Realtek RG Rome Driver Programming Guide

Version 1.1.2

(English Version)

November 1, 2013

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park,
Hsinchu 300, Taiwan
Tel: +886-3-578-0211 Fax: +886-3-577-6047
www.realtek.com

USING THIS DOCUMENT

This document is intended for use by the system engineer when integrating with Realtek Software SDK. Though every effort has been made to assure that this document is current and accurate, more information may have become available subsequent to the production of this guide. In that event, please contact your Realtek representative for additional information that may help in the development process.

REVISION HISTORY

Revision	Date	Description	Author
1.0.1	2012/08/27	First Release.	Tony
1.0.2	2012/09/12	Modify ACL/VLAN Binding/add WAN Interface exsample. Add “Wireless LAN Hardware Forward” chapter. Add “RTK RG APIs Prototype” chapter.	Tony
1.0.3	2012/11/7	Add “MAC/ARP/NAPT” example code ACL/QoS API name is changed RTK RG API document renew.	Czpinging
1.0.4	2012/11/8	Add chapter “Checkout Code”,”Configuration”,” Build Code”,”Test by Web UI configuration”,”Test by Diagnostics Shell Command”,” Realtek RG Debug Tools”,” Test Report”	Tony
1.0.5	2012/11/9	Update Chariot Throughput Change diag shell MAC/ARP/NAPT add seq.	Czpinging
1.0.6	2013/01/01	Modify Some API prototype	Tony
1.0.7	2013/03/29	Update RG API prototype & diag shell command Update Lite RomeDriver/ RomeDriver architechure figures & description English Translation	Chuck
1.0.8	2013/05/27	Update Gateway basic configuration example Add new RG APIs & example Add Gateway Advance Configuation examples Update Build Code commands Update diagshell introduction	Chuck
1.0.9	2013/08/28	Update RG Interfcae related API & CVLAN related issue. Add RG Qos APIs & example Add Chapter for explain NAPT External Port Collision Prevention mechanism.	Chuck
1.1.0	2013/09/25	Add GPON performace test report.	Tony
1.1.1	2013/10/28	Add RG SFU API. Add RG GPON/EPON API. Update Chapter 5.1.4 Add chapter 6.2.6, 6.2.7, 6.2.8 Update diag shell command	Tony
1.1.2	2013/11/1	Add Wifi Throughtput testing result	Chuck

		Update Chater 6.1 **Update Chapter 6.2.3** **Add Chapter 6.2.4**	

Table of Contents

List of Figures

List of Tables

1. Architecture

There are two architechture provided for Router/Gateway productivities speed up L2/L3/L4 packet transmition. The first architechture is RomeDriver, and the other is Lite-RomeDriver.
If the user doesn't want to adopt hacking OS(Operating System) Protocol Stack, and insert/add APIs in the software flow, then we suggest, the user to use RomeDriver way. On the other hand, if the customer has the capability to hack OS and wants to customize his own software data flow, the Lite-RomeDriver will do the job.

1.1. RomeDriver Architecture

Structural wise, RomeDriver is adding the Forwarding Engine on Lite-RomeDriver. Forwarding Engine is a lite Protocal Stack which use to handles Layer2 MAC learning, ARP Entry aging and Napt flow forwarding. Porting RomeDriver just need to configure the informations, which is call from UI or User Application, into RTK RG API. The Data Path will be automatically maintained by Forwarding Engine. Thus, to enable the hardware forwarding application, here is the process:

1. NIC Driver RX Interface：When the hardware can't process the Packet, all data will be processed by NIC Driver, and enable the Forwarding Enging Input function.
2. NIC Driver TX Interface：NIC TX Interface is to transfer the packet after the Forwarding Engine forward a packet or add the Hardware entry
3. The packet escalates to OS Interface : Forwarding Engine will escalate the packet to OS Protocal Stack when it can not be processed
4. UI or User Application Interface: RomeDriver and LiteRomeDriver provids an 1-to-1 mapping interface for UI and User Application to call RTK RG APIs. The specific interface is used to pass the initial configuration from UI, such as LAN Interface configuration or Static WAN configuration, and dynamic configuration from User Application, such as Gateway IP Address from DHCP or PPPoE to RTK RG APIs. Besides, the callback function will be waked up in the end of RTK RG API process, if the user has registered it. On the other hand, if the user hasn't registered callback function or RTK RG API executed failed will not wake the callback function up.
5. Callback Function Interface: RTK RG API reserves hook points for registering callback functions. Callback functions is used to synchronize Hardware Tables and Protocal Stack, such as Rouing Table, and call User Application, such as DHCP Client and PPPoE Client.

Below figure shows the architechure. The thick black frame stands for the above Interfaces, the blue line stands for Data Path, and the red line stands for Control Path.

The Data Path is divided into three layers. The lowest layer represents data forwarding by Hardware ASIC, the second layer represents data forwarding by Software Forwarding Engine, and the top layer represents data forwarding by OS Protocol-Stack.

The Control Path of Hardware comes from User Space, suach as UI or User Application , or Fowrading Engine. UI or User Applications control Hardware ASIC by 1-to-1 mapping User-Space RTK RG APIs Interface.

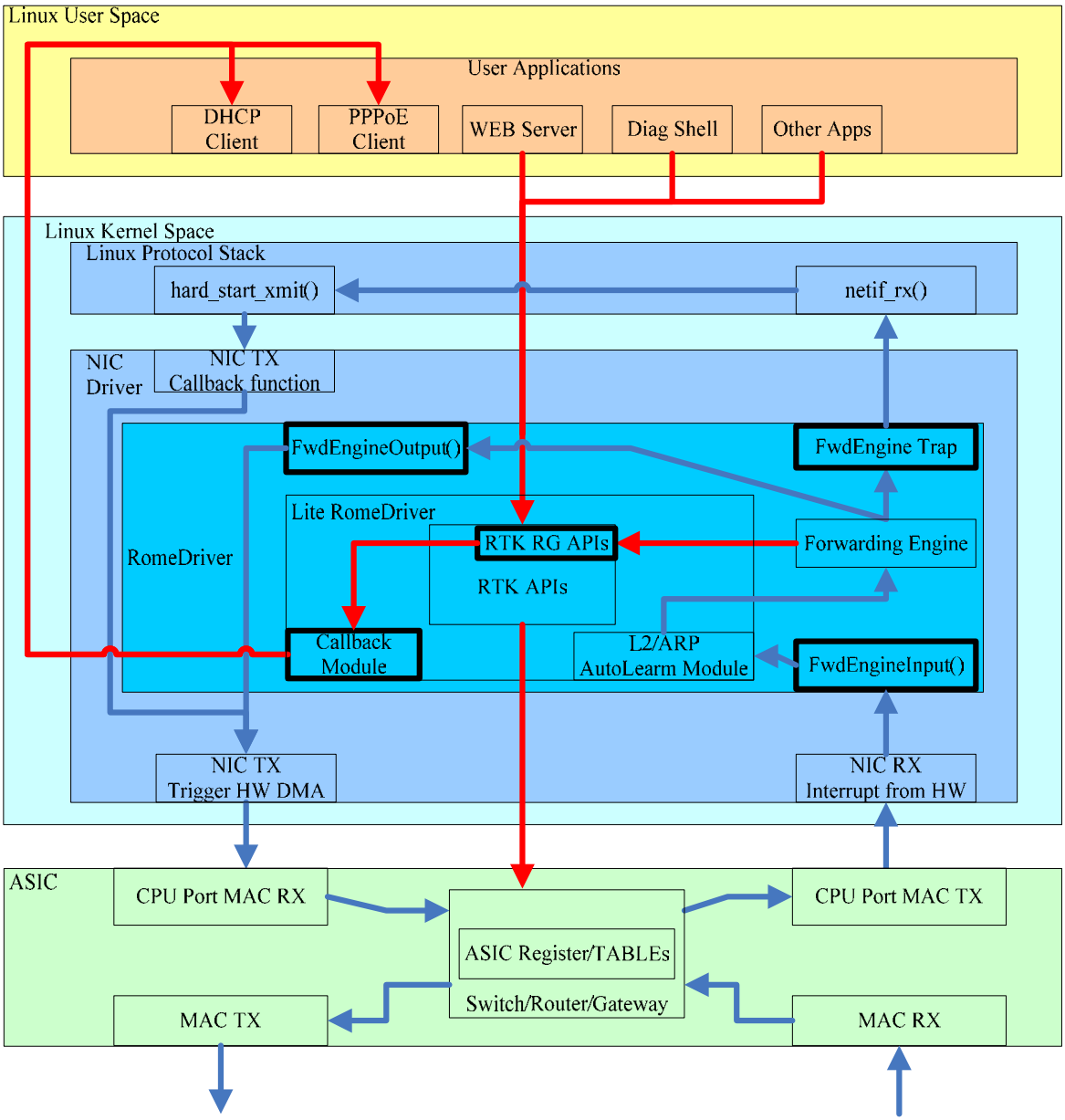


Figure1. RomeDriver Architecture

1.2. Lite RomeDriver Architecture

Lite-RomeDriver is a lighter version of RomeDriver without Forwarding Engine for which users have to call Protocal Stack and insert RTK RG API to extract data. The difference is that the Lite-RomeDriver will not maintain NAPT hardware/software related tables dynamic. Therefore, users should keep related NAPT tables, counter, and Aging Mechanism synchronous between Hardware and Protocal Stack by RTK RG APIs. Besides, Layer2 MAC & ARP Entry shoule be maintaind by L2/ARP AutoLearning Module automatically or maintaind by user calling RTK RG APIs to keep Layer2 MAC & ARP synchronous between Hardware and Protocal Stack. If L2/ARP AutoLearning Module is enabled, the Lite RomeDriver Input function should be registered in NIC RX Interface.

