).
8. Fix issue: No packet is delivered from AP when PsMode from PS to ACTIVE.
9. Fast Power Save: Use Null frame with PM=0 to get buffered packet, not PS-Poll.
10. miniupnpd supported.

11. AP-Client and WDS support fixed rate.
12. Support GreenAP.
13. Support Single SKU.
14. Fix issue: RT3062/RT3562 can't enable/disable Radio.
15. Discard IgmpSn enable setting per BSSID and change it to as global setting.

1.1.5 Version 2.2.0.0

1. New generation schema for multiple OS porting
2. New chip support for RT3572
3. New chip support for RT3062/RT3562.
4. Restrict the encryption type in HT mode..
5. Support 802.1x reauthentication mechanism.
6. Limit the STA connection count per BSS.
7. Some variables support MBSS setting.
8. Support WDS entry life check function
9. Support Dot11K RRM for all testing cases of voice-enterprise testing event.

1.1.6 Version 2.0.0.0

1. Added Global Country Domain supported.
2. Fix bug: suspend/resume error when ra0 down, rax up
3. Add new UAPSD SP counting mechanism.
4. Fix bug: Ikanos WDS, AP Client, Mess interface get problem.
5. Add new WSC hardware push button function for PCI & USB.
6. Added a function allow user to sepcific Tx rate for Mcast packets.
7. Migrate Mesh supporting to Draft-2.0.
8. Support WAPI functionality
9. Modify the priority of BAR transmission to solve the connection issue with Intel 4965 11n STA.

1.1.7 Version 1.9.0.0

1. Replace iwpriv cmd "AccessControlList" by "ACLAddEntry" and "ACLClearAll"
2. Fixed the wrong usage of AtoH ().
3. Support new Windows ATE GUI.
4. Add a command "iwpriv ra0 set ATERE2P=1" to display all EEPROM content.
5. Correct the limitation of the length of fragment
6. Fix bug: Fail to transmit packets through AMPDU way except the case that AP to STA.
7. Wrong Hareware packet length calculation of Mesh packet if it has been fragmented.
8. Support SIGMA 8622/8624 platform.
9. Add WPS PBC Session Overlap Detecting.
10. Add WPS 4-PinCode Support.
11. Fixed WPS enable PSP can not associate AP when AP security set to WPA-PSK.
12. If 11n station operated in power save mode, the AP should transmit none AMPDU nor AMSDU to the station for the Ps-Poll.
13. 20/40 overlapping BSS scan mechanism and bandwidth adjustment.
14. Support 802.11n draft 4.0

1.1.8 Version 1.8.0.0

1. Show Tx/Rx statistics per MBSS.
2. 802.1x supports failover mechanism.
3. Add watchdog to prevent MAC/BBP into the deadlock condition.
4. Support pure 11n with 5G band.
5. Update Timer Functions

6. In multiple cards application, the interface name is changed to raxx_k, where xx means card ID (0 ~ 31) and k means the BSS number (0 ~ 7)
7. Support individual MCS per BSS.
8. Add IKANOS Vx160 and Vx180.
9. Add station keep alive detection function in AP mode.
10. The SIFS of CCK is changed to 16 micro seconds to fix the connection problem with INTEL 2200bg cards.
11. QBSS Load Element is added to provide channel utilization information to all STAs.
12. Fix bug : After AP re-key, the ping connection from client to AP would be time-out within several seconds.
13. Support Mesh function.
14. Support SNMP function.
15. Big-endian ATE supported.

1.1.9 Version 1.7.0.0

1. Support IDS notification mechanism.
2. Change IRQ LOCK to SEM LOCK.
3. Fix bug : When QoS(non-BE) and fragment packets are received, AP would calculate wrong MIC in TKIP mode.
4. Support Non-GPL MD5.
5. Update Group rekey mechanism.
6. Fix BA time-out issue for Intel wireless card 4965AGN with version 11.5.0.32.
7. Add command "iwpriv ra0 set ATELDE2P=1" to overwrite all EEPROM contents from "/etc/Wireless/RT2860(/70)AP(/STA)/e2p.bin".
8. Fix RTS threshold issue in 5G-band.
9. Add DLS Function.
10. IPV6 MLDv2 support.
11. Fix VLAN ID >= 256 can not be used.
12. Added PCIE MSI supporting for RT2890.
13. Added new channel list builder that create channel list according to country-code and channel Geography (in/out door).

1.1.10 Version 1.6.0.0

1. Fix bug: Before AP shutdown, AP doesn't notify those associated STA through dis-association.
2. Fix bug: The Atheros wireless STA card built in MacBook can't work normally when HT mode and the encryption is WEP or TKIP.
3. The support region in A band synchronizes with EEPROM.
4. It supports to initialize current wireless MAC address from E2PROM or module parameter.
5. Support maximum 8 MBSS and each beacon maximum length is 512 bytes.
6. Support 5-GHz band ATE.
7. Send DisAssoc frame to timeout STA.
8. Workaround for Atheros STA on AES mode.
9. Tx RTS/CTS when AP setup BA.
10. Driver sends IAPP L2 frame instead of Daemon.
11. Correct some timeout values of WPS.
12. Fix bug: The 802.1x daemon (rt2860apd) has some problem for parsing multiple parameters in MBSS.
13. Fix bug: The AP site survey signal isn't correct.
14. Provide some 11n statistics variables.
15. Fix bug: RT2561 module can not be removed after RT2860 module is inserted.
16. Added DEO (100 ~ 140) channel list for Ganmany.
17. Support wds phy mode and security setting for each wds link.
18. Fix bug: The Atheros legacy USB STA card can't connect to our AP in WPA-TKIP.
19. Modify rate adaptation for fast ramp-up tuning.

20. Fix WPS IOT issue with Atheros external registrar. Need Sync the user space daemon "wscd" to version 0.1.0
21. Fix bug: The BlackBerry/HTC can't connect to our AP.
22. Correct the default values of those WMM EDCA parameters.

1.1.11 Version 1.5.0.0

1. Added McastPhyMode and McastMcs iwpriv commands let user to specify the rate for Multicast packets transmission.
2. Added two configurations of McastPhyMode and McastMcs.
3. Re-organize the WPA state machine in order to the consistency between AP and AP-Client.
4. Added DFS support.
5. Added Carrier-Sense support.
6. Fixed a bug about dissection issue about ';' in profile.
7. Fixed CountryRegion and channel map, when profile's channel have not on channel list.
8. Fixed 802.1x Authentication problem with 1x-WEP/WPA(2)-Enterprise when WPS is enabled.
9. Fix bug: Once the radio is off and on, the beacon frames disappear.

1.1.12 Version 1.4.0.0

1. Wireless IGMP snooping support for multimedia streaming.
2. Access control list support
3. Re-organize the Rx data path.
4. AP client WPS support.
5. Fix the Auto-selecting channel issue.
6. Add CountryString ioctl command function.
7. Bug fix for Atheros WPS STA can not config WPS AP when Atheros JumpStart STA is external Registrar.
8. Merge for WCN test modify to WPS functions.
9. Patch for 11n requirement, if HT mode is set and BW is 40MHz in A-band, the supported Channel number must be the multiple of 2.
10. Fix bug: If STA card operated with zero-config, the group rekey negotiation of WPA2(PSK)-AES always fails.
11. Add Tx & Rx Stream functionality.
12. Support QA user interface for ATE function.

1.1.13 Version 1.3.0.0

1. Add vlan tag support for each BSS.
2. Add support for 32bit/64bit Linux.
3. Merge in plugfest code.
4. Support Ap-Client function.
5. Add new parameter "bWiFiTest" for WPA & WMM WiFi-Test.
6. Add the setting of Japan filter coefficients for ATE.
7. Fix bug for channel have not update when auto channel select was true.
8. Add protect for RTMP_IRQ_LOCK to avoid in spin_lock_irqsave call spin_lock_bh cause kernel warning messages.

1.1.14 Version 1.2.0.0

1. Fix bug for counterMeasures in WiFi test.
2. Write TXWI in ATE's way and disable any protection mechanism when ATE is running.
3. Disable ATE RSSI statistics when ATE is not running.
4. Select DAC according to HT or Legacy mode.
5. Support WPA2 Pre-authentication.
6. Fix WDS panic bug.
7. Shift skb control block used by driver to offset 10 to avoid dirty cb[] from protocol stack.

8. Fix issue - 802.1X daemon may cause throughput reduction.
9. Support Wireless event log mechanism.
10. Add a 200ms-timer to enqueue EAPoL-Start for WPAPSK, not RTMPusecDelay.
11. Auto-selecting channel check.

1.1.15 Version 1.1.0.0

1. Add fast rate switch.
2. Modify fast rate switch timer form periodic to trigger by condition.
3. Fix UAPSD bugs for null frame was drop.
4. Fix management queue pass qos null frame ...
5. Code freeze for Wifi.
6. Merge code from Plugfest #6.
7. Add "iwpriv ra0 show driverinfo" to show the driver version.

1.1.16 Version 1.0.0.0:

1. Interface support and bugs fix for WMM (Under testing).
2. DFS support.
3. Support WPA over WDS.
4. Bug fix for two WPAPSK-STAs causes the AP to crash.
5. Bug fix for BG-STAs will link up with B-only-AP.
6. Fix compatibility issue in 802.11d.

1.2 RT 3352 Version History

1.2.1 Version 2.6.0.0

1. Support WPS 2.0.
2. Update TSSI code.
3. Support Linux Kernel 2.6.36
4. Update ATE code.
5. Fix does not recognize implicit block ACK buffer size.
6. Fixed Ad-hoc issue for OPEN-WEP or WPAPSK-TKIP.
7. Fixed WPS issue for beacon lost.
8. Fixed link status while using WAPI.
9. Fix a bug for BA action frames when the station enters power-save mode.
10. STA WPS IE is invalid in unicast probe req.
11. STA supports the WPA authentication which group cipher is WEP.
12. Fixed WPS with Ralink WpaSupplicant
13. Add new mechanism to update STA scan table.
14. HT IE wasn't included in broadcast probe request.
15. Add RSSI information with Ralink WpaSupplicant while doing scan.
16. Support P2P on STA

1.2.2 Version 2.5.0.0

1. Fix issue: The start address of HeaderBuf must be aligned by 4 when
2. VENDOR_FEATURE1_SUPPORT is enabled.
3. WMM ACM: see history of acm_comm.c.
4. Add an ATE command for AUTOALC.
5. Add an ATE command for interpacket GAP.
6. Add an ATE command for ATE payload pattern.
7. Fix issue: UAPSD SP can not be closed correctly for mix-mode power save. UAPSD + legacy PS.
8. [Bug fixed] The extended rate of probe-req is disappeared in AP-Client mode

9. Fix issue: kernel panic when we forward VLAN bc/mc packet to wireless LAN.
10. Fix issue: some reports from Prevent software analysis tool.
11. Fix issue: get wrong VLAN priority from VLAN tag in RTMPCheckEtherType().
12. Fix issue: no 11b basic rate in beacon when we change WirelessMode to 2.4G, then 5G, then 2.4G
13. Fix issue: no 11b-only mode can be set in AP mode.
14. MBSS: add function to set different phy mode for different BSS.
15. Phy Mode: add check if the chip supports 5G band when WirelessMode is 5G band for command WirelessMode and MBSSWirelessMode.
16. QLOAD: Fix issue for big endian. Or value of QLOAD element in beacon will be wrong.
17. Station: Add new rate switch algorithm (AGS) for 1*1, 2*2, 3*3.
18. AP: Fix power save problem when station is in power-save mode and send (re)associate frame again we will think the station is still in PS mode, but the station is in ACTIVE mode.
19. TX Block: Fix no packet can be sent when TX Fail count > a threshold for non-WDS interface in RTMPDeQueuePacket().
20. Fix WPS issue: WPS process failed with some WPS Client that sends dis-assoc close to WSC_DONE.

1.3 RT535X Version History

1.3.1 Version 2.6.0.0

1. Support WPS 2.0.
2. Update TSSI code.
3. Support Linux Kernel 2.6.36
4. Update ATE code.
5. Fix does not recognize implicit block ACK buffer size.
6. Fixed Ad-hoc issue for OPEN-WEP or WPAPSK-TKIP.
7. Fixed WPS issue for beacon lost.
8. Fixed link status while using WAPI.
9. Fix a bug for BA action frames when the station enters power-save mode.
10. STA WPS IE is invalid in unicast probe req.
11. STA supports the WPA authentication which group cipher is WEP.
12. Fixed WPS with Ralink WpaSupplicant
13. Add new mechanism to update STA scan table.
14. HT IE wasn't included in broadcast probe request.
15. Add RSSI information with Ralink WpaSupplicant while doing scan.
16. Support P2P on STA

1.3.2 Version 2.5.0.0

1. RT5350 init version.
2. AP/STA pass TGN and WPS QA pre test.
3. Feature Support: 16 Multiple SSID support.
4. New multiple BSSID (use MAC address Byte0 to distinguish different BSSID).
5. Hardware WAPI support.
6. Hardware Antenna Diversity support.
7. Internal TSSI support.
8. Software to config BBP diversity directly and skip the calibration in EEPROM.

2 CONFIGURATION

STA driver can be configured via following interfaces, i.e.

1. configuration file
2. "iwconfig" command

3. "iwpriv" command

Note:

- 1) modify configuration file "RT2860STA.dat" in /etc/Wireless/RT2860STA/RT2860STA.dat.
- 2) iwconfig/iwpriv comes with kernel.
- 3) iwpriv use, please refer to below sections for details.

2.1 Configuration File RT2860STA.dat

```
# Copy this file to /etc/Wireless/RT2860STA/RT2860STA.dat
# This file will be read on loading driver module.
#
# Use "vi RT2860STA.dat" to modify settings according to your need.
#
#
#The word of "Default" must not be removed
Default
CountryRegion=5
CountryRegionABand=7
CountryCode=
ChannelGeography=1
SSID=11n-AP
NetworkType=Infra
WirelessMode=5
Channel=0
BeaconPeriod=100
TxPower=100
BGProtection=0
TxPreamble=0
RTSThreshold=2347
FragThreshold=2346
TxBurst=1
PktAggregate=0
WmmCapable=1
AckPolicy=0;0;0;0
AuthMode=OPEN
EncryptType=NONE
WPAPSK=
DefaultKeyID=1
Key1Type=0
Key1Str=
Key2Type=0
Key2Str=
Key3Type=0
Key3Str=
Key4Type=0
Key4Str=
PSMode=CAM
AutoRoaming=0
RoamThreshold=70
APSDCapable=0
APSDAC=0;0;0;0
HT_RDG=1
```

HT_EXTCHA=0
 HT_OpMode=1
 HT_MpduDensity=4
 HT_BW=1
 HT_BADecline=0
 HT_AutoBA=1
 HT_BADecline=0
 HT_AMSDU=0
 HT_BAWinSize=64
 HT_GI=1
 HT_MCS=33
 HT_MIMOPSMODE=3
 HT_DisallowTKIP=1
 IEEE80211H=0
 TGnWifiTest=0
 WirelessEvent=0
 CarrierDetect=0
 AntDiversity=0
 BeaconLostTime=4
 FtSupport=1

NOTE:

WMM parameters
 WmmCapable
 AckPolicy1~4

Set it as 1 to turn on WMM Qos support
 Ack policy which support normal Ack or no Ack
 (AC_BK, AC_BE, AC_VI, AC_VO)

All WMM parameters do not support iwpriv command but 'WmmCapable', please store all parameter to RT2870STA.dat, and restart driver.

2.2 Configuration file use

Syntax is 'Param'='Value' and describes below.

SectionNumber	Param Value
	...
	...
	...

2.2.1 CountryRegion

value

Region	Channels
0	1-11
1	1-13
2	10-11
3	10-13
4	14
5	1-14

6	3-9
7	5-13
31	1-14
32	1-11 active scan, 12 and 13 passive scan
33	1-14 all active scan, 14 b mode only

2.2.2 CountryRegionForABand

value

Region	Channels
0	36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, 161, 165
1	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
2	36, 40, 44, 48, 52, 56, 60, 64
3	52, 56, 60, 64, 149, 153, 157, 161
4	149, 153, 157, 161, 165
5	149, 153, 157, 161
6	36, 40, 44, 48
7	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 165
8	52, 56, 60, 64
9	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 132, 136, 140, 149, 153, 157, 161, 165
10	36, 40, 44, 48, 149, 153, 157, 161, 165
11	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 149, 153, 157, 161

2.2.3 SSID

value

0~z, 1~32 ascii characters.

2.2.4 WirelessMode

value

- 0: legacy 11b/g mixed
- 1: legacy 11B only
- 2: legacy 11A only
- 3: legacy 11a/b/g mixed
- 4: legacy 11G only
- 5: 11ABGN mixed
- 6: 11N only
- 7: 11GN mixed
- 8: 11AN mixed
- 9: 11BGN mixed
- 10: 11AGN mixed
- 11: 11N only in 5G band only

2.2.5 Channel

value

depends on CountryRegion or CountryRegionForABand

2.2.6 BGProtection

value

0: Auto
1: Always on
2: Always off

2.2.7 TxPreamble

value

0:Preamble Long
1:Preamble Short
2:Auto

2.2.8 RTSThreshold

value

1~2347

2.2.9 FragThreshold

value

256~2346

2.2.10 TxBurst

value

0: Disable
1: Enable

2.2.11 PktAggregate

value

0: Disable
1: Enable

2.2.12 NetworkType

value

Infra: infrastructure mode
Adhoc: adhoc mode

2.2.13 AuthMode

value

OPEN For open system

SHARED For shared key system
 WPAUTO Auto switch between OPEN and SHARED
 WPAPSK For WPA pre-shared key (Infra)
 WPA2PSK For WPA2 pre-shared key (Infra)
 WPANONE For WPA pre-shared key (Adhoc)
 WPA
 WPA2

2.2.14 EncryptType

value
 NONE For AuthMode=OPEN
 WEP For AuthMode=OPEN or AuthMode=SHARED
 TKIP For AuthMode=WPAPSK or WPA2PSK
 AES For AuthMode=WPAPSK or WPA2PSK

2.2.15 DefaultKeyID

value
 1~4

2.2.16 WEP KeyType

Key1Type=value
 Key2Type=value
 Key3Type=value
 Key4Type=value
 value
 0 hexadecimal type
 1 assic type
 (use: reading profile only)

2.2.17 WEP Hex Key

Key1=value
 Key2=value
 Key3=value
 Key4=value
 value
 10 or 26 hexadecimal characters eg: 012345678
 5 or 13 ascii characters eg: passwd
 (use: "iwpriv" only)

2.2.18 WEP Key String

Key1Str=value
 Key2Str=value
 Key3Str=value
 Key4Str=value
 value
 10 or 26 characters (key type=0)
 5 or 13 characters (key type=1)

(use: reading profile only)

2.2.19 WPAPSK

value
8~63 ASCII or
64 HEX characters

2.2.20 WmmCapable

value
0: Disable WMM
1: Enable WMM

2.2.21 IEEE80211H

Enabel IEEE802.11h support
Value:
0:Disable
1:Enable

2.2.22 PSMode

value	CAM	Constantly Awake Mode
	Max_PSP	Max Power Saving
	Fast_PSP	Fast Power Saving
	Legacy_PSP	Legacy Power Saving

2.2.23 FastRoaming

value
0: Disabled
1: Enabled

2.2.24 RoamThreshold

value
0 ~ 255

2.2.25 TGnWifiTest

value
0: Disabled
1: Enabled

2.2.26 WirelessEvent

value
0: Disabled
1: Enabled (send custom wireless event)

2.2.27 CarrierDetect

value

0: Disabled
1: Enabled

2.2.28 HT_RDG

value

0: Disabled
1: Enabled

2.2.29 HT_EXTCHA

value

0: Below
1: Above

2.2.30 HT_OpMode

value

0: HT mixed format
1: HT greenfield format

(Note) If you want to do TGN WIFI green field item, please set HT_OpMode=1

2.2.31 HT_MpduDensity

value

0 ~ 7

2.2.32 HT_BW

value

0: 20MHz
1: 40MHz

2.2.33 HT_AutoBA

value

0: Disabled
1: Enabled

2.2.34 HT_AMSDU

value

0: Disabled
1: Enabled

2.2.35 HT_BAWinSize

value

1 ~ 64

2.2.36 HT_GI

value

- 0: long GI
- 1: short GI

2.2.37 HT_MCS

value

- 0 ~ 15
- 33: auto

2.2.38 HT_MIMOPSEnable

Enable/Disable the 802.11n SM power save function.

Value:

- 0:Disable
- 1:Enable (Default)

2.2.39 HT_MIMOPSMode

value

- 0: Static SM Power Save Mode
- 2: Reserved
- 1: Dynamic SM Power Save Mode
- 3: SM enabled
(not fully support yet)

2.2.40 HT_DisallowTKIP

Enable/Disable N rate with 11N ap when cipher is WEP or TKIP.

Value:

- 0: FALSE
- 1: TRUE

Default setting is disable.

2.2.41 HT_RxStream

Set the number of spatial streams for reception

Value:

- 1: 1 Rx stream
- 2: 2 Rx stream

2.2.42 HT_TxStream

Set the number of spatial streams for transimtion

Value:

1: 1 Tx stream

2: 2 Tx stream

2.2.43 HT_LinkAdapt

Enable/Disable HT Link Adaptation Control

Value:

0:Disable (Default)

1:Enable

2.2.44 HT_HTC

Enable/disable HTC field of data frames send with 802.11n data rates

Value:

0:Disable (Default)

1:Enable

2.2.45 HT_DisableReordering

Disable AMPDU re-ordering handling mechanism

Value:

0:Disable (Default)

1:Enable

2.2.46 BeaconLostTime

Change Beacon Lost Time

Value:

1 ~ 60 seconds

Default value is 4 seconds

2.2.47 AutoRoaming

Enable/disable auto roaming mechanism

Value:

0: disable (Default)

1: enable

2.2.48 MacAddress

MacAddress=value

Value: XX:XX:XX:XX:XX:XX

2.2.49 TDLSCapable

Enable/disable TDLSCapable function

Value:

- 0: disable
- 1: enable

2.2.50 AutoConnect

Enable/Disable driver connect to ANY AP when SSID is null.

Value:

- 0: disable (default)
- 1: enable

2.2.51 HT_40MHZ_INTOLERANT

Enable/Disable 40MHz channel bandwidth operation and also indicate other 20/40BSS Coex

Value:

- 0:Disable (default)
- 1:Enable

2.2.52 AntGain

Define peak antenna gain (dBi) for Single SKU setting.

Value:

- 0: Disable Single SKU TxPower Adjustment.
- 1~255: Enable Single SKU TxPower Adjustment.

2.2.53 BandedgeDelta

Define delta conducted power value which can pass bandedge of FCC certification at Ch1 and Ch11 (dBm) within HT_40 Bandwidth for Single SKU setting.

Value:

- 1~255: Delta value between HT_20 and HT_40 power value.

2.2.54 HwAntDiv=Value (RT5350 only)

Use this command to enable HW Antenna Diversity.

Value:

- 0: Disable
- 1: HW RX antenna diversity
- 2: Fixed RX at Main ANT
- 3: Fixed RX at AUX ANT

2.2.55 P2P_GOIntent

Relative value between 0 and 15 used to indicate the desire of the P2P device to be the P2P Group Owner, with a larger value indicating a higher desire.

Value:

0~15: GO Intent.

2.2.56 P2P_DevName

Define P2P device name for display

Value:

0~Z, less than 32 characters.

2.2.57 P2P_ListChannel

Set P2P device listen channel. (Channel 1, 6, 11)

Value:

1, 6, 11

2.2.58 P2P_OpChannel

Set P2P device GO operate channel.

Value:

Based on country region

3 WIRELESS TOOLS

3.1 Iwpriv ra0 set use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

`iwpriv ra0 set [parameters]=[Value]`

Note: Execute one iwpriv/set command at a time.

3.1.1 DriverVersion

Check driver version by issue iwpriv set command.

Range:

Any value

Value:

0

3.1.2 CountryRegion

Set country region.

Range:

{0~7}

Value:

Region	Channels
0	1-11
1	1-13
2	10-11
3	10-13
4	14
5	1-14
6	3-9
7	5-13
31	1-14
32	1-11 active scan, 12 and 13 passive scan
33	1-14 all active scan, 14 b mode only

3.1.3 CountryRegionABand

Set country region for A band.

Range:

{0~9}

Value:

Region	Channels
0	36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, 161, 165
1	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
2	36, 40, 44, 48, 52, 56, 60, 64
3	52, 56, 60, 64, 149, 153, 157, 161
4	149, 153, 157, 161, 165
5	149, 153, 157, 161
6	36, 40, 44, 48
7	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 165
8	52, 56, 60, 64
9	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 132, 136, 140, 149, 153, 157, 161, 165
10	36, 40, 44, 48, 149, 153, 157, 161, 165
11	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 149, 153, 157, 161

3.1.4 SSID

Set AP SSID

Range:

{0~z, 1~32 ascii characters}

Value:

3.1.5 WirelessMode

Set Wireless Mode

Range:

{0~10}

Value:

- 0: legacy 11b/g mixed
- 1: legacy 11B only
- 2: legacy 11A only
- 3: legacy 11a/b/g mixed
- 4: legacy 11G only
- 5: 11ABGN mixed
- 6: 11N only
- 7: 11GN mixed
- 8: 11AN mixed
- 9: 11BGN mixed
- 10: 11AGN mixed
- 11: 11N only in 5G band only

3.1.6 TxBurst:

Set TxBurst Enable or Disable

Range:

{0,1}

Value:

- 0:Disable,
- 1:Enable

3.1.7 PktAggregate:

Set Tx Aggregate Enable or Disable

Range:

{0,1}

Value:

- 0:Disable,
- 1:Enable

3.1.8 TxPreamble:

Set TxPreamble

Range:

{0~2}

Value:

- 0:Preamble Long,
- 1:Preamble Short,
- 2:Auto

3.1.9 TxPower:

Set Tx power in percentage

Range:

$\{0 \sim 100\}$

Value:

3.1.10 Channel

Set Channel, depends on CountryRegion or CountryRegionABand

3.1.11 BGProtection:

Set 11B/11G Protection

Range:

 $\{0 \sim 2\}$

Value:

0:Auto,
1:Always on,
2:Always off

3.1.12 RTSThreshold:

Set RTS Threshold

Range:

 $\{1 \sim 2347\}$

Value:

3.1.13 FragThreshold:

Set Fragment Threshold

Range:

{256~2346}

Value:

3.1.14 NetworkType:

Set Network type

Range:

{Infra,Adhoc}

Value:

3.1.15 AuthMode:

Set Authentication Mode

Range:

```
{OPEN,SHARED,WEPAUTO,WPA PSK,WPA2PSK,WPA NONE}
```

Value:

3.1.16 EncryptType:

Set Encryption Type

Range:

{NONE,WEP,TKIP,AES}

Value:

3.1.17 DefaultKeyID:

Set Default Key ID

Range:

{1~4}

Value:

3.1.18 Key1

Set Key1 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

3.1.19 Key2

Set Key2 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

3.1.20 Key3

Set Key3 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

3.1.21 Key4

Set Key4 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

3.1.22 WPAPSK

WPA Pre-Shared Key

Range:

{8~63 ascii or 64 hex characters}

Value:

3.1.23 WmmCapable

Set WMM Capable

Range:

{0,1}

Value:

0:Disable WMM,
1:Enable WMM

3.1.24 IEEE80211H

Enabel IEEE802.11h support

Range:

{0,1}

Value:

0:Disable

1:Enable

3.1.25 PSMode

Set Power Saving Mode

Range:

{CAM, MAX_PSP, FAST_PSP}

Value:

3.1.26 ResetCounter

Reset statistics counter

Range:

Any vlaue

Value:

0

3.1.27 Debug

Set on debug level

Range:

{0 ~ 5}

Value:

- 0: OFF no debug message display
- 1: ERROR display error message
- 2: WARN display warning message
- 3: TRACE display trace message, usually used.
- 4: INFO display informatic message
- 5: LOUD display all message

3.1.28 CarrierDetect

Value

0: Disabled

1: Enabled

3.1.29 HtRdg

Enable HT Reverse Direction Grant.

Value

0: Disabled

1: Enabled

3.1.30 HtExtcha

To locate the 40MHz channel in combination with the control.

Value

- 0: Below
- 1: Above

3.1.31 HtOpMode

Change HT operation mode.

Value

- 0: HT mixed format
- 1: HT greenfield format

3.1.32 HtMpduDensity

Minimum separation of MPDUs in an A-MPDU.

Value

- 0 ~ 7
- 0: no restriction
- 1: 1/4 μ s
- 2: 1/2 μ s
- 3: 1 μ s
- 4: 2 μ s
- 5: 4 μ s
- 6: 8 μ s
- 7: 16 μ s

3.1.33 HtBw

Support channel width.

Value

- 0: 20MHz
- 1: 40MHz

3.1.34 HtAutoBa

Enable auto block acknowledgment (Block Ack).

Value

- 0: Disabled
- 1: Enabled

3.1.35 HtAmsdu

Enable aggregation of multiple MSDUs in one MPDU.

Value

- 0: Disabled
- 1: Enabled

3.1.36 HtBaWinSize

Set BA WinSize.

Value
1 ~ 64

3.1.37 HtGi

Support Short/Long GI.

Value
0: long GI
1: short GI

3.1.38 HtMcs

MCS rate selection.

Value
0 ~ 15
33: auto

3.1.39 HtProtect

Enable HT protection for legacy device.

Value
0: Disable
1: Enable

3.1.40 HtMimoPs

MIMO power save.

Value
0: Disable
1: Enable

3.1.41 FixedTxMode

Set Fixed Tx Mode for fixed rate setting

Value

Mode = CCK		
MCS= 0	=>	1Mbps
MCS= 1	=>	2Mbps
MCS= 2	=>	5.5 Mbps
MCS= 3	=>	11 Mbps
Mode = OFDM		
MCS= 0	=>	6Mbps
MCS= 1	=>	9Mbps
MCS= 2	=>	12Mbps
MCS= 3	=>	18Mbps
MCS= 4	=>	24Mbps
MCS= 5	=>	36Mbps
MCS= 6	=>	48Mbps
MCS= 7	=>	54Mbps

3.1.42 LongRetry

USE:
iwpriv ra0 set LongRetry=value
Value:
0~255

3.1.43 ShortRetry

USE:
iwpriv ra0 set ShortRetry=value
Value:
0~255

3.1.44 HtTxStream=value

Value:
1: Support 1-Tx Stream for MCS0 ~ MCS7
2: Support 2-Tx Stream for MCS0 ~ MCS15

3.1.45 HtRxStream=value

Value:
1: Support 1-Rx Stream for MCS0 ~ MCS7
2: Support 2-Rx Stream for MCS0 ~ MCS15

3.1.46 HtDisallowTKIP=value

Enable/Disable N rate with 11N ap when cipher is WEP or TKIP.

Value:
0: FALSE (Default)
1: TRUE

3.1.47 HtBaDecline

Reject all Recipient's BA requests.

Value:
0: Disable (Default)
1: Enable

3.1.48 BeaconLostTime=value

Change Beacon Lost Time

Value:
1 ~ 60 seconds
Default value is 4 seconds

3.1.49 AutoRoaming=value

Enable/disable auto roaming mechanism

Value:

0: disable (Default)

1: enable

3.1.50 SiteSurvey=value

Scan with specific SSID after link up

Value:

0~z, 1~32 ascii characters

3.1.51 TdlsCapable=value

Enable/disable TDLS capable

Value:

0: disable

1: enable

Example: iwpriv ra0 set TdlsCapable=0

3.1.52 TdlsSetup=value

Manually add TDLS link

Value: MAC address

Example: iwpriv ra0 set TdlsSetup=00:11:22:33:44:55

3.1.53 AutoReconnect=value

Description: Enable/Disable driver auto reconnect functionality

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

3.1.54 AdhocN=value

Description: Enable/Disable Adhoc to support N or not

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

3.1.55 AntGain

Define peak antenna gain (dBi) for Single SKU setting.

Value:

0: Disable Single SKU TxPower Adjustment.

1~255: Enable Single SKU TxPower Adjustment.

3.1.56 HwAntDiv=Value (RT5350 only)

Use this command to enable HW Antenna Diversity.