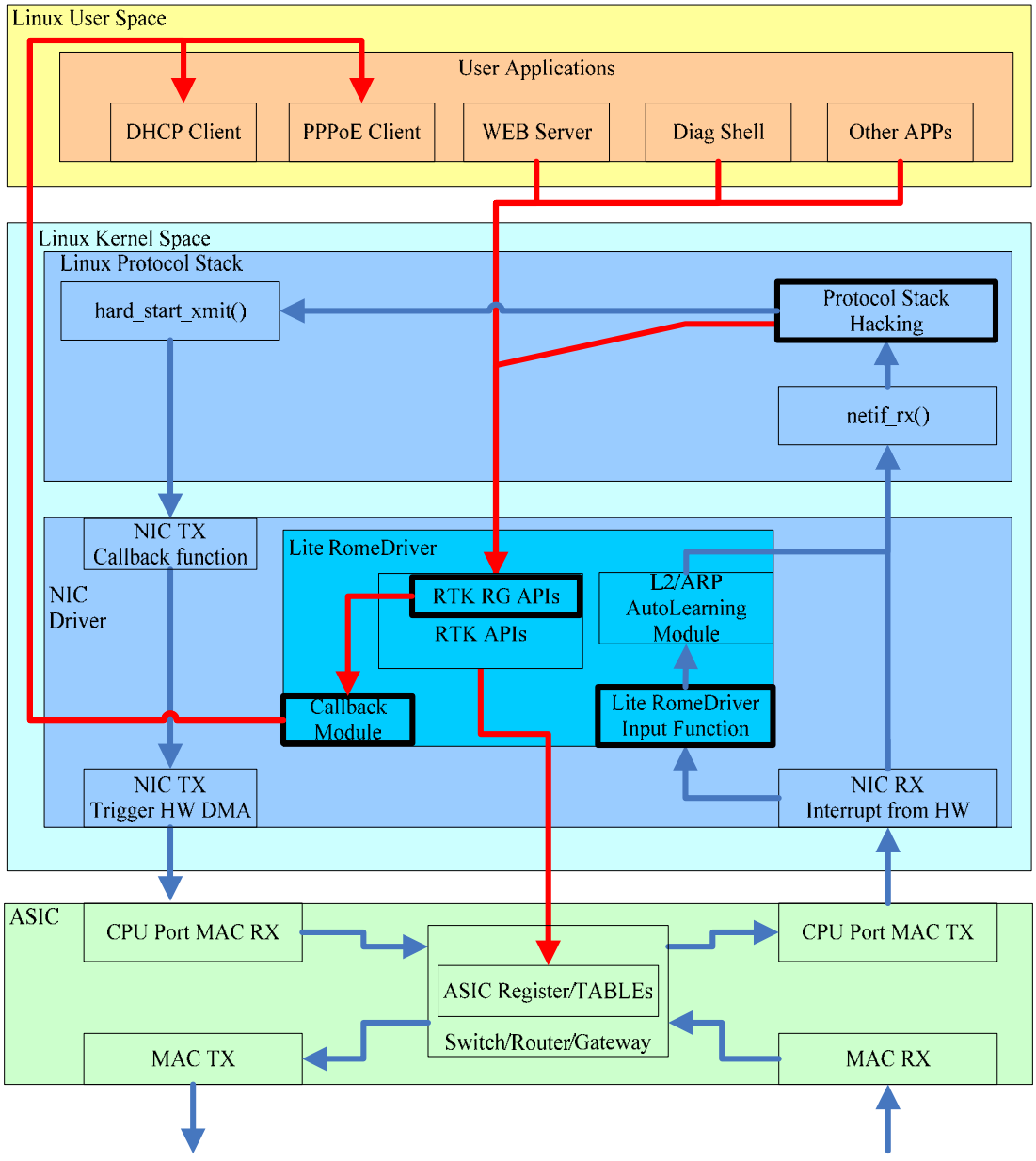


Figure2. Lite RomeDriver Architecture

2. Realtek RG APIs

There are two kinds of RG(Resident Gateway) APIs. One is pure RG APIs and the other is mapping RG APIs. Pure RG APIs are class by applicable features and focus for HGU(Home Gateway Unit) products. Mapping RG APIs are class by HW functions, they're one-to-one mapping to RTK APIs. (Please refer to RTK APIs Document to see more detail.), and focus for SFU(Single Family Unit) Product.

If your product is Router/Gateway then you can call most of pure RG APIs and some mapping RG APIs. else if your product is Switch then you can call most of mapping APIs and some RG APIs. But, you can't call RTK API directly.

The following Table lists all pure RG API according to their capabilities:

Category	R	L	S	Function Name	Function Description
System	●	●	●	rtk_rg_driverVersion_get	Get RTK RG Driver version.
				rtk_rg_initParam_get	Get the initial configuration.
				rtk_rg_initParam_set	Set initial parameters & Callback Function.
LAN Interface	●	●	●	rtk_rg_lanInterface_add	Add a LAN Interface.
				rtk_rg_interface_del	Delete a LAN Interface.
				rtk_rg_intfInfo_find	Find a LAN Interface which is added.
WAN Interface/ Port Binding	●	●	●	rtk_rg_wanInterface_add	Add a WAN Interface.
				rtk_rg_interface_del	Delete a WAN Interface.
				rtk_rg_intfInfo_find	Find a WAN Interface which is added.
CVLAN	●	●	●	rtk_rg_cvlan_add	Add an individual C-VLAN.
				rtk_rg_cvlan_del	Delete an individual C-VLAN.
				rtk_rg_portBasedCVlanId_get	Get Port-based VLAN ID.
WAN Type	●	●	○	rtk_rg_staticInfo_set	Set WAN Type:STATIC IP information.
				rtk_rg_dhcpRequest_set	Set WAN Type:DHCP request information.
				rtk_rg_dhcpClientInfo_set	Set WAN Type:DHCP IP information.
				rtk_rg_pppoeClientInfoBefore	Set WAN Type:PPPoE dial

				Dial_set	information.
				rtk_rg_pppoeClientInfoAfterDial_set	Set WAN Type:PPPoE IP information.
VLAN Binding	●	●	●	rtk_rg_vlanBinding_add	Insert a VLAN Binding Entry.
				rtk_rg_vlanBinding_del	Delete a VLAN Binding Entry.
				rtk_rg_vlanBinding_find	Finda VLAN Binding Entry which is added.
ALG	●	○	○	rtk_rg_algApps_set	Set the starting ALG Appication type.
				rtk_rg_algApps_get	Set the current ALG Appication type.
				rtk_rg_algServerInLanAppsIpAddr_add	Set the Server-in-LAN ALG Server IP Address.
				rtk_rg_algServerInLanAppsIpAddr_del	Get the Server-in-LAN ALG Server IP Address.
Virtual Server	●	○	○	rtk_rg_virtualServer_add	Add a Virtual Server Rule.
				rtk_rg_virtualServer_del	Delete a Virtual Server Rule.
				rtk_rg_virtualServer_find	Find a Virtual Server Rule.
DMZ	●	○	○	rtk_rg_dmzHost_set	Set DMZ Host to assigned WAN Interface.
				rtk_rg_dmzHost_get	Get DMZ Host information from assigned WAN Interface.
UPnP	●	○	○	rtk_rg_upnpConnection_add	Add an uPnP connection.
				rtk_rg_upnpConnection_del	Delete an uPnP connection.
				rtk_rg_upnpConnection_find	Find an uPnP connection which is added.
NAPT	○	●	○	rtk_rg_naptConnection_add	Add a NAPT connection.
				rtk_rg_naptConnection_del	Delete a NAPT connection.
				rtk_rg_naptConnection_find	Find a NAPT connection which is added.
NAPT External Port	●	○	○	rtk_rg_naptExtPortFree	For Sync with Linux & RomeDriver.
				rtk_rg_naptExtPortGet	For Sync with Linux & RomeDriver.
MAC	○	●	●	rtk_rg_macEntry_add	Add a MAC Entry.
				rtk_rg_macEntry_del	Delete a MAC Entry.

				rtk_rg_macEntry_find	Find a MAC Entry wich is added.
Multicast	○	●	●	rtk_rg_multicastFlow_add	Add a Multicast connection.
				rtk_rg_multicastFlow_del	Delete a Multicast connection.
				rtk_rg_multicastFlow_find	Find a Multicast connection which is added.
ARP	○	●	○	rtk_rg_arpEntry_add	Add an ARP Entry.
				rtk_rg_arpEntry_del	Delete an ARP Entry.
				rtk_rg_arpEntry_find	Find an ARP Entry which is added.
Neighbor	○	●	○	rtk_rg_neighborEntry_add	Add an Neighbor Entry
				rtk_rg_neighborEntry_del	Delete an Neighbor Entry
				rtk_rg_neighborEntry_find	Find an Neighbor Entry which is added.
ACL Filter and QoS Remarking	●	●	●	rtk_rg_aclFilterAndQos_add	Add an ACL Rule(include QoS Remarking).
				rtk_rg_aclFilterAndQos_del	Delete an ACL Rule(include QoS Remarking).
				rtk_rg_aclFilterAndQos_find	Find an ACL Rule which is added.
L2 Classification	●	●	●	rtk_rg_classifyEntry_add	Add a L2 CF rule.
				rtk_rg_classifyEntry_del	Delete a L2 CF rule.
				rtk_rg_classifyEntry_find	Find a L2 CF rule which is added.
MAC Filter	●	●	●	rtk_rg_macFilter_add	Add a MAC Filter Entry.
				rtk_rg_macFilter_del	Delete a MAC Filter Entry.
				rtk_rg_macFilter_find	Find a MAC Filter Entry which is added.
URL Filter	●	○	○	rtk_rg_urlFilterString_add	Add a URLFilter Entry.
				rtk_rg_urlFilterString_del	Delete a URLFilter Entry.
				rtk_rg_urlFilterString_find	Find a URLFilter Entry which is added.
StormControl	●	●	●	rtk_rg_stormControl_add	Add an StormControl Entry
				rtk_rg_stormControl_del	Delete an StormControl Entry
				rtk_rg_stormControl_find	Find an StormControl Entry which is added.
PortMirror	●	●	●	rtk_rg_portMirror_set	Set portMirror
				rtk_rg_portMirror_get	Get portMirror which has been set
				rtk_rg_portMirror_clear	Clear portMirror
Egress	●	●	●	rtk_rg_portEgrBandwidthCtrlR	Set Egress Bandwidth Control rate

Bandwidth Control				ate_set	
				rtk_rg_portEgrBandwidthCtrlRate_get	Get Egress Bandwidth Control rate which has been set
Ingress Bandwidth Control	●	●	●	rtk_rg_portIgrBandwidthCtrlRate_set	Set Ingress Bandwidth Control rate
				rtk_rg_portIgrBandwidthCtrlRate_get	Get Ingress Bandwidth Control rate which has been set
Physical Port	●	●	●	rtk_rg_phyPortForceAbility_set	Set Physical Port Ability
				rtk_rg_phyPortForceAbility_get	Get Physical Port Ability which has been set
				rtk_rg_portStatus_get	Get Physical Port link status.
ShareMeter	●	●	●	rtk_rg_shareMeter_set	Set an ShareMeter Entry.
				rtk_rg_shareMeter_get	Get an ShareMeter Entry which has been set
DoS	●	●	●	rtk_rg_dosPortMaskEnable_set	Enable DosPort
				rtk_rg_dosPortMaskEnable_get	Get Enabled DosPort
				rtk_rg_dosType_set	Set a dosType information
				rtk_rg_dosType_get	Get a dosType information which has been set
				rtk_rg_dosFloodType_set	Set a dosFloodType information
				rtk_rg_dosFloodType_get	Get a dosFloodType information which has been set
MAC Learning Limit	●	○	●	rtk_rg_softwareSourceAddrLearningLimit_get	Get software per-Port MAC learning limit.
				rtk_rg_softwareSourceAddrLearningLimit_set	Set software per-Port MAC learning limit.
MIB	●	●	●	rtk_rg_portMibInfo_clear	Clear per-Port MIB info.
				rtk_rg_portMibInfo_get	Get per-Port MIB info.
QoS Internal Priority Decision	●	●	●	rtk_rg_qosDot1pPriRemapToInternalPri_get	CVLAN-Priority remapping table get
				rtk_rg_qosDot1pPriRemapToInternalPri_set	CVLAN-Priority remapping table set
				rtk_rg_qosDscpRemapToInternalPri_get	DSCP remapping table get.
				rtk_rg_qosDscpRemapToInter	DSCP remapping table set

				nalPri_set	
				rtk_rg_qosInternalPriDecisionByWeight_get	Get the weight of the 9 kinds of priority decision source.
				rtk_rg_qosInternalPriDecisionByWeight_set	Set the weight of the 9 kinds of priority decision source.
				rtk_rg_qosPortBasedPriority_get	Get port based priority
				rtk_rg_qosPortBasedPriority_set	Set port based priority
QoS Egress Queue Decision	●	●	●	rtk_rg_qosInternalPriMapToQueueId_get	Get the “Internal priority to Queue” table.
				rtk_rg_qosInternalPriMapToQueueId_set	Set the “Internal priority to Queue” table.
QoS Remarking	●	●	●	rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get	CVLAN-Priority remarking egress port get.
				rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set	CVLAN-Priority remarking egress port set.
				rtk_rg_qosDot1pPriRemarkByInternalPri_get	CVLAN-Priority remarking table get
				rtk_rg_qosDot1pPriRemarkByInternalPri_set	CVLAN-Priority remarking table set
				rtk_rg_qosDscpRemarkByDscp_get	Get the “DSCP remarking by DSCP” mapping table.
				rtk_rg_qosDscpRemarkByDscp_set	Set the “DSCP remarking by DSCP” mapping table.
				rtk_rg_qosDscpRemarkByInternalPri_get	Get the “DSCP remarking by Internal Priority” mapping table.
				rtk_rg_qosDscpRemarkByInternalPri_set	Set the “DSCP remarking by Internal Priority” mapping table.
				rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get	Get DSCP remarking egress port.
				rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set	Set DSCP remarking egress port.
QoS Scheduling	●	●	●	rtk_rg_qosStrictPriorityOrWeightFairQueue_get	Get QoS Scheduling method (WFQ,Strict Priority)

by Queues				rtk_rg_qosStrictPriorityOrWeightFairQueue_set	Set QoS Scheduling method (WFQ,Strict Priority)

Table1. List of pure RG APIs

PS: In the Above field, “R” stands for RomeDriver API, “L” stands for Lite RomeDriver API, “S” stands for SFU Product used API.

The following Table lists all mapping RG APIs:

STP	rtk_rg_stp_init
	rtk_rg_stp_mstpState_get
	rtk_rg_stp_mstpState_set
GPON	rtk_rg_gpon_activate
	rtk_rg_gpon_alarmStatus_get
	rtk_rg_gpon_autoBoh_get
	rtk_rg_gpon_autoBoh_set
	rtk_rg_gpon_autoTcont_get
	rtk_rg_gpon_autoTcont_set
	rtk_rg_gpon_broadcastPass_get
	rtk_rg_gpon_broadcastPass_set
	rtk_rg_gpon_callbackExtMsgGetHandle_reg
	rtk_rg_gpon_callbackExtMsgSetHandle_reg
	rtk_rg_gpon_callbackQueryAesKey_reg
	rtk_rg_gpon_deActivate
	rtk_rg_gpon_debug_set
	rtk_rg_gpon_deinitial
	rtk_rg_gpon_device_deInitialize
	rtk_rg_gpon_device_initialize
	rtk_rg_gpon_devInfo_show
	rtk_rg_gpon_driver_deInitialize
	rtk_rg_gpon_driver_initialize
	rtk_rg_gpon_dsFecSts_get
	rtk_rg_gpon_dsFlow_get
	rtk_rg_gpon_dsFlow_set
	rtk_rg_gpon_dsFlow_show
	rtk_rg_gpon_eqdOffset_get
	rtk_rg_gpon_eqdOffset_set
	rtk_rg_gpon_evtHdlAlarm_reg