Value:

- 0: Disable
- 1: HW RX antenna diversity
- 2: Fixed RX at Main ANT
- 3: Fixed RX at AUX ANT

3.2 iwpriv ra0 show use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

A detailed explanation of each parameter for iwpriv is shown subsequently. Refer to the Readme before using this section. `iwpriv ra0 show [parameters]`

3.2.1 connStatus

Show STA connection Status

3.2.2 driverVer

Show STA current driver version

3.2.3 bainfo

Show STA current BA information

3.2.4 rxbulk

Show STA current rxblk information

3.2.5 txbulk

Show STA current txblk information

3.2.6 AutoReconnect

Show bAutoReconnect flag

3.2.7 WPAPSK

Show WPA Passphrase

3.2.8 PMK

Show PMK key

*MEDIA TEK CONFIDENTIAL
FOR panwl@laketune.com.cn USE ONLY*

3.3 iwpriv ra0 use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

```
iwpriv ra0 show [parameters]
```

3.3.1 radio_off

Turn STA radio off

3.3.2 radio_on

Turn STA radio on

3.4 Iwpriv Examples

3.4.1 Infrastructure

1.1.1.1 OPEN/NONE

Config STA to link with AP which is OPEN/NONE(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=OPEN
3. iwpriv ra0 set EncrypType=NONE
4. iwpriv ra0 set SSID="AP's SSID"

1.1.1.2 SHARED/WEP

Config STA to link with AP which is SHARED/WEP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=SHARED
3. iwpriv ra0 set EncrypType=WEP
4. iwpriv ra0 set DefaultKeyID=1
5. iwpriv ra0 set Key1="AP's wep key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.3 WPAPSK/TKIP

Config STA to link with AP which is WPAPSK/TKIP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPAPSK
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK="AP's wpa-preshared key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.4 WPAPSK/AES

Config STA to link with AP which is WPAPSK/AES(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPAPSK
3. iwpriv ra0 set EncrypType=AES
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK="AP's wpa-preshared key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.5 WPA2PSK/TKIP

Config STA to link with AP which is WPA2PSK/TKIP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPA2PSK
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK=12345678
6. iwpriv ra0 set SSID="AP's SSID"

3.4.2 Ad-Hoc

1.1.1.6 OPEN/NONE

Config STA to create/link as adhoc mode, which is OPEN/NONE(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Adhoc
2. iwpriv ra0 set AuthMode=OPEN
3. iwpriv ra0 set EncrypType=NONE
4. iwpriv ra0 set SSID="Adhoc's SSID"

1.1.1.7 WPANONE/TKIP

Config STA to create/link as adhoc mode, which is WPANONE/TKIP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Adhoc
2. iwpriv ra0 set AuthMode=WPANONE
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK=12345678
6. iwpriv ra0 set SSID="AP's SSID"

3.4.3 Get site survey

use:

```
iwpriv ra0 get_site_survey
```

3.4.4 Get Statistics

use:

```
iwpriv ra0 stat ; read statistic counter
iwpriv ra0 set ResetCounter=0 ; reset statistic counter
```

3.4.5 ANY SSID

Link with an AP which is the largest strength, set ANY SSID (ssidLen=0)

use:

```
iwconfig ra0 essid ""
or
iwpriv ra0 set SSID=""
```

3.5 iwlist

This section describes parameters set using iwlist. Please refer to the Readme section for more general data.

iwlist ra0 scanning - list the results after scanning(manual rescan)

3.6 iwconfig

The subsequent settings are used in the standard iwconfig configuration

- 1) iwconfig ra0 essid {NN|on|off} ; set essid

- | | |
|---|------------------------------|
| 2) iwconfig ra0 mode {managed ad-hoc ...} | ; set wireless mode |
| 3) iwconfig ra0 freq N.NNNN[k M G] | ; set frequency |
| 4) iwconfig ra0 channel N | ; set channel |
| 5) iwconfig ra0 ap {N off auto} | ; set AP address |
| 6) iwconfig ra0 nick N | ; set nickname |
| 7) iwconfig ra0 rate {N auto fixed} | ; set rate |
| 8) iwconfig ra0 rts {N auto fixed off} | ; set RTS threshold |
| 9) iwconfig ra0 frag {N auto fixed off} | ; set Fragment threshold |
| 10) iwconfig ra0 enc {NNNN-NNNN off} | ; set encryption type |
| 11) iwconfig ra0 power {period N timeout N} | ; set power management modes |

Note: Refer to the 'iwconfig', 'iwlist' and 'iwpriv' sections for wireless extension instructions.

4 WPS – WI-FI PROTECTED SETUP

Simple Config Architectural Overview

This section presents a high-level description of the Simple Config architecture. Much of the material is taken directly from the Simple Config specification.

Figure 1 depicts the major components and their interfaces as defined by Wi-Fi Simple Config Spec. There are three logical components involved: the Registrar, the access point (AP), and the Enrollee.

- The **Enrollee** is a device seeking to join a WLAN domain. Once an Enrollee obtains a valid credential, it becomes a member.
- A **Registrar** is an entity with the authority to issue and revoke domain credentials. A registrar can be integrated into an AP.
- The **AP** can be either a WLAN AP or a wireless router.

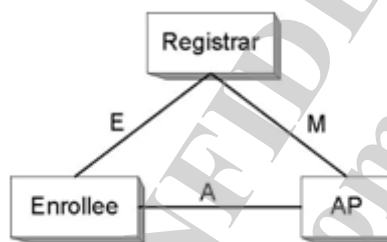


Figure 1. Components and Interfaces

Registration initiation is ordinarily accomplished by a user action such as powering up the Enrollee and, optionally, running a setup wizard on the Registrar (PC).

4.1 Iwpriv use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

`iwpriv ra0 [commands]=[Value]`

Note: Wireless extension private handlers.

4.1.1 wsc_conf_mode

Set WPS conf mode.

Range:

{0, 1, 2}

Value:

0: WPS Disabled

1: Enrollee

2: Registrar

4.1.2 wsc_mode

Set WPS mode, PIN or PBC.

Range:

{1, 2}

Value:

1: PIN

2: PBC

4.1.3 wsc_pin

Set Enrollee's PIN Code.

Range:

{00000000 ~ 99999999}

Value:

4.1.4 wsc_ssid

Set WPS AP SSID.

Range:

{0~z, 1~32 ascii characters}

Value:

4.1.5 wsc_bssid

BSSID of WSC AP that STA wants to do WPS with

Value:

xx:xx:xx:xx:xx:xx

4.1.6 wsc_start

Trigger RT3572 STA driver to do WPS process.

Range:

NULL

Value:

4.1.7 wsc_stop

Stop WPS process and don't wait upon two-minute timeout.

Range:

NULL

Value:

4.1.8 wsc_gen_pincode

Generate new PIN code.

Range:

NULL

Value:

4.1.9 wsc_cred_count

Set count of WPS credential, only support one credential for M8 in Registrar mode.

Range:

{1 ~ 8}

Value:

4.1.10 wsc_cred_ssid

Set SSID into credentail[idx].

Range:
{"idx ssid_str"}
Value:
idx: 0 ~ 7
ssid_str: 0~z, 1~32 ascii characters
Example:
iwpriv ra0 wsc_cred_ssid "0 wps_ap1"

4.1.11 wsc_cred_auth

Set AuthMode into credtentail[idx].
Range:
{"idx auth_str"}
Value:
idx: 0 ~ 7
auth_str: OPEN, WPAPSK, WPA2PSK, SHARED, WPA, WPA2
Example:
iwpriv ra0 wsc_cred_auth "0 WPAPSK"

4.1.12 wsc_cred_encr

Set EncryptType into credtentail[idx].
Range:
{"idx encr_str"}
Value:
idx: 0 ~ 7
encr_str: NONE, WEP, TKIP, AES
Example:
iwpriv ra0 wsc_cred_encr "0 TKIP"

4.1.13 wsc_cred_keyIdx

Set Key Index into credtentail[idx].
Range:
{"idx key_index"}
Value:
idx: 0 ~ 7
key_index: 1 ~ 4
Example:
iwpriv ra0 wsc_cred_keyIdx "0 1"

4.1.14 wsc_cred_key

Set Key into credtentail[idx].
Range:
{"idx key"}
Value:
idx: 0 ~ 7
key: ASCII string (wep_key_len(=5,13), passphrase_len(=8~63))
OR
Hex string (wep_key_len(=10,26), passphrase_len(=64))
Example:

```
iwpriv ra0 wsc_cred_key "0 12345678" ;; Passphrase
iwpriv ra0 wsc_cred_key "0 abcd" ;; WEP Key
```

4.1.15 wsc_cred_mac

Set AP's MAC into credtentail[idx].

Range:

{ "idx mac_str" }

Value:

idx: 0 ~ 7

mac_str: xx:xx:xx:xx:xx:xx

Example:

```
iwpriv ra0 wsc_cred_mac "0 00:11:22:33:44:55"
```

4.1.16 wsc_conn_by_idx

Connect AP by credential index.

Range:

{ 0 ~ 7 }

Value:

idx: 0 ~ 7

4.1.17 wsc_auto_conn

If the registration is successful, driver will re-connect to AP or not.

Range:

{ 0, 1 }

Value:

0: Disabled, driver won't re-connect to AP with new configurations.

1: Enabled, driver will re-connect to AP with new configurations.

4.1.18 wsc_ap_band

Setting prefer band to do WPS with dual band WPS AP.

Range:

{ 0, 1, 2 }

Value:

0: prefer 2.4G

1: prefer 5G

2: auto

Default value is auto (2)

4.1.19 Wsc4digitPinCode

Generate WPS 4-digits PIN

Value:

0: Disable

1: Enable

4.1.20 WscV2Support (WPS2.0)

Enable/Disable WSC V2 support

Value:

0: Disable

1: Enable

4.2 WPS STA as an Enrollee or Registrar

Build WPS function. Please set the "HAS_WSC" parameter value to "y".

4.2.1 Enrollee Mode

1.1.1.8 PIN mode

Running Scenarios (case 'a' and 'b')

- a. Adding an Enrollee to AP+Registrar (EAP)
[AP+Registrar]<----EAP-->[Enrollee Client]
- b. Adding an Enrollee with external Registrar (UPnP/EAP)
[External Registrar]<----UPnP-->[AP_Proxy]<----EAP-->[Enrollee Client]

Note:

'EAP' indicates to use wireless medium and 'UPnP' indicates to use wired or wireless medium.

- (i) [Registrar] or [AP+Registrar]
Enter the Enrollee PinCode on the Registrar and start WPS on the Registrar.
Note:
How to get the Enrollee PinCode? Use 'iwpriv ra0 stat' on the Enrollee.

- (ii) [RT3572 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 1 ;; Enrollee
iwpriv ra0 wsc_mode 1 ;; PIN
iwpriv ra0 wsc_ssid "AP's SSID"
iwpriv ra0 wsc_start

- (iii) If the registration is successful, the Enrollee will be re-configured with the new parameters, and will connect to the AP with these new parameters.

1.1.1.9 PBC mode

Running Scenarios (case 'a' only)

- a. Adding an Enrollee to AP+Registrar (EAP)
[AP+Registrar]<----EAP-->[Client]
- (i) [AP+Registrar]
Start PBC on the Registrar.
- (ii) [RT3572 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 1 ;; Enrollee

- ```
iwpriv ra0 wsc_mode 2 ;; PBC
iwpriv ra0 wsc_start
```
- (iii) If the registration is successful, the Enrollee will be re-configured with the new parameters, and will connect to the AP with these new parameters.

## 4.2.2 Registrar Mode

### 1.1.1.10 PIN mode

Running Scenarios (case 'a' and 'b')

- a. Configure the un-configured AP  
[Unconfigured AP]<----EAP-->[Registrar]
  - b. Configure the configured AP  
Configured AP]<----EAP-->[Registrar]
- (i) [AP]  
Start PIN on the Enrollee WPS AP.
- (ii) [RT3572 Linux WPS STA]  

```
iwpriv ra0 wsc_conf_mode 2 ;; Registrar
iwpriv ra0 wsc_mode 1 ;; PIN
iwpriv ra0 wsc_pin xxxxxxxx ;; AP's PIN Code
iwpriv ra0 wsc_ssid "AP's SSID"
iwpriv ra0 wsc_start
```
- (iii) If the registration is successful;
- in case 'a':  
The Registrar will be re-configured with the new parameters, and will connect to the AP with these new parameters;
- in case 'b':  
The Registrar will be re-configured with AP's configurations, and will connect to the AP with these new parameters.

### 1.1.1.11 PBC mode

Running Scenarios (case 'a' and 'b')

- a. Configure the un-configured AP  
[Unconfigured AP]<----EAP-->[Registrar]
  - b. Configure the configured AP  
Configured AP]<----EAP-->[Registrar]
- (i) [AP]  
Start PBC on the Enrollee WPS AP.
- (ii) [RT3572 Linux WPS STA]  

```
iwpriv ra0 wsc_conf_mode 2 ;; Registrar
iwpriv ra0 wsc_mode 2 ;; PBC
iwpriv ra0 wsc_start
```
- (iii) If the registration is successful;
- in case 'a':  
The Registrar will be re-configured with the new parameters, and will connect to the AP with these new parameters;
- in case 'b':



The Registrar will be re-configured with AP's configurations, and will connect to the AP with these new parameters.

### 4.3 WPS IOCTL use

*This section describes specific parameters and arguments. Please refer to the previous section for more general data.*

#### 4.3.1 iwpriv commands without argument

1. iwpriv ra0 wsc\_start
2. iwpriv ra0 wsc\_stop
3. iwpriv ra0 wsc\_gen\_pincode

**Example:**

```
memset(&lwreq, 0, sizeof(lwreq));
sprintf(lwreq.ifr_name, "ra0", 3);
lwreq.u.mode = WSC_STOP;

/* Perform the private ioctl */
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
{
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
 return -1;
}
```

#### 4.3.2 iwpriv commands with one INT argument

1. iwpriv ra0 wsc\_cred\_count 1
2. iwpriv ra0 wsc\_conn\_by\_idx 1
3. iwpriv ra0 wsc\_auto\_conn 1
4. iwpriv ra0 wsc\_conf\_mode 1
5. iwpriv ra0 wsc\_mode 1
6. iwpriv ra0 wsc\_pin 12345678

**Example:**

```
memset(&lwreq, 0, sizeof(lwreq));
lwreq.u.data.length = 1;
cred_count = 1;
((int *) buffer)[i] = (int) cred_count;
offset = sizeof(int);

sprintf(lwreq.ifr_name, "ra0", 3);
lwreq.u.mode = WSC_CREDENTIAL_COUNT;
memcpy(lwreq.u.name + offset, buffer, IFNAMSIZ - offset);

/* Perform the private ioctl */
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
{
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
 return -1;
}
```

#### 4.3.3 iwpriv commands with string argument

1. iwpriv ra0 wsc\_ssid "0 xxxxx"
2. iwpriv ra0 wsc\_cred\_ssid "0 xxxxx"
3. iwpriv ra0 wsc\_cred\_auth "0 WPAESK"
4. iwpriv ra0 wsc\_cred\_encr "0 TKIP"
5. iwpriv ra0 wsc\_cred\_keyidx "0 1"
6. iwpriv ra0 wsc\_cred\_key "0 12345"
7. iwpriv ra0 wsc\_cred\_mac "0 00:11:22:33:44:55"

**Example:**

```
memset(&lwreq, 0, sizeof(lwreq));
memset(buffer, 0, 2048);
sprintf(lwreq.ifr_name, "ra0", 3);
sprintf(buffer, "0 wps_ssid_1");
lwreq.u.data.length = strlen(buffer) + 1;
lwreq.u.data.pointer = (caddr_t) buffer;
lwreq.u.data.flags = WSC_CREDENTIAL_SSID;

/* Perform the private ioctl */
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM, &lwreq) < 0)
{
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
 return -1;
}
```

#### 4.4 WPS IOCTL Sample Program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <netinet/in.h> /* for sockaddr_in */
#include <fcntl.h>
#include <time.h>
#include <sys/times.h>
#include <unistd.h>
#include <sys/socket.h> /* for connect and socket*/
#include <sys/stat.h>
#include <err.h>
#include <errno.h>
#include <asm/types.h>
#include </usr/include/linux/wireless.h>
#include <sys/ioctl.h>

#define IFNAMSIZ 16

#define RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM (SIOCWFIRSTPRIV + 0x14)
#define RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM (SIOCWFIRSTPRIV + 0x16)

enum {
 WSC_CREDENTIAL_COUNT = 1,
 WSC_CREDENTIAL_SSID = 2,
 WSC_CREDENTIAL_AUTH_MODE = 3,
 WSC_CREDENTIAL_ENCR_TYPE = 4,
 WSC_CREDENTIAL_KEY_INDEX = 5,
 WSC_CREDENTIAL_KEY = 6,
 WSC_CREDENTIAL_MAC = 7,
 WSC_SET_DRIVER_CONNECT_BY_CREDENTIAL_IDX = 8,
 WSC_SET_DRIVER_AUTO_CONNECT = 9,
 WSC_SET_CONF_MODE = 10, // Enrollee or Registrar
 WSC_SET_MODE = 11, // PIN or PBC
 WSC_SET_PIN = 12,
 WSC_SET_SSID = 13,
 WSC_START = 14,
 WSC_STOP = 15,
 WSC_GEN_PIN_CODE = 16,
};

int main()
{
 struct iwreq lwreq;
 char buffer[2048] = {0};
 int cred_count;
 int offset = 0; /* Space for sub-ioctl index */
 int skfd, i = 0; /* generic raw socket desc. */

 skfd = socket(AF_INET, SOCK_DGRAM, 0);
 if (skfd < 0)
 return -1;

 //////////// WSC_STOP ////////////
 memset(&lwreq, 0, sizeof(lwreq));
 sprintf(lwreq.ifr_name, "ra0", 3);
 lwreq.u.mode = WSC_STOP;

 /* Perform the private ioctl */
 if (ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
 {
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
 return -1;
 }
}
```