rtk_rg_gpon_evtHdlDsFecChange_reg
rtk_rg_gpon_evtHdlOmci_reg
rtk_rg_gpon_evtHdlPloam_dreg
rtk_rg_gpon_evtHdlPloam_reg
rtk_rg_gpon_evtHdlStateChange_reg
rtk_rg_gpon_evtHdlUsFecChange_reg
rtk_rg_gpon_evtHdlUsPloamNrmEmpty_reg
rtk_rg_gpon_evtHdlUsPloamUrgEmpty_reg
rtk_rg_gpon_extMsg_get
rtk_rg_gpon_extMsg_set
rtk_rg_gpon_flowCounter_get
rtk_rg_gpon_flowCounter_show
rtk_rg_gpon_globalCounter_get
rtk_rg_gpon_globalCounter_show
rtk_rg_gpon_gtc_show
rtk_rg_gpon_initial
rtk_rg_gpon_isr_entry
rtk_rg_gpon_macEntry_add
rtk_rg_gpon_macEntry_del
rtk_rg_gpon_macEntry_get
rtk_rg_gpon_macFilterMode_get
rtk_rg_gpon_macFilterMode_set
rtk_rg_gpon_macTable_show
rtk_rg_gpon_mcForceMode_get
rtk_rg_gpon_mcForceMode_set
rtk_rg_gpon_multicastAddrCheck_get
rtk_rg_gpon_multicastAddrCheck_set
rtk_rg_gpon_nonMcastPass_get
rtk_rg_gpon_nonMcastPass_set
rtk_rg_gpon_omci_rx
rtk_rg_gpon_omci_tx
rtk_rg_gpon_parameter_get
rtk_rg_gpon_parameter_set
rtk_rg_gpon_password_get
rtk_rg_gpon_password_set
rtk_rg_gpon_ploam_send
rtk_rg_gpon_ponStatus_get

	rtk_rg_gpon_powerLevel_get
	rtk_rg_gpon_powerLevel_set
	rtk_rg_gpon_rdi_get
	rtk_rg_gpon_rdi_set
	rtk_rg_gpon_serialNumber_get
	rtk_rg_gpon_serialNumber_set
	rtk_rg_gpon_tcontCounter_get
	rtk_rg_gpon_tcontCounter_show
	rtk_rg_gpon_tcont_create
	rtk_rg_gpon_tcont_destroy
	rtk_rg_gpon_tcont_get
	rtk_rg_gpon_tcont_show
	rtk_rg_gpon_txForceIdle_get
	rtk_rg_gpon_txForceIdle_set
	rtk_rg_gpon_txForceLaser_get
	rtk_rg_gpon_txForceLaser_set
	rtk_rg_gpon_unit_test
	rtk_rg_gpon_usFlow_get
	rtk_rg_gpon_usFlow_set
	rtk_rg_gpon_usFlow_show
	rtk_rg_gpon_version_get
	rtk_rg_gpon_version_show
EPON	rtk_rg_epon_churningKey_get
	rtk_rg_epon_churningKey_set
	rtk_rg_epon_dsFecState_get
	rtk_rg_epon_dsFecState_set
	rtk_rg_epon_fecState_get
	rtk_rg_epon_fecState_set
	rtk_rg_epon_forceLaserState_get
	rtk_rg_epon_forceLaserState_set
	rtk_rg_epon_init
	rtk_rg_epon_intrMask_get
	rtk_rg_epon_intrMask_set
	rtk_rg_epon_intr_disableAll
	rtk_rg_epon_intr_get
	rtk_rg_epon_laserTime_get
	rtk_rg_epon_laserTime_set

	rtk_rg_epon_llidEntryNum_get
	rtk_rg_epon_llid_entry_get
	rtk_rg_epon_llid_entry_set
	rtk_rg_epon_losState_get
	rtk_rg_epon_mibCounter_get
	rtk_rg_epon_mibGlobal_reset
	rtk_rg_epon_mibLlidIdx_reset
	rtk_rg_epon_mpcpTimeoutVal_get
	rtk_rg_epon_mpcpTimeoutVal_set
	rtk_rg_epon_opticalPolarity_get
	rtk_rg_epon_opticalPolarity_set
	rtk_rg_epon_registerReq_get
	rtk_rg_epon_registerReq_set
	rtk_rg_epon_syncTime_get
	rtk_rg_epon_thresholdReport_get
	rtk_rg_epon_thresholdReport_set
	rtk_rg_epon_usFecState_get
	rtk_rg_epon_usFecState_set
PON MAC	rtk_rg_ponmac_flow2Queue_get
	rtk_rg_ponmac_flow2Queue_set
	rtk_rg_ponmac_init
	rtk_rg_ponmac_losState_get
	rtk_rg_ponmac_mode_get
	rtk_rg_ponmac_mode_set
	rtk_rg_ponmac_opticalPolarity_get
	rtk_rg_ponmac_opticalPolarity_set
	rtk_rg_ponmac_queueDrainOut_set
	rtk_rg_ponmac_queue_add
	rtk_rg_ponmac_queue_del
	rtk_rg_ponmac_queue_get
	rtk_rg_ponmac_transceiver_get
I2C	rtk_rg_i2c_clock_get
	rtk_rg_i2c_clock_set
	rtk_rg_i2c_eepMirror_get
	rtk_rg_i2c_eepMirror_read
	rtk_rg_i2c_eepMirror_set
	rtk_rg_i2c_eepMirror_write

	rtk_rg_i2c_enable_get
	rtk_rg_i2c_enable_set
	rtk_rg_i2c_init
	rtk_rg_i2c_read
	rtk_rg_i2c_width_get
	rtk_rg_i2c_width_set
	rtk_rg_i2c_write
Interrupt	rtk_rg_intr_gphyStatus_clear
	rtk_rg_intr_gphyStatus_get
	rtk_rg_intr_imr_get
	rtk_rg_intr_imr_restore
	rtk_rg_intr_imr_set
	rtk_rg_intr_ims_clear
	rtk_rg_intr_ims_get
	rtk_rg_intr_init
	rtk_rg_intr_linkdownStatus_clear
	rtk_rg_intr_linkdownStatus_get
	rtk_rg_intr_linkupStatus_clear
	rtk_rg_intr_linkupStatus_get
	rtk_rg_intr_polarity_get
	rtk_rg_intr_polarity_set
	rtk_rg_intr_speedChangeStatus_clear
	rtk_rg_intr_speedChangeStatus_get
Port	rtk_rg_port_adminEnable_get
	rtk_rg_port_adminEnable_set
	rtk_rg_port_cpuPortId_get
	rtk_rg_port_enhancedFid_get
	rtk_rg_port_enhancedFid_set
	rtk_rg_port_flowctrl_get
	rtk_rg_port_gigaLiteEnable_get
	rtk_rg_port_gigaLiteEnable_set
	rtk_rg_port_greenEnable_get
	rtk_rg_port_greenEnable_set
	rtk_rg_port_init
	rtk_rg_port_isolationCtagPktConfig_get
	rtk_rg_port_isolationCtagPktConfig_set
	rtk_rg_port_isolationEntryExt_get

	rtk_rg_port_isolationEntryExt_set
	rtk_rg_port_isolationEntry_get
	rtk_rg_port_isolationEntry_set
	rtk_rg_port_isolationExtL34_get
	rtk_rg_port_isolationExtL34_set
	rtk_rg_port_isolationExt_get
	rtk_rg_port_isolationExt_set
	rtk_rg_port_isolationIpmcLeaky_get
	rtk_rg_port_isolationIpmcLeaky_set
	rtk_rg_port_isolationL34PktConfig_get
	rtk_rg_port_isolationL34PktConfig_set
	rtk_rg_port_isolationL34_get
	rtk_rg_port_isolationL34_set
	rtk_rg_port_isolationLeaky_get
	rtk_rg_port_isolationLeaky_set
	rtk_rg_port_isolationPortLeaky_get
	rtk_rg_port_isolationPortLeaky_set
	rtk_rg_port_isolation_get
	rtk_rg_port_isolation_set
	rtk_rg_port_link_get
	rtk_rg_port_macExtMode_get
	rtk_rg_port_macExtMode_set
	rtk_rg_port_macExtRgmiiDelay_get
	rtk_rg_port_macExtRgmiiDelay_set
	rtk_rg_port_macForceAbilityState_get
	rtk_rg_port_macForceAbilityState_set
	rtk_rg_port_macForceAbility_get
	rtk_rg_port_macForceAbility_set
	rtk_rg_port_macLocalLoopbackEnable_get
	rtk_rg_port_macLocalLoopbackEnable_set
	rtk_rg_port_macRemoteLoopbackEnable_get
	rtk_rg_port_macRemoteLoopbackEnable_set
	rtk_rg_port_phyAutoNegoAbility_get
	rtk_rg_port_phyAutoNegoAbility_set
	rtk_rg_port_phyAutoNegoEnable_get
	rtk_rg_port_phyAutoNegoEnable_set
	rtk_rg_port_phyCrossOverMode_get

	rtk_rg_port_phyCrossOverMode_set
	rtk_rg_port_phyForceModeAbility_get
	rtk_rg_port_phyForceModeAbility_set
	rtk_rg_port_phyMasterSlave_get
	rtk_rg_port_phyMasterSlave_set
	rtk_rg_port_phyReg_get
	rtk_rg_port_phyReg_set
	rtk_rg_port_phyTestMode_get
	rtk_rg_port_phyTestMode_set
	rtk_rg_port_rtctResult_get
	rtk_rg_port_rtct_start
	rtk_rg_port_specialCongestStatus_clear
	rtk_rg_port_specialCongestStatus_get
	rtk_rg_port_specialCongest_get
	rtk_rg_port_specialCongest_set
	rtk_rg_port_speedDuplex_get
Rate	rtk_rg_rate_egrBandwidthCtrlIncludeIfg_get
	rtk_rg_rate_egrBandwidthCtrlIncludeIfg_set
	rtk_rg_rate_egrQueueBwCtrlEnable_get
	rtk_rg_rate_egrQueueBwCtrlEnable_set
	rtk_rg_rate_egrQueueBwCtrlMeterIdx_get
	rtk_rg_rate_egrQueueBwCtrlMeterIdx_set
	rtk_rg_rate_init
	rtk_rg_rate_portEgrBandwidthCtrlIncludeIfg_get
	rtk_rg_rate_portEgrBandwidthCtrlIncludeIfg_set
	rtk_rg_rate_portEgrBandwidthCtrlRate_get
	rtk_rg_rate_portEgrBandwidthCtrlRate_set
	rtk_rg_rate_portIgrBandwidthCtrlIncludeIfg_get
	rtk_rg_rate_portIgrBandwidthCtrlIncludeIfg_set
	rtk_rg_rate_portIgrBandwidthCtrlRate_get
	rtk_rg_rate_portIgrBandwidthCtrlRate_set
	rtk_rg_rate_shareMeterBucket_get
	rtk_rg_rate_shareMeterBucket_set
	rtk_rg_rate_shareMeterExceed_clear
	rtk_rg_rate_shareMeterExceed_get
	rtk_rg_rate_shareMeterMode_get
	rtk_rg_rate_shareMeterMode_set

	rtk_rg_rate_shareMeter_get
	rtk_rg_rate_shareMeter_set
	rtk_rg_rate_stormBypass_get
	rtk_rg_rate_stormBypass_set
	rtk_rg_rate_stormControlEnable_get
	rtk_rg_rate_stormControlEnable_set
	rtk_rg_rate_stormControlMeterIdx_get
	rtk_rg_rate_stormControlMeterIdx_set
	rtk_rg_rate_stormControlPortEnable_get
	rtk_rg_rate_stormControlPortEnable_set
RLDP	rtk_rg_rldp_config_get
	rtk_rg_rldp_config_set
	rtk_rg_rldp_init
	rtk_rg_rldp_portConfig_get
	rtk_rg_rldp_portConfig_set
	rtk_rg_rldp_portStatus_clear
	rtk_rg_rldp_portStatus_get
	rtk_rg_rldp_status_get
	rtk_rg_rlpp_init
	rtk_rg_rlpp_trapType_get
	rtk_rg_rlpp_trapType_set
Statistics	rtk_rg_stat_global_get
	rtk_rg_stat_global_getAll
	rtk_rg_stat_global_reset
	rtk_rg_stat_init
	rtk_rg_stat_logCtrl_get
	rtk_rg_stat_logCtrl_set
	rtk_rg_stat_log_get
	rtk_rg_stat_log_reset
	rtk_rg_stat_mibCntMode_get
	rtk_rg_stat_mibCntMode_set
	rtk_rg_stat_mibCntTagLen_get
	rtk_rg_stat_mibCntTagLen_set
	rtk_rg_stat_mibLatchTimer_get
	rtk_rg_stat_mibLatchTimer_set
	rtk_rg_stat_mibSyncMode_get
	rtk_rg_stat_mibSyncMode_set

	rtk_rg_stat_pktInfo_get
	rtk_rg_stat_port_get
	rtk_rg_stat_port_getAll
	rtk_rg_stat_port_reset
	rtk_rg_stat_rstCntValue_get
	rtk_rg_stat_rstCntValue_set
Switch	rtk_rg_switch_allExtPortMask_set
	rtk_rg_switch_allPortMask_set
	rtk_rg_switch_chip_reset
	rtk_rg_switch_deviceInfo_get
	rtk_rg_switch_init
	rtk_rg_switch_logicalPort_get
	rtk_rg_switch_maxPktLenLinkSpeed_get
	rtk_rg_switch_maxPktLenLinkSpeed_set
	rtk_rg_switch_mgmtMacAddr_get
	rtk_rg_switch_mgmtMacAddr_set
	rtk_rg_switch_nextPortInMask_get
	rtk_rg_switch_phyPortId_get
	rtk_rg_switch_port2PortMask_clear
	rtk_rg_switch_port2PortMask_set
	rtk_rg_switch_portIdInMask_check
	rtk_rg_switch_portMask_Clear
	rtk_rg_switch_version_get
Trap	rtk_rg_trap_igmpCtrlPkt2CpuEnable_get
	rtk_rg_trap_igmpCtrlPkt2CpuEnable_set
	rtk_rg_trap_init
	rtk_rg_trap_ipMcastPkt2CpuEnable_get
	rtk_rg_trap_ipMcastPkt2CpuEnable_set
	rtk_rg_trap_l2McastPkt2CpuEnable_get
	rtk_rg_trap_l2McastPkt2CpuEnable_set
	rtk_rg_trap_mldCtrlPkt2CpuEnable_get
	rtk_rg_trap_mldCtrlPkt2CpuEnable_set
	rtk_rg_trap_oamPduAction_get
	rtk_rg_trap_oamPduAction_set
	rtk_rg_trap_oamPduPri_get
	rtk_rg_trap_oamPduPri_set
	rtk_rg_trap_portIgmpMldCtrlPktAction_get

	rtk_rg_trap_portIgmpMldCtrlPktAction_set
	rtk_rg_trap_reasonTrapToCpuPriority_get
	rtk_rg_trap_reasonTrapToCpuPriority_set
	rtk_rg_trap_rmaAction_get
	rtk_rg_trap_rmaAction_set
	rtk_rg_trap_rmaPri_get
	rtk_rg_trap_rmaPri_set

Table2. List of mapping RG APIs (Mapping from RTK APIs)

Each pure RG API has access to corresponding Hardware Table as follow:

Category	Function Name	L2	VL	AR	NI	L3	NH	NB	BI	WA	IP	PP	NA	AC	CF
	rtk_rg_initParam_set	●	●	●	●	●	●	●	●	●	●	●	●	●	●
LAN Intf	rtk_rg_lanInterface_add		●		●	●									
	rtk_rg_interface_del		●		●	●									
WAN Intf/ Port Binding	rtk_rg_wanInterface_add		●		●	●	●		●	●		●			
	rtk_rg_interface_del		●		●	●	●		●	●		●			
WAN Type	rtk_rg_staticInfo_set	●		●		●		●			●				
	rtk_rg_dhcpClientInfo_set	●		●		●		●			●				
	rtk_rg_pppoeClientInfoAfterDial_set	●		●		●		●			●	●			
VLAN Binding	rtk_rg_vlanBinding_add								●						
	rtk_rg_vlanBinding_del								●						
ACL/ Qos Remarking	rtk_rg_aclFilterAndQos_add													●	●
	rtk_rg_aclFilterAndQos_del													●	●
MAC Filter	rtk_rg_macFilter_add	●													
	rtk_rg_macFilter_del	●													
URL Filter	rtk_rg_urlFilterString_add													●	
	rtk_rg_urlFilterString_del													●	
Multi-cast	rtk_rg_multicastFlow_add	●													
	rtk_rg_multicastFlow_del	●													
	rtk_rg_multicastFlow_find	●													
NAPT	rtk_rg_naptConnection_add												●		

	rtk_rg_naptConnection_del												●		
MAC	rtk_rg_macEntry_add	●													
	rtk_rg_macEntry_del	●													
ARP	rtk_rg_arpEntry_add	●		●											
	rtk_rg_arpEntry_del	●		●											
Neigh-bor	rtk_rg_neighborEntry_add	●						●							
	rtk_rg_neighborEntry_del	●						●							

Table3. List of RTK RG HGU APIs access Hardware Table

PS: Glossary for the above field as below:

L2: LUT Table	VL: CVLAN Table	AR: ARP Table
NI: Network Interface Table	L3: Routing Table	NH: Next Hop Table
BI: Binding Table	WA: WAN Type Table	IP: Internal/External IP Table
PP: PPPoE Table	NA: NAPT Table	AC: ACL Table
CF: Classification Table	NB:Neighbor Table	

3. Checkout Code

The developer should Checkout Code from SVN Server:

$ **mkdir ~/RTK**

$ **cd ~/RTK**

$ **svn co http://dtdinfo.realtek.com.tw/svn/CN/lunar/trunk/develop/uClinux-dist**

Customers should acquire source code package from Realtek FTP.(172.21.146.57).

Please get the FTP account information from our FAE.