```
////////////////////////////////////
```

```
//////// WSC_CREDENTIAL_COUNT //////////
```

```
memset(&lwreq, 0, sizeof(lwreq));
```

```
lwreq.u.data.length = 1;
```

```
cred_count = 1;
```

```
((int *) buffer)[i] = (int) cred_count;
```

```
offset = sizeof(int);
```

```
sprintf(lwreq.ifr_name, "ra0", 3);
```

```
lwreq.u.mode = WSC_CREDENTIAL_COUNT;
```

```
memcpy(lwreq.u.name + offset, buffer, IFNAMSIZ - offset);
```

```
/* Perform the private ioctl */
```

```
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
```

```
{
```

```
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
```

```
 return -1;
```

```
}
```

```
////////////////////////////////////
```

```
//////// WSC_CREDENTIAL_SSID //////////
```

```
memset(&lwreq, 0, sizeof(lwreq));
```

```
memset(buffer, 0, 2048);
```

```
sprintf(lwreq.ifr_name, "ra0", 3);
```

```
sprintf(buffer, "0 wps_ssid_1");
```

```
lwreq.u.data.length = strlen(buffer) + 1;
```

```
lwreq.u.data.pointer = (caddr_t) buffer;
```

```
lwreq.u.data.flags = WSC_CREDENTIAL_SSID;
```

```
/* Perform the private ioctl */
```

```
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM, &lwreq) < 0)
```

```
{
```

```
 fprintf(stderr, "Interface doesn't accept private ioctl...\n");
```

```
 return -1;
```

```
}
```

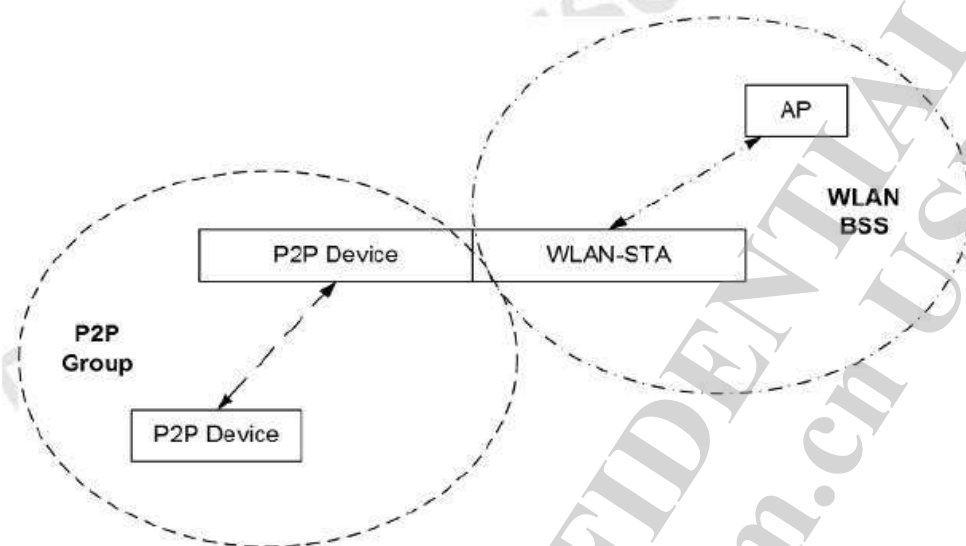
```
////////////////////////////////////
```

```
close(skfd);
```

```
return 0;
```

```
}
```

## 5 WIFI DIRECT - P2P COMMAND



- A Wi-fi Direct Device may operate concurrently with a WLAN (infrastructure network)
- A P2P Group may operate in the same or different regulatory class and channel as a concurrently operating WLAN BSS

Wifi direct feature Makes direct connections to one another quickly and conveniently to do things like print, sync, and share content even when an access point or router is unavailable.

### 5.1 Iwpriv use

#### 5.1.1 P2pOpMode

Set p2p interface operate mode to GO.

Value:

1: Auto (Force) GO mode

Example:

```
#iwpriv p2p0 set P2pOpMode=1
```

#### 5.1.2 p2pLisCh

Set p2p device Channel in Listen stage.

Value:

1, 6, 11 (Define in P2P spec Page 26 & 36)

Example:

```
#iwpriv p2p0 set P2pLisCh=x
```

#### 5.1.3 p2pOpCh

Set p2p Operation Channel if negotiate as GO

Value:

Based on country region

Example:

```
iwpriv p2p0 set p2pOpCh=1
```

#### 5.1.4 p2pGoInt

Set p2p device GO Intent value

- This value is set to nego the art for become GO or Client

x1 = Group Owner Intent Value of P2P Device 1  
x2 = Group Owner Intent Value of P2P Device 2

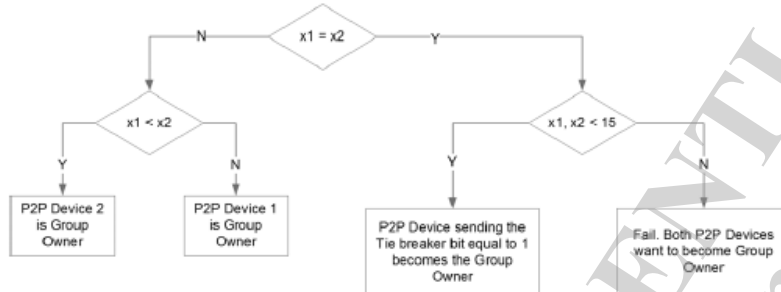


Figure 7—Group Owner determination flowchart

Value:

0~15

Example:

#iwpriv p2p0 set p2pGoInt=x

#### 5.1.5 p2pDevName

Set p2p device display Device Name

Value:

0~Z, less than 32 characters.

Example:

#iwpriv p2p0 set p2pDevName=xxxxx

#### 5.1.6 p2pWscMode

Set p2p device WSC Mode for P2P negotiate.

Value:

1: PIN

2: PBC

Example:

#iwpriv p2p0 set p2pWscMode=x

#### 5.1.7 p2pWscConf

Set p2p device WSC Configure Method

Value:

1: Display

2: KeyPad

3: PBC

Example:

#iwpriv p2p0 set p2pWscConf =x

#### 5.1.8 p2pScan

Set p2p device to start P2P Scanning.

Value:

0: Disable (Force Stop this action and cancel timer)

1: Enable (Do action)

Example:

```
#iwpriv p2p0 set p2pScan=x
```

---

#### 5.1.9 p2pInv

Select p2p device ID to Invite (send p2p Invite to peer p2p device join our p2p Group)

Value:

0~29 (Software setting)

Example:

```
#iwpriv p2p0 set p2pInv =x
```

---

#### 5.1.10 p2pDevDisc

Send p2p device discoverability to GO query specific CLIENT is in Group or not(debug use)

Value:

0~29 (Software setting)

Example:

```
#iwpriv p2p0 set p2pDevDisc=x
```

---

#### 5.1.11 p2pLink

Select p2p device ID to do GO Negotiation

Value:

0~29 (Software setting)

Example:

```
#iwpriv p2p0 set p2pLink =x
```

---

#### 5.1.12 p2pCfg

Dump/Show p2p configuration (In kernel background message)

Value:

Any

Example:

```
#iwpriv p2p0 set p2pCfg
```

---

#### 5.1.13 p2pTab

Show Group Table status in kernel background message.

Value:

Any

Example:

```
#iwpriv p2p0 set p2pTab
```

---

#### 5.1.14 p2pProv

Select p2p device ID to Provision

Value:

0~29 (Software setting)

Example:

```
#iwpriv p2p0 set p2pProv=x
```

---

#### 5.1.15 p2pStat

Dump/Show p2p current rule, state machine status(In kernel background message)

Value:

Any

Example:

```
#iwpriv p2p0 set p2pStat
```

---

#### 5.1.16 p2pReset

Reset p2p configuration the stat machine set to initial stage.

Value:

Any

Example:

```
#iwpriv p2p0 set p2pReset
```

---

#### 5.1.17 p2pPerTab

Show p2p Persistent Table in kernel background message.

Value:

Any

Example:

```
#iwpriv p2p0 set p2pPerTab
```

---

#### 5.1.18 p2pDefConfMthd

Set default WSC Config Method to Provision

Value:

1: Display

2: KeyPad

3: PBC

Example:

```
#iwpriv p2p0 set p2pDefConfMthd=x
```

---

#### 5.1.19 p2pLinkDown

Tear down p2p session and change as p2p device mode.

Value:

Any

Example:

```
#iwpriv p2p0 set p2pLinkDown=x
```

---

#### 5.1.20 p2pSigmaEnable

For p2p Sigma auto testing, we need enable some flag to pass WPS.

Value:

Any

Example:

```
#iwpriv p2p0 set p2pSigmaEnable=x
```

---

#### 5.1.21 Other P2P command

The other command wrote in source code which is for debug only. Please ignore them.



### 5.1.22 P2P example:

#### **P2P device enable as autonomous GO :**

autonomous GO:

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set P2pOpMode=1
```

#### **P2P device reset to default setting:**

```
#iwpriv p2p0 set p2pReset=1
```

#### **P2P device start to scan and Listen as Channel 11:**

```
#iwpriv p2p0 set p2pLisCh=11
#iwpriv p2p0 set p2pScan=1
```

#### **P2P Device Start Device Discovery.**

```
#iwpriv p2p0 set p2pScan=1
```

#### **P2P Device Stop Device Discovery.**

```
#iwpriv p2p0 set p2pScan=0
```

#### **Connect the P2P Device on Scan Table of index 0.**

```
Table.Client[0]: DeviceName[R A L I N K - P C]
 DevCapability = 32.
 ServiceDiscovery = 0. P2P Client Discoverability = 0
 ConcurrentOperation = 0. InfraManaged = 0.
 DeviceLimit = 0. InvitationProcedure = 1.
 GroupCapability = 8.
 GroupOwner = 0. PersistentP2PGroup = 0
 GroupLimit = 0. IntraBSS = 1.
 CrossConnection = 0. PersistentReconnect = 0.
 GroupFormation = 0.
 Addr[00:0c:43:21:64:81]
 BSSID[00:0c:43:21:64:81]
 InterfaceAddr[00:0c:43:21:64:81]
 SSID[D I R E C T - 1 2]
 WscMode = PBC. PIN = 00 00 00 00 00 00 00 00.
 CfgMethod = PBC PBC. GoIntent = 6.
 PrimDevType[00 01 00 50 f2 04 00 01] SecDevType[00 00 00 00 00 00 00 00]
 ChNumber = 0. OpChannel = 1. ListenChannel = 1.
 P2pClientState = P2PSTATE_GO_WPS. MyGOIndex = 255.
 P2pIP = 00000000. P2pFlag = 2.
 Dpid = 65535. StateCount = 1. Rule = I am P2P GO.
 GeneralToken = 76. NoAToken = 0.
 RegClass = 0. ConfigTimeOut = 0.
 ExtListenPeriod = 0. ExtListenInterval = 0.
```

#### **PIN:**

```
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pLink=0
```

#### **PBC:**

```
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pLink=0
```

#### **P2P scan and select p2p device do link:**

```
#iwpriv p2p0 set p2pScan=1
#sleep 10
#iwpriv p2p0 set p2pTab=1
#iwpriv p2p0 set p2pLink=2
```

#### P2P device Show P2P Scan Table:

```
#iwpriv p2p0 set p2pTab=1
```

```
Table.Client[0]: DeviceName[R A L I N K - P C]
 DevCapability = 32.
 ServiceDiscovery = 0. P2P Client Discoverability = 0
 ConcurrentOperation = 0. InfraManaged = 0.
 DeviceLimit = 0. InvitationProcedure = 1.
 GroupCapability = 8.
 GroupOwner = 0. PersistentP2PGroup = 0
 GroupLimit = 0. IntraBSS = 1.
 CrossConnection = 0. PersistentReconnect = 0.
 GroupFormation = 0.
 Addr[00:0c:43:21:64:81]
 BSSID[00:0c:43:21:64:81]
 InterfaceAddr[00:0c:43:21:64:81]
 SSID[D I R E C T - 1 2]
 WscMode = PBC. PIN = 00 00 00 00 00 00 00 00.
 CfgMethod = PBC PBC. GoIntent = 6.
 PrimDevType[00 01 00 50 f2 04 00 01] SecDevType[00 00 00 00 00 00 00 00]
 ChNumber = 0. OpChannel = 1. ListenChannel = 1.
 P2pClientState = P2PSTATE_GO_WPS. MyGOIndex = 255.
 P2pIP = 00000000. P2pFlag = 2.
 Dpid = 65535. StateCount = 1. Rule = I am P2P GO.
 GeneralToken = 76. NoAToken = 0.
 RegClass = 0. ConfigTimeOut = 0.
 ExtListenPeriod = 0. ExtListenInterval = 0.
```

#### P2P device Show P2P configuration:

```
#iwpriv p2p0 set p2pCfg
```

```
P2P Device Config :
=====
Device Name[9] = P2P-linux.
Device Addr = 00:0c:43:32:40:45.
OpChannel = 1. Listen Channel = 11.
My Go Intent = 0.
WscMode = PBC. ConfigMethod = PBC PBC.
SSID[9] = DIRECT-12.
NoA_Count = 1. NoA_Duration = 51200. NoA_Interval = 102400. StartTime = 0.
ExtListenPeriod = 0. ExtlistenInterval = 0.
Intra-Bss = 0.
ConenctMAC = 00:00:00:00:00:00.
```

#### P2P Show current rule and state machine status

```
#iwpriv p2p0 set p2pStat
```

```
P2P Current State
=====
My Rule = I am P2P Client
CTRL Machine State = P2P_CTRL_IDLE.
DISC Machine State = P2P_DISC_IDLE.
GO_FORM Machine State = P2P_GO_FORM_IDLE.
```

#### P2P device GO security setting change:

```
#iwpriv p2p0 set p2pReset=1
#iwpriv p2p0 set p2pOpCh=1
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pWscConf=3
#iwpriv p2p0 set p2pDevName=Ralink-P2P-Device
#iwpriv p2p0 set SSID=DIRECT- Ralink
#iwpriv p2p0 set AuthMode=WPA2PSK
#iwpriv p2p0 set EncrypType=AES
#iwpriv p2p0 set WPAPSK=12345678
iwpriv p2p0 set SSID=DIRECT- Ralink
#iwpriv p2p0 set p2pScan=1
```

Ralink P2P module provides three WPS configuration methods such as PBC, PIN-Display, PIN-Keypad.

**Case 1: Enable autonomous GO on Channel 11 start WPS (PBC):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pWscConf=3
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=2
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

**Case 2: Enable autonomous GO on Channel 11 start WPS (PIN-Display):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=1
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

**Case 3: Enable autonomous GO on Channel 11 start WPS (PIN-Keypad):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=2
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set p2pLink=0 (The index on P2P Scan Table)
#iwpriv p2p0 set WscPinCode=12345670 (read from enrollee's PIN Code)
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

**P2P device GO Negotiation as GO or CLIENT:****Case 1: To Do P2P GO Negotiation start WPS (PBC):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pWscConf=3
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=2
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

**Case 2: To Do P2P GO Negotiation start WPS (PIN-Display):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=1
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

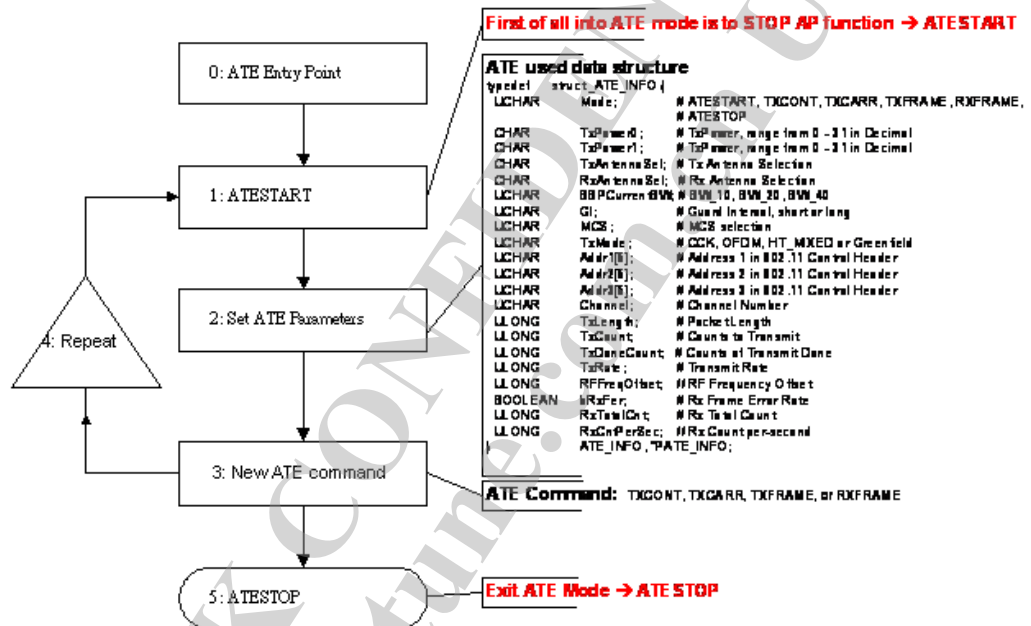
**Case 3: To Do P2P GO Negotiation start WPS (PIN-Keypad):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=2
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set p2pLink=0 (The index on P2P Scan Table)
#iwpriv p2p0 set WscPinCode=12345670 (read from enrollee's PIN Code)
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

## 6 ATE TEST COMMAND FORMAT

IF YOU ARE NOT FAMILIAR WITH HARDWARE, IT IS RECOMMENDED NOT TO MODIFY HARDWARE  
DEFAULT VALUE.

### Ralink ATE Operation Flow



#### Note:

- Channel setting would take effect on next ATE command.
- TxPower would take effect after frame transmit start.  
TxPower can be changed dynamically on any ATE command operating.
- Any ATE parameters have to be included into ATE\_INFO structure.
- Enter ATE mode by set ATE command "ATESTART".
  - Abort all TX rings
  - AsicDisableSync → Stop Beacon.
  - Stop REKEYTimer
  - Stop CounterMeasureTimer
  - MacTableReset
- Use TXCONT to check transmit power mask.
- Use TXCARR to check frequency lock (under 25ppm).

## 6.1 iwpriv ra0 set [parameters]=[val]

| Syntax:             | Example             |
|---------------------|---------------------|
| Section# parameters | 11.1.5 ATECHANNEL   |
| Value:              | Value:              |
| 0: ...              | 1: Set ATE channel. |
| 1: ...              | 2: ...              |
| ..: ...             | ..: ...             |

### 6.1.1 ATE

Description:  
Set ATE actions.