4. Build Code

Following command shows how to build source code to Image:

Read the default configuration setting:

```
$ cd uClinux-dist
$ make -f LDLMakefile preconfig2630_DL8696RG_VoIP_demo CONFIG_PRODUCT=luna
```

Make menuconfig. If unknown config shows, just press Enter(which means using default setting).

```
$make -f LDLMakefile menuconfig
```

Compile Kernal & User space code and to make vm.img:

```
$make -f LDLMakefile all
```

5. NAPT External Port Collsion Prevention

In Current SDK, Forwarding Engine can not support some advanced flow connection, such as User-defined ALG., Out of RomeDriver NAPT range flows. So, these kind of connection packet will be trapped to Protocal-stack and handled by OS(Operating System). The OS will establish and maintain the connection by its own networking module. While establish TCP/UDP NAPT connetion, no matter forwardingEngine or OS take care of this, it must choose an external port. In cuurent SDK, both forwardingEngine and OS maintains these NAPT flow by Symmetric way, and each of them are independent. For avoiding the confliction of the same external port choosed by forwardingEngine and OS but establish different NAPT flow, we must provide a NAPT External Port Collision Prevention Mechanism in RomeDriver to make sure each side will not choose the same external port.

If using the NAPT External Port Collision Prevention Mechanism in RomeDriver, the user has to modify the OS protocol stack. Considering the transplating in different OS, and the maintaince in RomeDriver, here we provide two API in RomeDriver for modifying OS easier.

- rtk_rg_naptExtPortGet (Please reference to 5.1.2)
- rtk_rg_naptExtPortFree (Please reference to 5.1.3)

5.1.1. Enable/Disable NAPT External Port Collision Prevention

The NAPT External Port Collision Prevention Mechanism can be enabled by compile flag CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION.

In Linux, We can enable this feature by menuconfig as following:

Step1:Choosing the Kernal Setting configuration & Exit the main menuconfig:

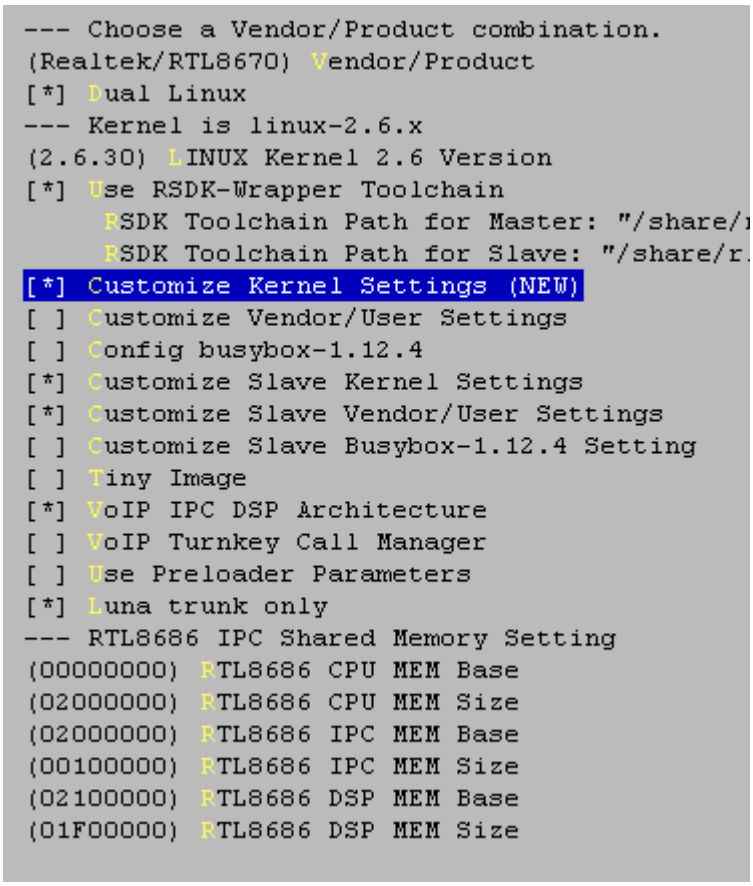


Figure3. Enable NAPT External Port Collision Prevention, step1

Step2: Enter to “Device Drivers”

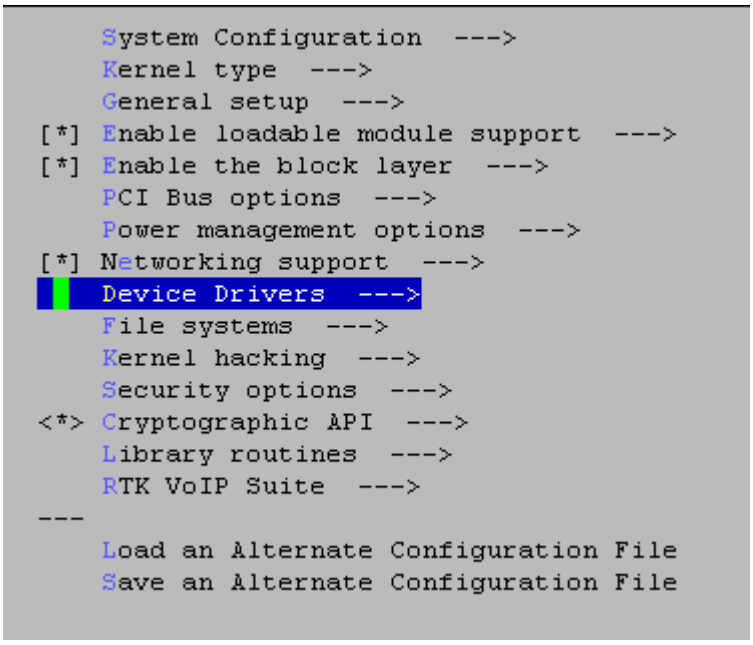


Figure4. Enable NAPT External Port Collision Prevention, step2

Step3: Enter to “Realtek RG support”

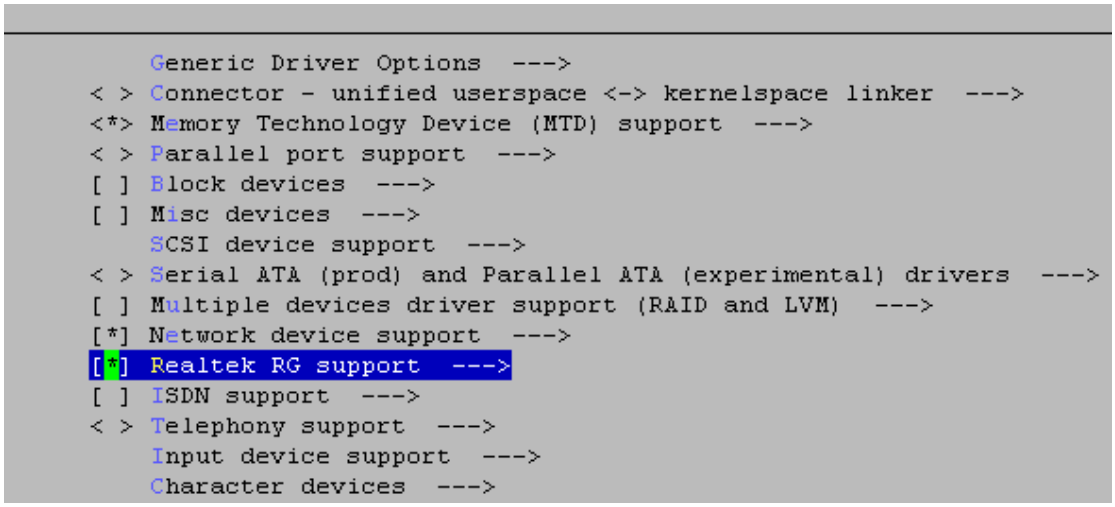


Figure5. Enable NAPT External Port Collision Prevention, step3

Step4: Enable “Realtek RG NAPT external port collision prevenction.”

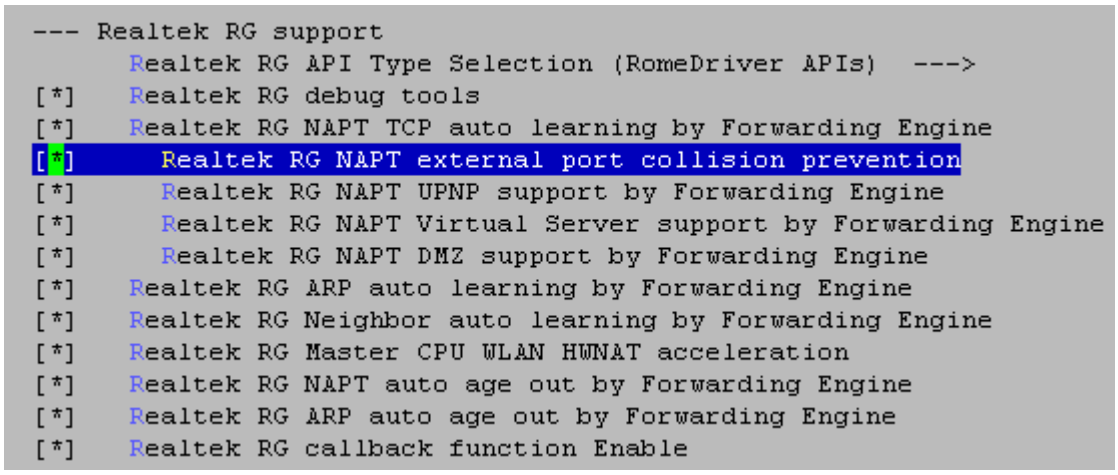


Figure6. Enable NAPT External Port Collision Prevention, step4

5.1.2. API: rtk_rg_naptExtPortGet

This API maintain an external port databased and check the choosed external port is used by other NAPT connetion or UPNP/Virtual Server or not. If the choosed external port is used, then the API will auto increase the port number and check again until find a legal port number. Following figure shows the external port checking flow.