Value:

ATESTART: Enter/Reset ATE mode and set Tx/Rx Idle.

ATESTOP: Leave ATE mode.

TXCARR: Send out single carrier wave at channel frequency from hardware for frequency calibration.

TXCONT: Send out frames without time gap from hardware for power mask.

TXFRAME: Send out WIFI frames from driver, Transmit frame, for EVM.

RXFRAME: Receive all frames from MAC block, Continuous RX, for PER/FER.

TXSTOP: MAC TX disable, ONLY for QA GUI.

RXSTOP: MAC RX disable, ONLY for QA GUI.

### 6.1.2 ATEDA

Description:  
Set ATE frame header addr1.

Value:

xx:xx:xx:xx:xx:xx ; hex

### 6.1.3 ATESA

Description:  
Set ATE frame header addr2.

Value:

xx:xx:xx:xx:xx:xx ; hex

### 6.1.4 ATEBSSID

Description:  
Set ATE frame header addr3.

Value:

xx:xx:xx:xx:xx:xx ; hex

### 6.1.5 ATECHANNEL

Description:

Set ATE Channel, deimal.

Value:

802.11b/g: 1 ~ 14 depends on CountryRegion setting

#### 6.1.6 ATETXPOW0

Description:

Set ATE Tx power for Antenna 1.

Value:

0 ~ 31 ; 2.4GHz,5-bits only, deimal  
-7 ~ 15 ; 5GHz,5-bits only, deimal

#### 6.1.7 ATETXPOW1

Description:

Set ATE Tx power for Antenna 2.

Value:

0 ~ 31 ; 5-bits only, decimal  
-7 ~ 15 ; 5GHz,5-bits only, deimal

#### 6.1.8 ATETXFREQOFFSET

Description:

Set ATE RF frequency offset.

Value:

0 ~ 63 ; unit: 2KHz, decimal

#### 6.1.9 ATETXLEN

Description:

Set ATE frame length.

Value:

24 ~ 1500 ; decimal

#### 6.1.10 ATETXCNT

Description:

Set ATE frame Tx count.

Value:

1 ~ ; 32-bit, decimal

#### 6.1.11 ATETXMODE (Refer to TxMode)

Description:

Set ATE Tx Mode.

Value:

|    |        |             |
|----|--------|-------------|
| 0: | CCK    | 802.11b     |
| 1: | OFDM   | 802.11g     |
| 2: | HT_MIX | 802.11b/g/n |

---

#### 6.1.12 ATETXBW (Refer to TxMode)

Description:

Set ATE Tx and Rx Bandwidth.

Value:

- 0: 20MHz
- 1: 40MHz

---

#### 6.1.13 ATETXGI (Refer to TxMode)

Description:

Set ATE Tx Guard Interval.

Value:

- 0: Long
- 1: Short

---

#### 6.1.14 ATETXMCS (Refer to TxMode)

Description:

Set ATE Tx MCS type.

Value:

0 ~ 15

---

#### 6.1.15 ATETXANT

Description:

Set ATE TX antenna.

Value:

- 0: All
- 1: Antenna one
- 2: Antenna two

---

#### 6.1.16 ATERXANT

Description:

Set ATE RX antenna.

Value:

- 0: All
- 1: Antenna one
- 2: Antenna two
- 3: Antenna three

---

#### 6.1.17 ATERXFER

Description:

Set ATE to periodically reset and show up RxCount (per-second) and RxTotalCount.

Value:

- 0: Disable counter visibility
- 1: Enable counter visibility



---

#### 6.1.18 ATESHOW

Description:

Show all parameters of ATE.

Value:

1

---

#### 6.1.19 ATEHELP

Description:

List all commands of ATE.

Value:

1

---

#### 6.1.20 ResetCounter

Description:

Reset statistic counter.

Value:

0

---

#### 6.1.21 ATERRF

Description:

Read all of the RF registers.

Value:

1

---

#### 6.1.22 ATELDE2P

Description:

Overwrite all EEPROM contents from "/etc/Wireless/RT2860/(70)AP/(STA)/e2p.bin".

Value:

1

E.g.

iwpriv ra0 set ATELDE2P=1

---

#### 6.1.23 ATETSSICBAEX (For RT5350 only)

Description:

Write the temperature compensation reference value into EEPROM relation field (0x6E).

Value:

1

E.g.

iwpriv ra0 set ATETSSICBAEX=1

---

#### 6.1.24 ATEIPG

Description:

Set ATE Tx frame Interpacket gap.

Value:

200 ; decimal

#### 6.1.25 ATEPAYLOAD

Description:

Set ATE payload pattern for TxFrame.

Value:

x ; only one octet acceptable

#### 6.1.26 ATETSSICBA

Description:

Calibrate TSSI power delta per channel and write them into EEPROM for normal driver

Value:

xx ;8-bit, decimal,get it from e2p 0x52);  
1 ;for RT5350

### 6.2 Tx Mode, MCS, BW and GI Selection Table

| 6.2.1 MODE = 0, Legacy CCK                                                                                                                                                                                  |                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| MCS = 0                                                                                                                                                                                                     | Long Preamble CCK 1Mbps                  |
| MCS = 1                                                                                                                                                                                                     | Long Preamble CCK 2Mbps                  |
| MCS = 2                                                                                                                                                                                                     | Long Preamble CCK 5.5Mbps                |
| MCS = 3                                                                                                                                                                                                     | Long Preamble CCK 11Mbps                 |
| MCS = 8                                                                                                                                                                                                     | Short Preamble CCK 1Mbps, * illegal rate |
| MCS = 9                                                                                                                                                                                                     | Short Preamble CCK 2Mbps                 |
| MCS = 10                                                                                                                                                                                                    | Short Preamble 5.5Mbps                   |
| MCS = 11                                                                                                                                                                                                    | Short Preamble 11Mbps                    |
| Notes:                                                                                                                                                                                                      |                                          |
| <ol style="list-style-type: none"> <li>Other MCS codes are reserved in legacy CCK mode.</li> <li>BW, SGI and STBC are reserved in legacy CCK mode.</li> </ol>                                               |                                          |
| 6.2.2 MODE = 1, Legacy OFDM                                                                                                                                                                                 |                                          |
| MCS = 0                                                                                                                                                                                                     | 6Mbps                                    |
| MCS = 1                                                                                                                                                                                                     | 9Mbps                                    |
| MCS = 2                                                                                                                                                                                                     | 12Mbps                                   |
| MCS = 3                                                                                                                                                                                                     | 18Mbps                                   |
| MCS = 4                                                                                                                                                                                                     | 24Mbps                                   |
| MCS = 5                                                                                                                                                                                                     | 36Mbps                                   |
| MCS = 6                                                                                                                                                                                                     | 48Mbps                                   |
| MCS = 7                                                                                                                                                                                                     | 54Mbps                                   |
| Notes:                                                                                                                                                                                                      |                                          |
| <ol style="list-style-type: none"> <li>Other MCS code in legacy CCK mode are reserved.</li> <li>When BW = 1, duplicate legacy OFDM is sent.</li> <li>SGI, STBC are reserved in legacy OFDM mode.</li> </ol> |                                          |
| 6.2.3 MODE = 2, HT Mixed Mode                                                                                                                                                                               |                                          |
| 6.2.4 MODE = 3, HT Greenfield                                                                                                                                                                               |                                          |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| MCS = 0 (1S)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | (BW=0, SGI=0) 6.5Mbps            |
| MCS = 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 13Mbps             |
| MCS = 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 19.5Mbps           |
| MCS = 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 26Mbps             |
| MCS = 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 39Mbps             |
| MCS = 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 52Mbps             |
| MCS = 6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 58.5Mbps           |
| MCS = 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 65Mbps             |
| MCS = 8 (2S)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | (BW=0, SGI=0) 13Mbps             |
| MCS = 9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (BW=0, SGI=0) 26Mbps             |
| MCS = 10                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 39Mbps             |
| MCS = 11                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 52Mbps             |
| MCS = 12                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 78Mbps             |
| MCS = 13                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 104Mbps            |
| MCS = 14                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 117Mbps            |
| MCS = 15                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=0, SGI=0) 130Mbps            |
| MCS = 32                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (BW=1, SGI=0) HT duplicate 6Mbps |
| Notes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                  |
| <ol style="list-style-type: none"> <li>1. When BW=1, PHY_RATE = PHY_RATE * 2</li> <li>2. When SGI=1, PHY_RATE = PHY_RATE * 10/9</li> <li>3. The effects of BW and SGI are accumulative.</li> <li>4. When MCS=0~7(1S, One Tx Stream), STBC option is supported. SGI option is supported. BW option is supported.</li> <li>5. When MCS=8~15(2S, Two Tx Stream), STBC option is NOT supported. SGI option is supported. BW option is supported.</li> <li>6. When MCS=32, only SGI option is supported. BW and STBC option are not supported. (BW =1, STBC=0)</li> <li>7. Other MCS code in HT mode are reserved.</li> <li>8. When STBC is supported. Only STBC = 1 is allowed. STBC will extend the transmission range but will not increase transmission rate.</li> </ol> |                                  |

## 6.3 Examples

### 6.3.1 Check EVM & Power

```

iwpriv ra0 set ATE=AESTART
iwpriv ra0 set ATEDA=00:11:22:33:44:55
iwpriv ra0 set ATESA=00:aa:bb:cc:dd:ee
iwpriv ra0 set ATEBSSID=00:11:22:33:44:55
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ATETXMODE=1 ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7 ; set MCS type.
iwpriv ra0 set ATETXBW=0 ; set Bandwidth
iwpriv ra0 set ATETXGI=0 ; set Long GI.
iwpriv ra0 set ATETXLEN=1024 ; set packet length.
iwpriv ra0 set ATETXPOW0=18
iwpriv ra0 set ATETXPOW1=18
iwpriv ra0 set ATETXCNT=100000
iwpriv ra0 set ATE=TXFRAME
...
iwpriv ra0 set ATETXPOW0=19
...
iwpriv ra0 set ATETXPOW0=20
...
iwpriv ra0 set ATE=AESTART

```

### 6.3.2 Check Carrier

```

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ATETXMODE=1 ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7 ; set MCS type.
iwpriv ra0 set ATETXBW=0 ; set Bandwidth
iwpriv ra0 set ATETXCNT=200 ; Tx frame count(decimal)
iwpriv ra0 set ATE=TXFRAME ; Start Tx Frame(inform BBP to change, modulation mode)
iwpriv ra0 set ATE=TXCARR ; Start Tx carrier, Measure carrier with instrument
iwpriv ra0 set ATETXPOW0=05
iwpriv ra0 set ATETXPOW1=05
iwpriv ra0 set ATETXFREQOFFSET=19
iwpriv ra0 set ATE=ATESTART

```

### 6.3.3 Check spectrum mask

```

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ATETXMODE=1 ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7 ; set MCS type.
iwpriv ra0 set ATETXBW=0 ; set Bandwidth
iwpriv ra0 set ATETXCNT=200 ; Tx frame count(decimal)
iwpriv ra0 set ATE=TXFRAME ; Start Tx Frame(inform BBP to change, modulation mode)
iwpriv ra0 set ATE=TXCONT ; Start continuous TX, Measure spectrum mask with instrument
iwpriv ra0 set ATETXPOW0=5
iwpriv ra0 set ATETXPOW1=5
iwpriv ra0 set ATE=ATESTART

```

### 6.3.4 Frequency offset tuning

```

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ATETXMODE=1 ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7 ; set MCS type.
iwpriv ra0 set ATETXCNT=200 ; Tx frame count(decimal)
iwpriv ra0 set ATETXFREQOFFSET=0 ; Set frequency offset 0(decimal)
iwpriv ra0 set ATE=TXFRAME ; Start Tx Frame
iwpriv ra0 set ATE=TXCARR ; Start Tx carrier, Measure carrier frequency with instrument
iwpriv ra0 set ATETXFREQOFFSET=10 ; Dynamic turning frequency offset, 10(decimal)
iwpriv ra0 set ATETXFREQOFFSET=20 ; Dynamic turning frequency offset, 20(decimal)
iwpriv ra0 set ATE=ATESTART ; Stop, Store the tuning result to EEPROM

```

### 6.3.5 Rx

```

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ResetCounter=0 ; Reset statistic counter
iwpriv ra0 set ATETXMODE=1 ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7 ; set MCS type.
iwpriv ra0 set ATETXBW=0 ; set Bandwidth
iwpriv ra0 set ATETXFREQOFFSET=0 ; set Tx frequency offset
iwpriv ra0 set ATE=RXFRAME ; Start Rx,
iwpriv ra0 set ATERXFER=1 ; show RxCnt and RSSI/per-antenna, Transmit test packets
iwpriv ra0 set ATE=ATESTART ; Stop
iwpriv ra0 stat ; get statistics counter
iwpriv ra0 set ATERXFER=1
iwpriv ra0 set ATERXANT=1

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATERXANT=0
iwpriv ra0 set ATE=RXFRAME

```

### 6.3.6 Show all ate parameters

***iwpriv ra0 set ATESHOW=1***

Mode=4  
TxPower0=0  
TxPower1=0  
TxAntennaSel=0  
RxAntennaSel=0  
BBPCurrentBW=0  
GI=0  
MCS=7  
TxMode=1  
Addr1=00:11:22:aa:bb:cc  
Addr2=00:11:22:aa:bb:cc  
Addr3=00:11:22:aa:bb:cc  
Channel=1  
TxLength=1024  
TxCount=40000  
TxRate=11  
RFFreqOffset=0

### 6.3.7 Online help

***iwpriv ra0 set ATEHELP=1***

ATE=ATESTART, ATESTOP, TXCONT, TXCARR, TXFRAME, RXFRAME  
ATEDA  
ATESA  
ATEBSSID  
ATECHANNEL, range:0~14  
ATETXPOW0, set power level of antenna 1.  
ATETXPOW1, set power level of antenna 2.  
ATETXANT, set TX antenna. 0:all, 1:antenna one, 2:antenna two.  
ATERXANT, set RX antenna.0:all, 1:antenna one, 2:antenna tow, 3:antenna three.  
ATETXFREQOFFSET, set frequency offset, range 0~63  
ATETXBW, set BandWidth, 0:20MHz, 1:40MHz.  
ATETXLEN, set Frame length, range 24~1500  
ATETXCNT, set how many frame going to transmit.  
ATETXRATE, set rate, reference to rate table.  
ATETXMCS, set MCS, reference to rate table.  
ATETXMODE, set Mode 0:CCK, 1:OFDM, 2:HT-Mix, 3:GreenField, reference to rate table.  
ATETXGI, set GI interval, 0:Long, 1:Short  
ATERXFER, 0:disable Rx Frame error rate. 1:enable Rx Frame error rate.  
ATESHOW, display all parameters of ATE.  
ATEHELP, online help.

### 6.3.8 Display Rx Packet Count and RSSI

***iwpriv ra0 set ATE=RXFRAME***

➔ **Start Rx**

***iwpriv ra0 set ATERXANT=0***

➔ **Enable All Three Rx Antennas**

***iwpriv ra0 set ATERXFER=1***

➔ **Enable Rx Frame Error Rate: RxCnt/RxTotal**

MlmePeriodicExec: Rx packet cnt = 2/4  
MlmePeriodicExec: Rx AvgRssi0=-88, AvgRssi1=-80, AvgRssi2=-91

MlmePeriodicExec: Rx packet cnt = 2/6  
MlmePeriodicExec: Rx AvgRssi0=-86, AvgRssi1=-77, AvgRssi2=-89...