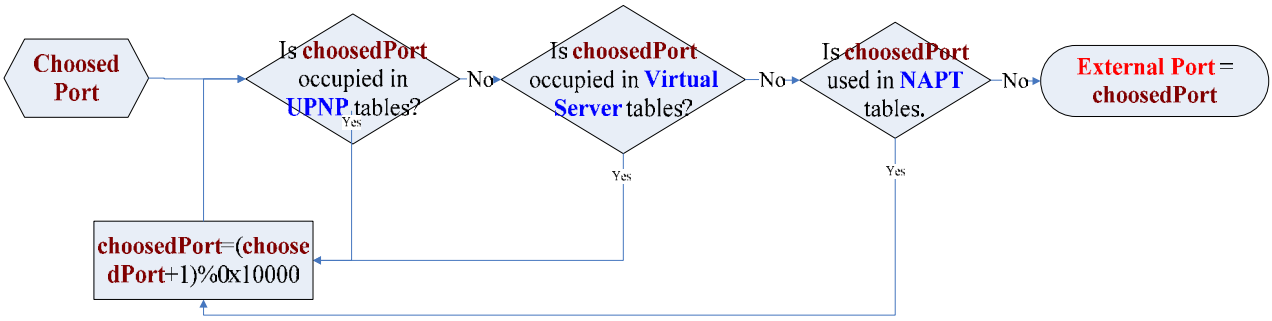


Figure7. NAPT External Port choosing flow.

5.1.3. API: rtk_rg_naptExtPortFree

This API is used to free the external port in database. After the free action, the external port can be used in the other flow connetion.

5.1.4. Sample Codes for Hooking NAPT External Port Collision Prevention Mechanism in Linux 2.6.30

Here is an example for hooking the NAPT External Port Collision Prevention Mechanism into Linux Kernal 2.6.30.

Before OS netfilter establish the conntrack, the user should call *rtk_rg_naptExtPortGet* for letting RomeDriver check and maintained all external ports. User should hack Protocal-Stack sample code as following:

In Linux 2.6.30: linux-2.6.x/net/ipv4/netfilter/nf_nat_core.c

```
EXPORT_SYMBOL(nf_nat_used_tuple);

#if defined(CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION)
extern int nf_conntrack_drop(const struct nf_conn *ct);
void nf_nat_conntrack_replace(const struct nf_conntrack_tuple *tuple,
                              const struct nf_conn *ignored_conntrack)
        {
                struct net *net = nf_ct_net(ignored_conntrack);
                struct nf_conntrack_tuple_hash *h;
                struct hlist_nulls_node *n;
                unsigned int hash;
                struct nf_conn * ct=NULL;
                struct nf_conn * tmpct=NULL;
                unsigned long longest_expires=0;
                unsigned long begin_jiffies=jiffies;
                int i;
```

```
                rcu_read_lock_bh();
                for (i = 0; i < nf_conntrack_htable_size; i++)
                {

                        /* Disable BHs the entire time since we need to disable them at
                          * least once for the stats anyway.
                          */
                        hlist_nulls_for_each_entry_rcu(h, n, &net->ct.hash[i], hnnode)
                        {
                                ct = nf_ct_tuplehash_to_ctrack(h);
                                if(ct && (ct != ignored_conntrack))
                                {
                                        if (!test_bit(IPS_ASSURED_BIT, &ct->status))
                                        {
                                                tmpct = ((ct->timeout.expires-(jiffies-begin_jiffies)) >
longest_expires)?ct:tmpct;
                                                longest_expires = tmpct->timeout.expires;
                                        }
                                }
                        }
                }
                rcu_read_unlock_bh();
                if(tmpct)
                {
                        printk("Early drop connection: protocol:%d SRC[%d.%d.%d.%d:%d]
GW[%d.%d.%d.%d:%d]DST[%d.%d.%d.%d:%d]\n",tmpct->tuplehash[IP_CT_DIR_ORIGINAL].tuple.dst.protonum
                        nf_conntrack_drop(tmpct);
                }

                return 0;
        }

EXPORT_SYMBOL(nf_nat_conntrack_replace);
#endif

/* If we source map this tuple so reply looks like reply_tuple, will
  * that meet the constraints of range. */
static int
in_range(const struct nf_conntrack_tuple *tuple,
             const struct nf_nat_range *range)

……

static void
get_unique_tuple(struct nf_conntrack_tuple *tuple,
                 const struct nf_conntrack_tuple *orig_tuple,
                 const struct nf_nat_range *range,
                 struct nf_conn *ct,
                 enum nf_nat_manip_type maniptype)
{
        ……
out:
#if defined(CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION)
        if((maniptype == IP_NAT_MANIP_SRC)          //SNAT
                && ((orig_tuple->dst.protonum==IPPROTO_TCP)
                    ||(orig_tuple->dst.protonum==IPPROTO_UDP)))
        {
                extern int32 rtk_rg_naptExtPortGet(int isTcp,uint16 *pPort);
                extern int32 rtk_rg_naptExtPortFree(int isTcp,uint16 port);
```

```
                    /* Keep Linux NAT core and RG forwarding engine using separated NAPT external port. */
                    int i=0;
                    unsigned short *pPort;
                    int isTCP;
                    int ret;

                    isTCP = (tuple->dst.protonum==IPPROTO_TCP)?1:0;
                    pPort = &tuple->src.u.all;

                    while(pPort)
                    {
                            if(i++>65535) break;
                            ret = rtk_rg_naptExtPortGet(isTCP,pPort);
                            if(ret==0) //Port is available
                            {
                                    if ((nf_nat_used_tuple(tuple, ct))==0)
                                            break;
                            }
                            else if(ret==-1) //NAPT flow full
                            {
                                    printk("PS used napt flow full\n");
                                    nf_nat_conntrack_replace(tuple, ct);
                                    break;
                            }
                            else
                            {
                                    *pPort=((*pPort+1)&0xffff);
                            }
                    }
            }
            else //DNAT  {
              //Do nothing!
        }
#endif
        rcu_read_unlock();
}
```

Table4. Example: hook rtk_rg_naptExtPortGet

After OS delete the conntrack, user should call rtk_rg_naptExtPortFree to free the especial external port, and following shows the sample code:

In Linux 2.6.30: linux-2.6.x/net/netfilter/nf_conntrack_core.c

```
static void
destroy_conntrack(struct nf_conntrack *nfct)
{
        ……
        rcu_read_lock();
        l4proto = __nf_ct_l4proto_find(nf_ct_l3num(ct), nf_ct_protonum(ct));
        if (l4proto && l4proto->destroy)
                l4proto->destroy(ct);
        rcu_read_unlock();

#if defined(CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION)
```

```
	extern unsigned int rtk_rg_naptExtPortFree(int isTcp,unsigned short port);
	if(nf_ct_l3num(ct)==PF_INET)
	{
		if (test_bit(IPS_SEEN_REPLY_BIT, &ct->status))
		{
			if(nf_ct_protonum(ct)==IPPROTO_TCP)
				rtk_rg_naptExtPortFree(1,ct->tuplehash[IP_CT_DIR_REPLY].tuple.src.u.all);
			else if(nf_ct_protonum(ct)==IPPROTO_UDP)
				rtk_rg_naptExtPortFree(0,ct->tuplehash[IP_CT_DIR_REPLY].tuple.src.u.all);
		}
	}
#endif
	……
	nf_conntrack_free(ct);
}

……

EXPORT_SYMBOL_GPL(nf_conntrack_tuple_taken);

#if defined(CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION)
int nf_conntrack_drop(const struct nf_conn *ct)
{
		del_timer(&ct->timeout);
		death_by_timeout((unsigned long)ct);

		return 0;
}
EXPORT_SYMBOL_GPL(nf_conntrack_drop);
#endif

#define NF_CT_EVICTION_RANGE	8
```