\*\*\*

***iwpriv ra0 set ATE=RXFRAME***

➔ **Start Rx**

***iwpriv ra0 set ATERXANT=1***  
***iwpriv ra0 set ATERXFER=1***

➔ ***Enable Three Rx Antenna-1***  
➔ ***Enable Rx Frame Error Rate: RxCnt/RxTotal***

MImePeriodicExec: Rx packet cnt = 0/7  
MImePeriodicExec: Rx AvgRssi=-87

MImePeriodicExec: Rx packet cnt = 7/14  
MImePeriodicExec: Rx AvgRssi=-90

### 6.3.9 Internal ALC calibration (For RT5350 only)

***iwpriv ra0 set ATE=ATESTART***  
***iwpriv ra0 set ATETSSICBA=DAC@CH1***

( Note : DAC@CH1 is referred to the value of channel 1 TX0 power, stored in EEPROM 0x52 b7~b0. When user is finish this procedure, the EEPROM 0x6E will be stuffed the reference value for internal ALC function)

### 6.3.10 Internal ALC function testing in ATE mode (For RT5350 only)

***iwpriv ra0 set ATE=ATESTART***  
***iwpriv ra0 set ATEDA=00:11:22:33:44:55***  
***iwpriv ra0 set ATESA=00:aa:bb:cc:dd:ee***  
***iwpriv ra0 set ATEBSSID=00:11:22:33:44:55***  
***iwpriv ra0 set ATECHANNEL=1***  
***iwpriv ra0 set ATETXMODE=1***  
***iwpriv ra0 set ATETXMCS=7***  
***iwpriv ra0 set ATETXBW=0***  
***iwpriv ra0 set ATETXGI=0***  
***iwpriv ra0 set ATETXLEN=1024***  
***iwpriv ra0 set ATETXPOW=12***  
***iwpriv ra0 set ATETXCNT=10000000***  
***iwpriv ra0 set ATE TXFRAME***  
***iwpriv ra0 set ATEAUTOALC=1 (Note:Enable temperature compensation)***

***Below is recommend testing flow :***

***Make sure the device is calibrated already.***

***Record the channel 1 power DAC value such as #iwpriv ra0 e2p 52 which is 0x0C***

***Run below command for temperature compensation process:***

***#iwpriv ra0 set ATE=ATE ATESTART***

***# iwpriv ra0 set ATETSSICBA=12 ( Note : 12 is the decimal value of 0x0C)***

***Measure the Tx power status in room temperature. (The output power should be +/- 1dBm)***

***If the output power is normal, please change the temperature and check the Tx power status.***

***iwpriv ra0 set ATE=ATESTART***  
***iwpriv ra0 set ATEDA=00:11:22:33:44:55***  
***iwpriv ra0 set ATESA=00:aa:bb:cc:dd:ee***  
***iwpriv ra0 set ATEBSSID=00:11:22:33:44:55***  
***iwpriv ra0 set ATECHANNEL=1***  
***iwpriv ra0 set ATETXMODE=1***  
***iwpriv ra0 set ATETXMCS=7***  
***iwpriv ra0 set ATETXBW=0***  
***iwpriv ra0 set ATETXGI=0***  
***iwpriv ra0 set ATETXLEN=1024***  
***iwpriv ra0 set ATETXPOW=12***  
***iwpriv ra0 set ATETXCNT=10000000***  
***iwpriv ra0 set ATE TXFRAME***

***iwpriv ra0 set ATEAUTOALC=1 (Note:Enable temperature compensation)***

## 6.4 iwpriv ra0 bbp [parameters]=[Value]

Read/Write BBP register by ID number.

### 6.4.1 BBPID

Read BBP register, BBPID only, no "=" symbol.

BBPID:

0 ~ xx ; decimal, 8-bit

### 6.4.2 BBPID=Value

Write BBP register.

BBPID:

0 ~ xx ; decimal, 8-bit

Value:

00 ~ FF ; hexadecimal, 8-bit

## 6.5 iwpriv ra0 mac [parameters]=[val]

Read/Write MAC register by offset.

### 6.5.1 MAC\_OFFSET

Read MAC register, MAC\_OFFSET only, no "=" symbol.

MAC\_OFFSET:

0000 ~ FFFF ; hexadecimal, 16-bit

### 6.5.2 MAC\_OFFSET=Value

Write MAC register.

MAC\_OFFSET:

0000 ~ FFFF ; hexadecimal, 16-bit

Value:

0000 ~ FFFF ; hexadecimal, 32-bit

## 6.6 iwpriv ra0 e2p [parameters]=[val]

Read/Write EEPROM content by address.

### 6.6.1 EEP\_ADDR

Read EEPROM content, EEP\_ADDR only, no "=" symbol.

EEP\_ADDR:

00 ~ FF ; hexadecimal, 16-bit alignment (0, 2, 4, 6, 8, A, C, ...)

### 6.6.2 EEP\_ADDR=Value

Write EEPROM content.

EEP\_ADDR:

00 ~ FF ; hexadecimal, 16-bit alignment (0, 2, 4, 6, 8, A, C, ...)

Value:

0000 ~FFFF ; hexadecimal, 16-bit

## 6.7 Example

### 6.7.1 Hardware access

|                           |                                      |
|---------------------------|--------------------------------------|
| iwpriv ra0 bbp 0          | # read BBP register 0                |
| iwpriv ra0 bbp 0=12       | # write BBP register 0 as 0x12       |
| iwpriv ra0 mac 0          | # read MAC register 0                |
| iwpriv ra0 mac 0=1234abcd | # write MAC register 0 as 0x1234abcd |
| iwpriv ra0 e2p 0          | # read E2PROM 0                      |
| iwpriv ra0 e2p c=12ab     | # write E2PROM 0xc as 0x12ab         |

### 6.7.2 Statistic counter operation

|                               |                           |
|-------------------------------|---------------------------|
| iwpriv ra0 stat               | # read statistic counter  |
| iwpriv ra0 set ResetCounter=0 | # reset statistic counter |

### 6.7.3 Suggestion:

1. To turn on ATE functionality, you have to add compile flag "RALINK\_ATE" to Makefile
2. Before doing ATE testing, please stop AP function
3. If you want to test another ATE action, prefer to stop AP & ATE function
4. All ATE function settings will lose efficacy after reboot.
5. Before hardware register access, please reference hardware spec.



Note.

In ATE mode, the channel must set via "ATECHANNEL"

MEDIA TEK CONFIDENTIAL  
FOR panwl@laketune.com.cn USE ONLY

## 6.8 ated

- ated - user space ATE agent program for RT3572 linux driver, Ralink Tech. Corp.
- RT3572 ATE daemon - ated, comes with RT3572 Linux driver.

The section describes the use of the Linux driver, Windows QA GUI and RT3572 ATE daemon.

### 6.8.1 Introduction

The ated is an optional user space component for the RT3572 linux driver. AP immediately starts working in ATE mode when "ated" starts (i.e. ATESTART). It behaves as a proxy between Windows QA GUI and RT3572 linux driver when ATE process proceeds.

**ATED AUTOMATICALLY STOPS WHEN THE WINDOWS QA GUI IS CLOSED.**

Ated can be stopped manually (key in '\$killall ated'). The RT3572 linux driver will stop working in ATE mode when ated is stopped or the QA GUI is closed.

### 6.8.2 Environment setup

1. Connect the platform you want to test directly with a Windows host by ether network line.
2. In the Windows host, run WinPcap\_4\_0.exe for the QA GUI.

### 6.8.3 How to use ated for ATE purpose

1. First you should set **both "HAS\_ATE=y" and "HAS\_28XX\_QA=y"** in the file `~/Module/os/linux/config.mk` and compile the driver.
2. Modify the Makefile according to our target "PLATFORM".
3. Change the path of "CROSS\_COMPILE" if needed.
4. Remove "-I\$(INCLUDE)" about in line 39 if your target "PLATFORM" is not "PC".
5. Then type 'make' command to compile the source code of the daemon.
6. After the driver interface "ra0" has started up, attach both of "ra0" and the ethernet interface to the bridge interface "br0".
7. Manually start ated, type '\$ated -bbrX -iraX'. (For further use of options, type \$ated -h)
8. In the Windows host, run RT2870QA\_ATE.exe.
9. Select the wired network adapter.
10. Choose 2870\_ATE, then press OK.

Note:

The names of WLAN interface(default is "ra0") and Bridge interface(default is "br0") must be specified manually (for example: '\$ated -b br1 -ira2') if your WLAN interface or Bridge interface is not "ra0" or "br0" respectively !

## 7 IOCTL

### 7.1 Parameters for iwconfig

| Access | Description                 | ID            | Parameters                                                                                                                                                                                                                                                                                                                        |
|--------|-----------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get    | BSSID, MAC Address          | SIOCGIFHWADDR | wrq->u.name, (length = 6)                                                                                                                                                                                                                                                                                                         |
|        | WLAN Name                   | SIOCGIWNAME   | wrq->u.name = "RT3572Wireless", length = strlen(wrq->u.name)                                                                                                                                                                                                                                                                      |
|        | SSID                        | SIOCGIWESSID  | <pre> erq = &amp;wrq-&gt;u.essid;  if(OPSTATUS_TEST_FLAG(pAd,fOP_STATUS_MEDIA_STATE_CONNECTED)) {     erq-&gt;flags=1;     erq-&gt;length = pAd-&gt; CommonCfg.SsidLen;     Status = copy_to_user(erq-&gt;pointer,         pAd-&gt; CommonCfg.Ssid, erq-&gt;length); } else {     erq-&gt;flags=0;     erq-&gt;length=0; } </pre> |
|        | Channel / Frequency<br>(Hz) | SIOCGIWFREQ   | <pre> wrq-&gt;u.freq.m = pAd-&gt; CommonCfg.Channel; wrq-&gt;u.freq.e = 0; wrq-&gt;u.freq.i = 0; </pre>                                                                                                                                                                                                                           |
|        | Node name/nickname          | SIOCGIWNICKN  | <pre> erq = &amp;wrq-&gt;u.data; erq-&gt;length = strlen(pAd-&gt;nickn); Status = copy_to_user(erq-&gt;pointer, pAd-&gt;nickn, erq-&gt;length); </pre>                                                                                                                                                                            |
|        | Bit Rate<br>(bps)           | SIOCGIWRATE   | <pre> wrq-&gt;u.bitrate.value = RateIdTo500Kbps[pAd-&gt; CommonCfg.TxRate] * 500000; wrq-&gt;u.bitrate.disabled = 0; </pre>                                                                                                                                                                                                       |
|        | RTS/CTS threshold           | SIOCGIWRTS    | <pre> wrq-&gt;u.rts.value = (INT) pAd-&gt; CommonCfg.RtsThreshold; wrq-&gt;u.rts.disabled = (wrq-&gt;u.rts.value == MAX_RTS_THRESHOLD); wrq-&gt;u.rts.fixed = 1; </pre>                                                                                                                                                           |
|        | Fragmentation threshold     | SIOCGIWFRAG   | wrq->u.frag.value = (INT) pAd-> CommonCfg.FragmentThreshold;                                                                                                                                                                                                                                                                      |

|                          |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (bytes)                  |               | <pre> wrq-&gt;u.frag.disabled = (wrq-&gt;u.frag.value &gt;= MAX_FRAG_THRESHOLD); wrq-&gt;u.frag.fixed = 1; </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Encoding<br>token & mode | SIOCGIWENCODE | <pre> index = (wrq-&gt;u.encoding.flags &amp; IW_ENCODE_INDEX) - 1; if ((index &lt; 0)    (index &gt;= NR_WEP_KEYS))     index = pAd-&gt;CommonCfg.DefaultKeyId; // Default key for tx     (shared key) if (pAd-&gt;CommonCfg.AuthMode == Ndis802_11AuthModeOpen)     wrq-&gt;u.encoding.flags = IW_ENCODE_OPEN; else if (pAd-&gt;CommonCfg.AuthMode == Ndis802_11AuthModeShared)     wrq-&gt;u.encoding.flags = IW_ENCODE_RESTRICTED; if (pAd-&gt;CommonCfg.WepStatus == Ndis802_11WEPDisabled)     wrq-&gt;u.encoding.flags  = IW_ENCODE_DISABLED; else {     if(wrq-&gt;u.encoding.pointer)     {         wrq-&gt;u.encoding.length = pAd-&gt;SharedKey[index].KeyLen;         Status = copy_to_user(wrq-&gt;u.encoding.pointer,                                pAd-&gt;SharedKey[index].Key,                                pAd-&gt;SharedKey[index].KeyLen);         wrq-&gt;u.encoding.flags  = (index + 1);     } } </pre> |
| AP's MAC address         | SIOCGIWAP     | <pre> wrq-&gt;u.ap_addr.sa_family = ARPHRD_ETHER; memcpy(wrq-&gt;u.ap_addr.sa_data, &amp;pAd-&gt;CommonCfg.Bssid, ETH_ALEN); </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Operation<br>Mode        | SIOCGIWMODE   | <pre> if (ADHOC_ON(pAd)) {     BssType = Ndis802_11IBSS;     wrq-&gt;u.mode = IW_MODE_ADHOC; } else if (INFRA_ON(pAd)) {     BssType = Ndis802_11Infrastructure; } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|        |             |              | <pre> wrq-&gt;u.mode = IW_MODE_INFRA; } else {     BssType = Ndis802_11AutoUnknown;     wrq-&gt;u.mode = IW_MODE_AUTO; } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|-------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Access | Description | ID           | Parameters                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Set    | SSID        | SIOCSIWESSID | <pre> erq = &amp;wrq-&gt;u.essid; memset(&amp;Ssid, 0x00, sizeof(NDIS_802_11_SSID)); if (erq-&gt;flags) {     if (erq-&gt;length &gt; IW_ESSID_MAX_SIZE)     {         Status = -E2BIG;         break;     }     Status = copy_from_user(Ssid.Ssid, erq-&gt;pointer, (erq-&gt;length - 1));     Ssid.SsidLength = erq-&gt;length - 1; //minus null character. } else {     Ssid.SsidLength = 0; // ANY ssid     memcpy(pSsid-&gt;Ssid, "", 0);     pAd-&gt;CommonCfg.BssType = BSS_INFRA;     pAd-&gt;CommonCfg.AuthMode = Ndis802_11AuthModeOpen;     pAd-&gt;CommonCfg.WepStatus = Ndis802_11EncryptionDisabled; } pSsid = &amp;Ssid; if (pAd-&gt;Mlme.CntlMachine.CurrState != CNTL_IDLE) { </pre> |

|                             |              |  |                                                                                                                                                                                                                                                                                                                               |
|-----------------------------|--------------|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |              |  | <pre> MlmeRestartStateMachine(pAd); }  pAd-&gt;MlmeAux.CurrReqlsFromNdis = FALSE;  MlmeEnqueue(pAd,              MLME_CNTL_STATE_MACHINE,              OID_802_11_SSID,              sizeof(NDIS_802_11_SSID),              (VOID *)pSsid);  Status = NDIS_STATUS_SUCCESS;  StateMachineTouched = TRUE; </pre>                |
| Channel / Frequency<br>(Hz) | SIOCSIWFREQ  |  | <pre> frq = &amp;wrq-&gt;u.freq;  if((frq-&gt;e == 0) &amp;&amp; (frq-&gt;m &lt;= 1000))     chan = frq-&gt;m; // Setting by channel number else     MAP_KHZ_TO_CHANNEL_ID( (frq-&gt;m /100) , chan);  pAd-&gt;CommonCfg.Channel = chan; </pre>                                                                               |
| node name/nickname          | SIOCSIWNICKN |  | <pre> erq = &amp;wrq-&gt;u.data;  if (erq-&gt;flags) {     if (erq-&gt;length &lt;= IW_ESSID_MAX_SIZE)         Status = copy_from_user(pAd-&gt;nickn, erq-&gt;pointer, erq-&gt;length);     else         Status = -E2BIG; } </pre>                                                                                            |
| Bit Rate<br>(bps)           | SIOCSIWRATE  |  | <pre> RTMPSetDesiredRates(pAd, wrq-&gt;u.bitrate.value); </pre>                                                                                                                                                                                                                                                               |
| RTS/CTS threshold           | SIOCSIWRTS   |  | <pre> RtsThresh = wrq-&gt;u.rts.value;  if (wrq-&gt;u.rts.disabled)     RtsThresh = MAX_RTS_THRESHOLD;  if((RtsThresh &gt; 0) &amp;&amp; (RtsThresh &lt;= MAX_RTS_THRESHOLD))     pAd-&gt;CommonCfg.RtsThreshold = (USHORT)RtsThresh; else if (RtsThresh == 0)     pAd-&gt;CommonCfg.RtsThreshold = MAX_RTS_THRESHOLD; </pre> |

|                                    |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Fragmentation threshold<br>(bytes) | SIOCSIWFRAG  | <pre> FragThresh = wrq-&gt;u.frag.value;  if (wrq-&gt;u.rts.disabled)      FragThresh = MAX_FRAG_THRESHOLD;  if ( (FragThresh &gt;= MIN_FRAG_THRESHOLD) &amp;&amp;     (FragThresh &lt;= MAX_FRAG_THRESHOLD))      pAd-&gt;CommonCfg.FragmentThreshold = (USHORT)FragThresh;  else if (FragThresh == 0)      pAd-&gt;CommonCfg.FragmentThreshold = MAX_FRAG_THRESHOLD;  if (pAd-&gt;CommonCfg.FragmentThreshold == MAX_FRAG_THRESHOLD)      pAd-&gt;CommonCfg.bFragmentZeroDisable = TRUE;  else      pAd-&gt;CommonCfg.bFragmentZeroDisable = FALSE; </pre>                                                                                                                                                                                                                                                                                                                                    |
| Encoding<br>token & mode           | SIOCSIWENCOD | <pre> index = (wrq-&gt;u.encoding.flags &amp; IW_ENCODE_INDEX) - 1; if((index &lt; 0)    (index &gt;= NR_WEP_KEYS))      index = pAd-&gt;CommonCfg.DefaultKeyId; // Default key for tx (shared key)  if(wrq-&gt;u.encoding.pointer) {     len = wrq-&gt;u.encoding.length;      if(len &gt; WEP_LARGE_KEY_LEN)          len = WEP_LARGE_KEY_LEN;      memset(pAd-&gt;SharedKey[index].Key, 0x00, MAX_LEN_OF_KEY);      Status = copy_from_user(pAd-&gt;SharedKey[index].Key,          wrq-&gt;u.encoding.pointer, len);      pAd-&gt;SharedKey[index].KeyLen = len &lt;= WEP_SMALL_KEY_LEN ?         WEP_SMALL_KEY_LEN :         WEP_LARGE_KEY_LEN;  }  pAd-&gt;CommonCfg.DefaultKeyId = (UCHAR) index;  if (wrq-&gt;u.encoding.flags &amp; IW_ENCODE_DISABLED)      pAd-&gt;CommonCfg.WepStatus = Ndis802_11WEPDisabled;  else      pAd-&gt;CommonCfg.WepStatus = Ndis802_11WEPEnabled; </pre> |

|                   |             |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|-------------|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |             |  | <pre> if (wrq-&gt;u.encoding.flags &amp; IW_ENCODE_RESTRICTED)      pAd-&gt;CommonCfg.AuthMode = Ndis802_11AuthModeShared;  else      pAd-&gt;CommonCfg.AuthMode = Ndis802_11AuthModeOpen;  if(pAd-&gt;CommonCfg.WepStatus == Ndis802_11WEPDisabled)      pAd-&gt;CommonCfg.AuthMode = Ndis802_11AuthModeOpen; </pre>                                                                                                                                             |
| AP's MAC address  | SIOCSIWAP   |  | <pre> Status = copy_from_user(&amp;Bssid, &amp;wrq-&gt;u.ap_addr.sa_data,     sizeof(NDIS_802_11_MAC_ADDRESS));  if (pAd-&gt;Mlme.CntlMachine.CurrState != CNTL_IDLE) {     MlmeRestartStateMachine(pAd); }  pAd-&gt;MlmeAux.CurrReqIsFromNdis = FALSE;  MlmeEnqueue(pAd,      MLME_CNTL_STATE_MACHINE,      OID_802_11_BSSID,      sizeof(NDIS_802_11_MAC_ADDRESS),      (VOID *)&amp;Bssid);  Status = NDIS_STATUS_SUCCESS;  StateMachineTouched = TRUE; </pre> |
| Operation<br>Mode | SIOCSIWMODE |  | <pre> if(wrq-&gt;u.mode == IW_MODE_ADHOC) {     if (pAd-&gt;CommonCfg.BssType != BSS_ADHOC)     {         pAd-&gt;bConfigChanged = TRUE;     }      pAd-&gt;CommonCfg.BssType = BSS_ADHOC; }  else if (wrq-&gt;u.mode == IW_MODE_INFRA) {     if (pAd-&gt;CommonCfg.BssType != BSS_INFRA)     { </pre>                                                                                                                                                            |



|  |  |  |                                                                                                                                                                     |
|--|--|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  |  |  | <pre>pAd-&gt;bConfigChanged = TRUE;  }  pAd-&gt;CommonCfg.BssType = BSS_INFRA;  }  else  {      Status = -EINVAL;  }  pAd-&gt;CommonCfg.WpaState = SS_NOTUSE;</pre> |
|--|--|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 7.2 Parameters for iwpriv

Please refer section 3 to have iwpriv parameters and values.

### Parameters:

```
int socket_id;

char name[25]; // interface name

char data[255]; // command string

struct iwreq wrq;
```

### Default setting:

```
wrq.ifr_name = name = "ra0"; // interface name

wrq.u.data.pointer = data; // data buffer of command string

wrq.u.data.length = strlen(data); // length of command string

wrq.u.data.flags = 0;
```

### Data Structure:

Please refer to `./include/oid.h` for update and detail definition.

### 7.2.1 Set Data, Parameters is Same as iwpriv

| Command and IOCTL Function |                                           |                                                                                                                                                                                                                                    |
|----------------------------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Set Data                   |                                           |                                                                                                                                                                                                                                    |
| Function Type              | Command                                   | IOCTL                                                                                                                                                                                                                              |
| <b>RTPRIV_IOCTL_SET</b>    | <code>iwpriv ra0 set SSID=RT3572AP</code> | <pre>printf(name, "ra0"); strcpy(data, "SSID=RT3572AP"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, <b>RTPRIV_IOCTL_SET</b>, &amp;wrq);</pre> |

## 7.2.2 Get Data, Parameters is Same as iwpriv

| Command and IOCTL Function      |                            |                                                                                                                                                                                                                                          |
|---------------------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get Data                        |                            |                                                                                                                                                                                                                                          |
| Function Type                   | Command                    | IOCTL                                                                                                                                                                                                                                    |
| <b>RTPRIV_IOCTL_STATISTICS</b>  | lwpriv ra0 stat            | <pre> sprintf(name, "ra0"); strcpy(data, "stat"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, RTPRIV_IOCTL_STATISTICS, &amp;wrq); </pre>             |
| <b>RTPRIV_IOCTL_GSITESURVEY</b> | lwpriv ra0 get_site_survey | <pre> sprintf(name, "ra0"); strcpy(data, "get_site_survey"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, RTPRIV_IOCTL_GSITESURVEY, &amp;wrq); </pre> |

### 7.2.3 Set Raw Data with Flags

| IOCTL Function                                   |                                                                                                                                                                                                                                                                                 |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Set Raw Data by I/O Control Interface with Flags |                                                                                                                                                                                                                                                                                 |
| Function Type                                    | IOCTL                                                                                                                                                                                                                                                                           |
| <b>RT_OID_802_11_COUNTRY_REGION</b>              | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UCHAR)); wrq.u.data.length = sizeof(UCHAR); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_COUNTRY_REGION</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>          |
| <b>OID_802_11_BSSID_LIST_SCAN</b>                | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_BSSID_LIST_SCAN</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                        |
| <b>OID_802_11_SSID</b>                           | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_SSID)); wrq.u.data.length = sizeof(NDIS_802_11_SSID); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_SSID</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_BSSID</b>                          | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_MAC_ADDRESS)); wrq.u.data.length = sizeof(NDIS_802_11_MAC_ADDRESS); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_BSSID</b>; </pre>                                    |

|                                 |                                                                                                                                                                                                                                                                                                  |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 | ioctl(socket_id, <b>RT_PRIV_IOCTL</b> , &wrq);                                                                                                                                                                                                                                                   |
| <b>RT_OID_802_11_RADIO</b>      | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN)); wrq.u.data.length = sizeof(BOOLEAN); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_RADIO</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                |
| <b>RT_OID_802_11_PHY_MODE</b>   | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_PHY_MODE)); wrq.u.data.length = sizeof(RT_802_11_PHY_MODE); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_PHY_MODE</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>       |
| <b>RT_OID_802_11_STA_CONFIG</b> | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_STA_CONFIG)); wrq.u.data.length = sizeof(RT_802_11_STA_CONFIG); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_STA_CONFIG</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_DESIRED_RATES</b> | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_RATES)); wrq.u.data.length = sizeof(NDIS_802_11_RATES); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_DESIRED_RATES</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>       |
| <b>RT_OID_802_11_PREAMBLE</b>   | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_PREAMBLE)); </pre>                                                                                                                                                                                      |

|                                       |                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                       | <pre> wrq.u.data.length = sizeof(RT_802_11_PREAMBLE);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_PREAMBLE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                                          |
| <b>OID_802_11_WEP_STATUS</b>          | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_WEP_STATUS));  wrq.u.data.length = sizeof(NDIS_802_11_WEP_STATUS);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_WEP_STATUS</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                  |
| <b>OID_802_11_AUTHENTICATION_MODE</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_AUTHENTICATION_MODE));  wrq.u.data.length = sizeof(NDIS_802_11_AUTHENTICATION_MODE);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_AUTHENTICATION_MODE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>       |
| <b>OID_802_11_INFRASTRUCTURE_MODE</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_NETWORK_INFRASTRUCTURE));  wrq.u.data.length = sizeof(NDIS_802_11_NETWORK_INFRASTRUCTURE);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_INFRASTRUCTURE_MODE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_REMOVE_WEP</b>          | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_KEY_INDEX));  wrq.u.data.length = sizeof(NDIS_802_11_KEY_INDEX);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_REMOVE_WEP</b>; </pre>                                                                                       |

|                                           |                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                           | ioctl(socket_id, <b>RT_PRIV_IOCTL</b> , &wrq);                                                                                                                                                                                                                                                                                          |
| <b>RT_OID_802_11_RESET_COUNTERS</b>       | <pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_RESET_COUNTERS</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                               |
| <b>OID_802_11_RTS_THRESHOLD</b>           | <pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_RTS_THRESHOLD)); wrq.u.data.length = sizeof(NDIS_802_11_RTS_THRESHOLD); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_RTS_THRESHOLD</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                               |
| <b>OID_802_11_FRAGMENTATION_THRESHOLD</b> | <pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_FRAGMENTATION_THRESHOLD)); wrq.u.data.length = sizeof(NDIS_802_11_FRAGMENTATION_THRESHOLD); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_FRAGMENTATION_THRESHOLD</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_POWER_MODE</b>              | <pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_POWER_MODE)); wrq.u.data.length = sizeof(NDIS_802_11_POWER_MODE); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_POWER_MODE</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                        |
| <b>OID_802_11_TX_POWER_LEVEL</b>          | <pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_TX_POWER_LEVEL)); </pre>                                                                                                                                                                                                                      |

|                                       |                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                       | <pre> wrq.u.data.length = sizeof(NDIS_802_11_TX_POWER_LEVEL);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_TX_POWER_LEVEL</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                             |
| <b>RT_OID_802_11_TX_POWER_LEVEL_1</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_TX_POWER_LEVEL_1</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                         |
| <b>OID_802_11_NETWORK_TYPE_IN_USE</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_NETWORK_TYPE));  wrq.u.data.length = / sizeof(NDIS_802_11_NETWORK_TYPE);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_NETWORK_TYPE_IN_USE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_RX_ANTENNA_SELECTED</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_ANTENNA));  wrq.u.data.length = sizeof(NDIS_802_11_ANTENNA);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_RX_ANTENNA_SELECTED</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>             |
| <b>OID_802_11_TX_ANTENNA_SELECTED</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_ANTENNA));  wrq.u.data.length = sizeof(NDIS_802_11_ANTENNA);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_TX_ANTENNA_SELECTED</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>             |



|                                             |                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RT_OID_802_11_ADD_WPA</b>                | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, 32);  wrq.u.data.length = 32;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_ADD_WPA</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                         |
| <b>OID_802_11_REMOVE_KEY</b>                | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_REMOVE_KEY));  wrq.u.data.length = sizeof(NDIS_802_11_REMOVE_KEY);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_REMOVE_KEY</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_ADD_KEY</b>                   | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, keylength); //5,10,13,26  wrq.u.data.length = keylength L;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_ADD_KEY</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                               |
| <b>OID_802_11_SET_IEEE8021X</b>             | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(BOOLEAN));  wrq.u.data.length = sizeof(BOOLEAN);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_SET_IEEE8021X</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                            |
| <b>OID_802_11_SET_IEEE8021X_REQUIRE_KEY</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(BOOLEAN));  wrq.u.data.length = sizeof(BOOLEAN); </pre>                                                                                                                                                                |

|                                 |                                                                                                                                                                                                                                                                                                                 |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 | <pre> wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_SET_IEEE8021X_REQUIRE_KEY</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                                                     |
| <b>OID_802_11_ADD_WEP</b>       | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, keylength); //5,10,13,26  wrq.u.data.length = keylength;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_RADIO</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                         |
| <b>OID_802_11_CONFIGURATION</b> | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_CONFIGURATION));  wrq.u.data.length = sizeof(NDIS_802_11_CONFIGURATION);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_CONFIGURATION</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_SET_COUNTERMEASURES</b>  | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  wrq.u.data.length = 0;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_SET_COUNTERMEASURES</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                       |
| <b>OID_802_11_DISASSOCIATE</b>  | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  wrq.u.data.length = 0;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_DISASSOCIATE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                       |
| <b>OID_802_11_PMKID</b>         | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  wrq.u.data.length = keylength; //follow your setting </pre>                                                                                                                                                                                            |

|                                                              |                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                              | <pre> wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_PMKID</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                                          |
| <b>RT_OID_WPA_SUPPLICANT_SUPPORT</b>                         | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(BOOLEAN));  wrq.u.data.length = sizeof(BOOLEAN);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_WPA_SUPPLICANT_SUPPORT</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>RT_OID_WPA_SUPPLICANT_SUPPORT</b>                         | <pre> printf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_WPA_SUPPLICANT_SUPPORT</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>     |
| <b>RT_SET_DEL_MAC_ENTRY</b>                                  | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0xdd, 6);  strcpy(wrq.ifr_name, name);  wrq.u.data.length = 6;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_SET_DEL_MAC_ENTRY</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>     |
|                                                              |                                                                                                                                                                                                                                                                                  |
| <b>RT_OID_802_11_SET_HT_PHYMODE  <br/>OID_GET_SET_TOGGLE</b> | <pre> typedef struct {      RT_802_11_PHY_MODE PhyMode;      UCHAR TransmitNo;      UCHAR HtMode; //HTMODE_GF or HTMODE_MM      UCHAR ExtOffset; //extension channel above or below      UCHAR MCS; </pre>                                                                       |

|  |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <pre>         UCHAR BW;          UCHAR STBC;          UCHAR SHORTGI;          UCHAR rsv;      } OID_SET_HT_PHYMODE ;      RT_802_11_PHY_MODE tmp_ht_mode;      sprintf(wrq.ifr_name, "ra0");      wrq.u.data.pointer = (caddr_t) &amp;tmp_ht_mode;      wrq.u.data.length = sizeof(RT_802_11_PHY_MODE);      wrq.u.data.flags = RT_OID_802_11_SET_HT_PHYMODE       OID_GET_SET_TOGGLE;      ioctl(socket_id, RT_PRIV_IOCTL, &amp;wrq); </pre> |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 7.2.4 Get Raw Data with Flags

| IOCTL Function                                   |                                                                                                                                                                                                                                                                                          |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get Raw Data by I/O Control Interface with Flags |                                                                                                                                                                                                                                                                                          |
| Function Type                                    | IOCTL                                                                                                                                                                                                                                                                                    |
| RT_OID_DEVICE_NAME                               | <pre>         sprintf(name, "ra0");          strcpy(wrq.ifr_name, name);          memset(data, 0, 255);          wrq.u.data.length = 255;          wrq.u.data.pointer = data;          wrq.u.data.flags = RT_OID_DEVICE_NAME;          ioctl(socket_id, RT_PRIV_IOCTL, &amp;wrq); </pre> |
| RT_OID_VERSION_INFO                              | <pre>         sprintf(name, "ra0");          strcpy(wrq.ifr_name, name);          memset(data, 0, sizeof(RT_VERSION_INFO));          wrq.u.data.length = sizeof(RT_VERSION_INFO);          wrq.u.data.pointer = data;          wrq.u.data.flags = RT_OID_VERSION_INFO; </pre>            |

|                                     |                                                                                                                                                                                                                                                                                                |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     | ioctl(socket_id, <b>RT_PRIV_IOCTL</b> , &wrq);                                                                                                                                                                                                                                                 |
| <b>OID_802_11_BSSID_LIST</b>        | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, BssLen); wrq.u.data.length = BssLen; wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_BSSID_LIST</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                              |
| <b>OID_802_3_CURRENT_ADDRESS</b>    | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(CurrentAddress)); wrq.u.data.length = sizeof(CurrentAddress); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_3_CURRENT_ADDRESS</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>          |
| <b>OID_GEN_MEDIA_CONNECT_STATUS</b> | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_MEDIA_STATE)); wrq.u.data.length = sizeof(NDIS_MEDIA_STATE); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_GEN_MEDIA_CONNECT_STATUS</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>   |
| <b>OID_802_11_BSSID</b>             | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_MAC_ADDRESS)); wrq.u.data.length = sizeof(NDIS_802_11_MAC_ADDRESS); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_BSSID</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_SSID</b>              | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_SSID)); </pre>                                                                                                                                                                                      |

|                                        |                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                        | <pre> wrq.u.data.length = sizeof(NDIS_802_11_SSID);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_SSID</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                            |
| <b>RT_OID_802_11_QUERY_LINK_STATUS</b> | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(RT_802_11_LINK_STATUS));  wrq.u.data.length = sizeof(RT_802_11_LINK_STATUS);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_QUERY_LINK_STATUS</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>  |
| <b>OID_802_11_CONFIGURATION</b>        | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_CONFIGURATION));  wrq.u.data.length = sizeof(NDIS_802_11_CONFIGURATION);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_CONFIGURATION</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_802_11_RSSI_TRIGGER</b>         | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ulInfo));  wrq.u.data.length = sizeof(ulInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_RSSI_TRIGGER</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                        |
| <b>RT_OID_802_11_RSSI</b>              | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ulInfo));  wrq.u.data.length = sizeof(ulInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_RSSI</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                             |

|                              |                                                                                                                                                                                                                                                                                                         |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RT_OID_802_11_RSSI_1</b>  | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ullInfo));  wrq.u.data.length = sizeof(ullInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_RSSI_1</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                |
| <b>RT_OID_802_11_RSSI_2</b>  | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ullInfo));  wrq.u.data.length = sizeof(ullInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_RSSI_2</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                |
| <b>OID_802_11_STATISTICS</b> | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(NDIS_802_11_STATISTICS));  wrq.u.data.length = sizeof(NDIS_802_11_STATISTICS);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_STATISTICS</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>OID_GEN_RCV_OK</b>        | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ullInfo));  wrq.u.data.length = sizeof(ullInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_GEN_RCV_OK</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                      |
| <b>OID_GEN_RCV_NO_BUFFER</b> | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ullInfo));  wrq.u.data.length = sizeof(ullInfo); </pre>                                                                                                                                                                |

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 | <pre> wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_GEN_RCV_NO_BUFFER</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>RT_OID_802_11_PHY_MODE</b>   | <pre> typedef enum _RT_802_11_PHY_MODE {      PHY_11BG_MIXED = 0,      PHY_11B,      PHY_11A,      PHY_11ABG_MIXED,      PHY_11G,      PHY_11ABGN_MIXED,    // both band  5      PHY_11N,              //  6      PHY_11GN_MIXED,      // 2.4G band  7      PHY_11AN_MIXED,      // 5G band   8      PHY_11BGN_MIXED,     // if check 802.11b.  9      PHY_11AGN_MIXED,     // if check 802.11b. 10  } RT_802_11_PHY_MODE  sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ulInfo));  wrq.u.data.length = sizeof(ulInfo);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_PHY_MODE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>RT_OID_802_11_STA_CONFIG</b> | <pre> sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(RT_802_11_STA_CONFIG));  wrq.u.data.length = sizeof(RT_802_11_STA_CONFIG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_STA_CONFIG</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>OID_802_11_RTS_THRESHOLD</b> | <pre> sprintf(name, "ra0"); </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |



|                                           |                                                                                                                                                                                                                                                                                            |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                           | <pre>strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(RtsThresh));  wrq.u.data.length = sizeof(RtsThresh);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_RTS_THRESHOLD</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                                    |
| <b>OID_802_11_FRAGMENTATION_THRESHOLD</b> | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(FragThresh));  wrq.u.data.length = sizeof(FragThresh);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_FRAGMENTATION_THRESHOLD</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre> |
| <b>OID_802_11_POWER_MODE</b>              | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(PowerMode));  wrq.u.data.length = sizeof(PowerMode);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_POWER_MODE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                |
| <b>RT_OID_802_11_RADIO</b>                | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(RadioState));  wrq.u.data.length = sizeof(RadioState);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_RADIO</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                |
| <b>OID_802_11_INFRASTRUCTURE_MODE</b>     | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(BssType));  wrq.u.data.length = sizeof(BssType);  wrq.u.data.pointer = data;</pre>                                                                                                                         |

|                                       |                                                                                                                                                                                                                                                                              |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                       | <pre>wrq.u.data.flags = <b>OID_802_11_INFRASTRUCTURE_MODE</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                                                                                                                                                       |
| <b>RT_OID_802_11_PREAMBLE</b>         | <pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(PreamType)); wrq.u.data.length = sizeof(PreamType); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_PREAMBLE</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>       |
| <b>OID_802_11_AUTHENTICATION_MODE</b> | <pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(AuthMode)); wrq.u.data.length = sizeof(AuthMode); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_AUTHENTICATION_MODE</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre> |
| <b>OID_802_11_WEP_STATUS</b>          | <pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(WepStatus)); wrq.u.data.length = sizeof(WepStatus); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_WEP_STATUS</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>        |
| <b>OID_802_11_TX_POWER_LEVEL</b>      | <pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = <b>OID_802_11_TX_POWER_LEVEL</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>            |
| <b>OID_802_11_TX_POWER_LEVEL_1</b>    | <pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name);</pre>                                                                                                                                                                                                                 |

|                                             |                                                                                                                                                                                                                                                                                  |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                             | <pre>memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_TX_POWER_LEVEL_1</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                                                            |
| <b>OID_802_11_NETWORK_TYPES_SUPPORTED</b>   | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, 16);  wrq.u.data.length = 16;  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_NETWORK_TYPES_SUPPORTED</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>                       |
| <b>OID_802_11_NETWORK_TYPE_IN_USE</b>       | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>OID_802_11_NETWORK_TYPE_IN_USE</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre>     |
| <b>RT_OID_802_11_QUERY_EEPROM_VERSION</b>   | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_QUERY_EEPROM_VERSION</b>;  ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq);</pre> |
| <b>RT_OID_802_11_QUERY_FIRMWARE_VERSION</b> | <pre>sprintf(name, "ra0");  strcpy(wrq.ifr_name, name);  memset(data, 0, sizeof(ULONG));  wrq.u.data.length = sizeof(ULONG);  wrq.u.data.pointer = data;  wrq.u.data.flags = <b>RT_OID_802_11_QUERY_FIRMWARE_VERSION</b>;</pre>                                                  |

|                                         |                                                                                                                                                                                                                                                                           |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                         | ioctl(socket_id, <b>RT_PRIV_IOCTL</b> , &wrq);                                                                                                                                                                                                                            |
| <b>RT_OID_802_11_QUERY_NOISE_LEVEL</b>  | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UCHAR)); wrq.u.data.length = sizeof(UCHAR); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_QUERY_NOISE_LEVEL</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre> |
| <b>RT_OID_802_11_EXTRA_INFO</b>         | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_EXTRA_INFO</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>        |
| <b>RT_OID_802_11_QUERY_PIDVID</b>       | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_802_11_QUERY_PIDVID</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>      |
| <b>RT_OID_WE_VERSION_COMPILED</b>       | <pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UINT)); wrq.u.data.length = sizeof(UINT); wrq.u.data.pointer = data; wrq.u.data.flags = <b>RT_OID_WE_VERSION_COMPILED</b>; ioctl(socket_id, <b>RT_PRIV_IOCTL</b>, &amp;wrq); </pre>        |
|                                         |                                                                                                                                                                                                                                                                           |
| <b>RT_OID_802_11_QUERY_LAST_TX_RATE</b> | <pre> HTTRANSMIT_SETTING tmpHT; sprintf(wrq.ifr_name, "ra0"); </pre>                                                                                                                                                                                                      |

|                                  |                                                                                                                                                                                                                      |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | <pre> wrq.u.data.pointer = (caddr_t) &amp; tmpHT;  wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_TX_RATE;  ioctl(socket_id, RT_PRIV_IOCTL, &amp;wrq); </pre>                                                           |
| RT_OID_802_11_QUERY_LAST_RX_RATE | <pre> HTTRANSMIT_SETTING tmpHT;  sprintf(wrq.ifr_name, "ra0");  wrq.u.data.pointer = (caddr_t) &amp; tmpHT;  wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_RX_RATE;  ioctl(socket_id, RT_PRIV_IOCTL, &amp;wrq); </pre> |
| SHOW_CONN_STATUS                 | <pre> u_char buffer[IW_PRIV_SIZE_MASK];  sprintf(wrq.ifr_name, "ra0");  wrq.u.data.pointer = (caddr_t) buffer;  wrq.u.data.flags = SHOW_CONN_STATUS;  ioctl(socket_id, RTPRIV_IOCTL_SHOW, &amp;wrq); </pre>          |

### 7.2.5 Set Raw Data with Flags

| IOCTL Function                                       |                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get Raw Data by I/O Control Interface with Flags     |                                                                                                                                                                                                                                                                                                                                         |
| Function Type                                        | IOCTL                                                                                                                                                                                                                                                                                                                                   |
| RT_OID_802_11_SET_HT_PHYMODE  <br>OID_GET_SET_TOGGLE | <pre> typedef struct {      RT_802_11_PHY_MODE PhyMode;      UCHAR TransmitNo;      UCHAR HtMode; //HTMODE_GF or HTMODE_MM      UCHAR ExtOffset; //extension channel above or below      UCHAR MCS;      UCHAR BW;      UCHAR STBC;      UCHAR SHORTGI;      UCHAR rsv;  } OID_SET_HT_PHYMODE ;  RT_802_11_PHY_MODE tmp_ht_mode; </pre> |

```
printf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) & tmp_ht_mode;

wrq.u.data.length = sizeof(RT_802_11_PHY_MODE);

wrq.u.data.flags = RT_OID_802_11_SET_HT_PHYMODE |
OID_GET_SET_TOGGLE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

## 8 IOCTL INSTRUCTIONS

### 8.1 Get Data

#### 8.1.1 GET station connection status:

Linux console command: iwpriv ra0 connStatus

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) buffer;
wrq.u.data.flags = SHOW_CONN_STATUS;
ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);
```

#### 8.1.2 GET station statistics information:

Linux console command: iwpriv ra0 stat

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) buffer;
wrq.u.data.flags = 0;
ioctl(socket_id, RTPRIV_IOCTL_STATISTICS, &wrq);
```

#### 8.1.3 GET AP list table:

Linux console command: iwpriv ra0 get\_site\_survey

sample code =>

```
u_char buffer[4096];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) buffer;
wrq.u.data.flags = 0;
ioctl(socket_id, RTPRIV_IOCTL_GSITESURVEY, &wrq);
```

#### 8.1.4 GET scan table:

sample code =>

```
u_char buffer[4096];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) buffer;
wrq.u.data.length = 4096;
wrq.u.data.flags = OID_802_11_BSSID_LIST;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
PNDIS_802_11_BSSID_LIST_EX pBssidList = (PNDIS_802_11_BSSID_LIST_EX) buffer;
```

#### 8.1.5 GET station's MAC:

sample code =>

```
u_char buffer[6];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) buffer;
wrq.u.data.flags = OID_802_3_CURRENT_ADDRESS;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.6 GET station connection status:

Sample code =>

```
#define NdisMediaStateConnected 1
#define NdisMediaStateDisconnected 0
NDIS_MEDIA_STATE MediaState;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & MediaState;
wrq.u.data.flags = OID_GEN_MEDIA_CONNECT_STATUS;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.7 GET AP's BSSID

Sample code =>

```
char BSSID[6];
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) BSSID;
wrq.u.data.flags = OID_802_11_BSSID;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.8 GET SSID

Sample code =>

```
NDIS_802_11_SSID SSID;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) &SSID;
wrq.u.data.flags = OID_802_11_SSID;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.9 GET station's last TX related information:

Sample code =>

```
HTTRANSMIT_SETTING tmpHT;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) &tmpHT;
wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_TX_RATE;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.10 GET station's last RX related information:

Sample code =>

```
HTTRANSMIT_SETTING tmpHT;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) &tmpHT;
wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_RX_RATE;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.11 GET station's wireless mode:

Sample code =>

```
typedef enum _RT_802_11_PHY_MODE {
 PHY_11BG_MIXED = 0,
 PHY_11B,
 PHY_11A,
 PHY_11ABG_MIXED,
```



```

PHY_11G,
PHY_11ABGN_MIXED, // both band 5
PHY_11N, // 6
PHY_11GN_MIXED, // 2.4G band 7
PHY_11AN_MIXED, // 5G band 8
PHY_11BGN_MIXED, // if check 802.11b. 9
PHY_11AGN_MIXED, // if check 802.11b. 10
} RT_802_11_PHY_MODE

unsigned long tmp_mode;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & tmp_mode;
wrq.u.data.flags = RT_OID_802_11_PHY_MODE;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);

```

#### 8.1.12 GET Bss type:

Sample code =>

```

typedef enum _NDIS_802_11_NETWORK_INFRASTRUCTURE
{
 Ndis802_11IBSS,
 Ndis802_11Infrastructure,
 Ndis802_11AutoUnknown,
 Ndis802_11Monitor,
 Ndis802_11InfrastructureMax // Not a real value, defined as upper bound
} NDIS_802_11_NETWORK_INFRASTRUCTURE

NDIS_802_11_NETWORK_INFRASTRUCTURE BssType;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & BssType;
wrq.u.data.flags = OID_802_11_INFRASTRUCTURE_MODE;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);

```

#### 8.1.13 GET Authentication Mode:

Sample code =>

```

typedef enum _NDIS_802_11_AUTHENTICATION_MODE
{
 Ndis802_11AuthModeOpen,
 Ndis802_11AuthModeShared,
 Ndis802_11AuthModeAutoSwitch,
 Ndis802_11AuthModeWPA,
 Ndis802_11AuthModeWPAPSK,
 Ndis802_11AuthModeWPANone,
 Ndis802_11AuthModeWPA2,
 Ndis802_11AuthModeWPA2PSK,
 Ndis802_11AuthModeWPA1WPA2,
 Ndis802_11AuthModeWPA1PSKWPA2PSK,
 Ndis802_11AuthModeMax // Not a real mode, defined as upper bound
} NDIS_802_11_AUTHENTICATION_MODE

NDIS_802_11_AUTHENTICATION_MODE AuthMode;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & AuthMode;
wrq.u.data.flags = OID_802_11_AUTHENTICATION_MODE;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);

```

---

#### 8.1.14 GET Encryption Type:

Sample code =>

```
typedef enum _NDIS_802_11_WEP_STATUS
{
 Ndis802_11WEPEnabled,
 Ndis802_11Encryption1Enabled = Ndis802_11WEPEnabled,
 Ndis802_11WEPDisabled,
 Ndis802_11EncryptionDisabled = Ndis802_11WEPDisabled,
 Ndis802_11WEPKeyAbsent,
 Ndis802_11Encryption1KeyAbsent = Ndis802_11WEPKeyAbsent,
 Ndis802_11WEPNotSupported,
 Ndis802_11EncryptionNotSupported = Ndis802_11WEPNotSupported,
 Ndis802_11Encryption2Enabled,
 Ndis802_11Encryption2KeyAbsent,
 Ndis802_11Encryption3Enabled,
 Ndis802_11Encryption3KeyAbsent,
 Ndis802_11Encryption4Enabled, // TKIP or AES mix
 Ndis802_11Encryption4KeyAbsent,
} NDIS_802_11_WEP_STATUS, *PNDIS_802_11_WEP_STATUS,

NDIS_802_11_WEP_STATUS WepStatus;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & WepStatus;
wrq.u.data.flags = OID_802_11_WEP_STATUS;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.15 GET RSSI 0 (unit: db)

Sample code =>

```
long rssi_0
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & rssi_0;
wrq.u.data.flags = RT_OID_802_11_RSSI;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.16 GET RSSI 1 (unit: db)

Sample code =>

```
long rssi_1
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & rssi_1;
wrq.u.data.flags = RT_OID_802_11_RSSI_1;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.17 GET RSSI 2 (unit: db)

Sample code =>

```
long rssi_2
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & rssi_2;
wrq.u.data.flags = RT_OID_802_11_RSSI_2;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

---

#### 8.1.18 GET Driver wireless extension version

Sample code =>

```
Unsigned int wext_version;
sprintf(wrq.ifr_name, "ra0");
wrq.u.data.pointer = (caddr_t) & wext_version;
wrq.u.data.flags = RT_OID_WE_VERSION_COMPILED;
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

## 8.2 How to display rate, BW:

```

HTTRANSMIT_SETTING HTSetting;
Double Rate;
double b_mode[] = {1, 2, 5.5, 11};
float g_Rate[] = { 6,9,12,18,24,36,48,54};
switch(HTSetting.field.MODE)
{
 case 0:
 if (HTSetting.field.MCS >=0 && HTSetting.field.MCS<=3)
 Rate = b_mode[HTSetting.field.MCS];
 else if (HTSetting.field.MCS >=8 && HTSetting.field.MCS<=11)
 Rate = b_mode[HTSetting.field.MCS-8];
 else
 Rate = 0;
 break;
 case 1:
 if ((HTSetting.field.MCS >= 0) && (HTSetting.field.MCS < 8))
 Rate = g_Rate[HTSetting.field.MCS];
 else
 Rate = 0;
 break;
 case 2:
 case 3:
 if (0 == bGetHTTxRateByBW_GI_MCS(HTSetting.field.BW, HTSetting.field.ShortGI,
 HTSetting.field.MCS,
 &Rate))
 {
 Rate = 0;
 break;
 }
 default:
 Rate = 0;
 break;
}

char bGetHTTxRateByBW_GI_MCS(int nBW, int nGI, int nMCS, double* dRate)
{
 double HTTxRate20_800[16]={6.5, 13.0, 19.5, 26.0, 39.0, 52.0, 58.5, 65.0, 13.0, 26.0, 39.0, 52.0, 78.0, 104.0, 117.0, 130.0};
 double HTTxRate20_400[16]={7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65.0, 72.2, 14.444, 28.889, 43.333, 57.778, 86.667, 115.556, 130.000, 144.444};
 double HTTxRate40_800[18]={13.5, 27.0, 40.5, 54.0, 81.0, 108.0, 121.5, 135.0, 27.0, 54.0, 81.0, 108.0, 162.0, 216.0, 243.0, 270.0, 6.0, 39.0};
 double HTTxRate40_400[18]={15.0, 30.0, 45.0, 60.0, 90.0, 120.0, 135.0, 150.0, 30.0, 60.0, 90.0, 120.0, 180.0, 240.0, 270.0, 300.0, 6.7, 43.3};

 // no TxRate for (BW = 20, GI = 400, MCS = 32) & (BW = 20, GI = 400, MCS = 32)
 if (((nBW == BW_20) && (nGI == GI_400) && (nMCS == 32)) ||
 ((nBW == BW_20) && (nGI == GI_800) && (nMCS == 32)))
 return 0; //false

 if (nBW == BW_20 && nGI == GI_800)
 *dRate = HTTxRate20_800[nMCS];
 else if (nBW == BW_20 && nGI == GI_400)
 *dRate = HTTxRate20_400[nMCS];
 else if (nBW == BW_40 && nGI == GI_800)
 *dRate = HTTxRate40_800[nMCS];
 else if (nBW == BW_40 && nGI == GI_400)
 *dRate = HTTxRate40_400[nMCS];
 else
 return 0; //false

 return 1; //true
}

```