Table5. Example: hook rtk_rg_naptExtPortFree

6. (Lite)RomeDriver Initiation

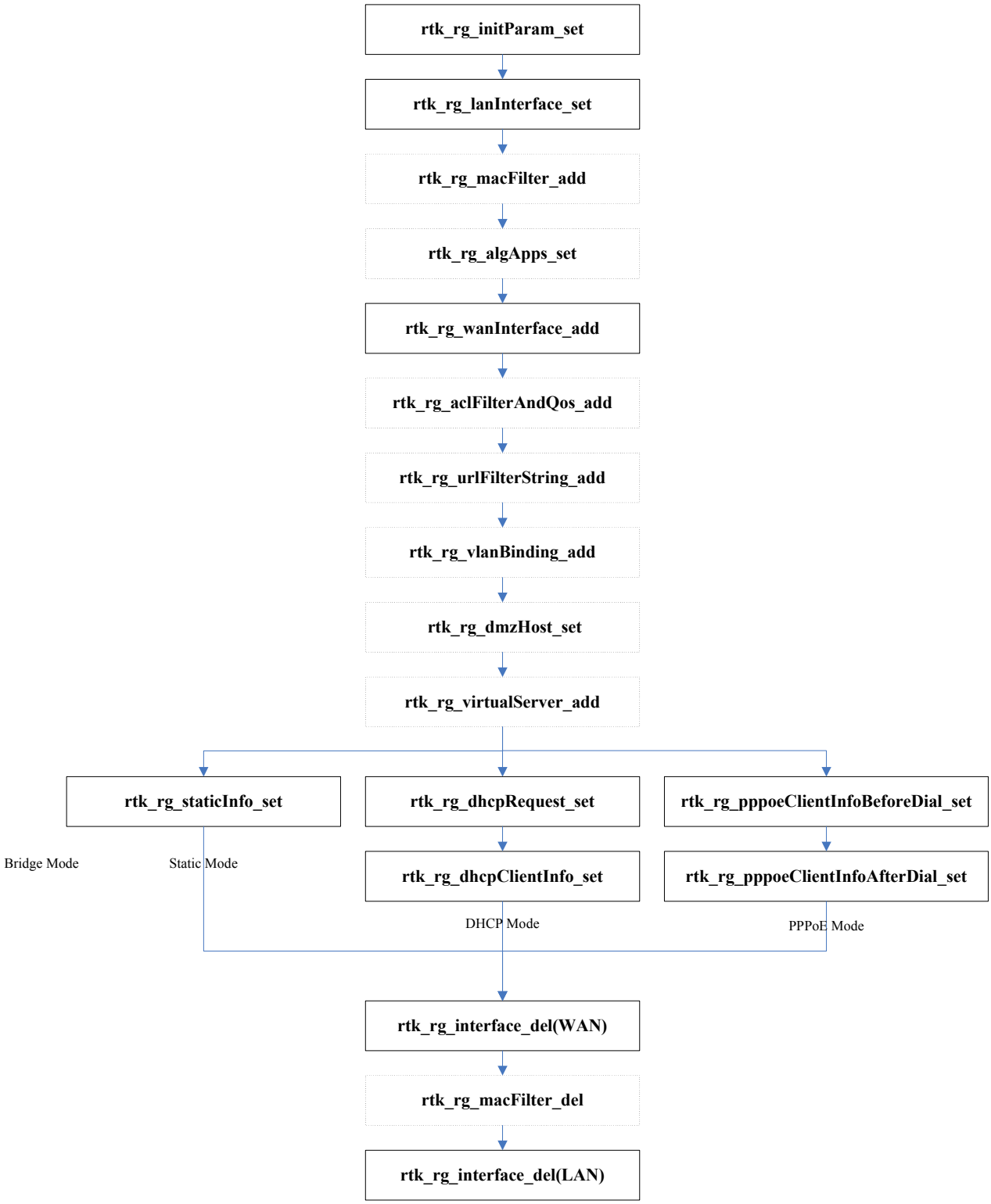


Figure8. RomeDriver Initiation Flow Chart

The above figure shows the RomeDriver and Lite RomeDriver initial flow of WAN Type is Bridge

WAN, Static WAN, DHCP WAN or PPPoE WAN, respectively. The dotted line is for setting advance feature. For basic setting, 3~4 steps is needed for Bridge WAN/Router/Gateway configuration.

1. rtk_rg_initParam_set
2. rtk_rg_lanInterface_add
3. rtk_rg_wanInterface_add
4. rtk_rg_staticInfo_set (for L3/L4)

The following description shows the Gateway basic setting process.

6.1. Gateway Basic Conifguration

There are 4 steps for setting up basic conversion by RTK RG APIs:

Step1-initialize RG Driver parameters:

rtk_rg_initParam_set is used to initial the Driver, and set up basic parameter and callback functions. Following code shows the default setting:

```
if(rtk_rg_initParam_set(NULL)) return -1;
```

Table6. Example: rtk_rg_initParam_set (Default)

If user wants to control the initialization, the parameter *init_param.igmpSnoopingEnable* is use to enable/disable the Igmpsnooping, the parameter *init_param.macBasedTagDecision* is use to enable/disable the MacBasedVlan feature, and *init_param. wanPortGponMode* is use to configurate the Wan port as PON feature or not.. The rest parameter is use to register callback functions, which implement by user, to each configuration step. The callback functions usually are used to notify user for Hardware add/modify/delete informations and synchronize the information between Hardware and Protocol-Stack.

If user wants to modify the parameter should first get the default parameter and set back after modified.

Following code shows how to enable Igmpsnooping and disable MacBasedVlan and register a callback function in init parameters for notifing user that the hardware wants to add ARP Entry.

```
int *arpAddByHw_CallBack(rtk_rg_arpEntry_t *arp) { ... }
...
rtk_rg_initParams_t init_param;
if(rtk_rg_initParam_get(&init_param)) return -1;
init_param.igmpSnoopingEnable=ENABLED;
init_param. macBasedTagDecision= DISABLED;
```

```
init_param.arpAddByHwCallBack=arpAddByHw_CallBack;
if(rtk_rg_initParam_set(&init_param)) return -1;
```

Table7. Example: rtk_rg_initParam_set (Customise)

Following description explain the usage of parameter *init_param.macBasedTagDecision*:
This parameter use to control the Hardware determind VLAN by MAC_BASED or VLAN_BASED. Here is the configuration rules that shoud be followed either in MAC_BASED setting or VLAN_BASED setting :
If *init_param.macBasedTagDecision*=DISABLED (VLAN_BASED):
The Hardware DMAC_TO_CVID will be disabled, so the vlan/port binding can not be set while calling *rtk_rg_wanInterface_add*. And if the added Interfcace, which bring up by *rtk_rg_wanInterface_add*., is BridgeWan, the VLAN must same as Lan Interface.
If *init_param.macBasedTagDecision*=ENABLED (MAC_BASED)*:*
The Hardware DMAC_TO_CVID will be enabled,.However, while calling API *rtk_rg_lanInterface_add/ rtk_rg_wanInterface_add/ rtk_rg_cvlan_add*, the parameter *isIVL* can only be set to SVL.

Step2-Add LAN Interface:

The API rtk_rg_lanInterface_add is used to add LAN Interface. There are two parameters, the first parameter rtk_rg_lanIntfConf_t *lan_info is used to send LAN information to Driver, the second one int *intf_idx is the interface index returns from Driver. Since LAN interface index and WAN interface index are two separate sets of data, therefore, LAN/WAN can use the same API rtk_rg_interface_del as deleting interface.

Following code shows how to set up ipv4/ipv6 lan interface, and add Port0~3 and SSID0~3 into LAN Interface. LAN configuration: VID=9, GMAC=00:11:22:33:44:55, LAN Gateway IP/MASK=192.168.1.1/255.255.255.0 , IPv6/MASK_LENGH=2001::1/64，MTU=1500.

```
rtk_rg_lanIntfConf_t lan_info;
int lanIdx;
memset(&lan_info,0,sizeof(lan_info));
lan_info.ip_version=IPVER_V4V6;
lan_info.gmac.octet[0]=0x00;
lan_info.gmac.octet[1]=0x11;
lan_info.gmac.octet[2]=0x22;
lan_info.gmac.octet[3]=0x33;
lan_info.gmac.octet[4]=0x44;
lan_info.gmac.octet[5]=0x55;
lan_info.intf_vlan_id=9;
```

```
lan_info.vlan_based_pri_enable=RTK_RG_DISABLED;
lan_info.vlan_based_pri=0;
lan_info.ip_addr=0xc0a80101; //192.168.1.1
lan_info.ip_network_mask=0xffffff00;
lan_info.ipv6_addr.ipv6_addr[0]=0x20;
lan_info.ipv6_addr.ipv6_addr[1]=0x01;
lan_info.ipv6_addr.ipv6_addr[15]=0x01;
lan_info.ipv6_network_mask_length =64;
lan_info.mtu=1500;
lan_info.isIVL=0; //0:SVL 1:IVL
lan_info.port_mask.portmask= ((1<<RTK_RG_PORT0)|(1<<RTK_RG_PORT1)|
      (1<<RTK_RG_PORT2)|(1<<RTK_RG_PORT3) | (1<<RTK_RG_EXT_PORT0)|
      (1<<RTK_RG_EXT_PORT1)| (1<<RTK_RG_EXT_PORT2) |
      (1<<RTK_RG_EXT_PORT3));
lan_info.untag_mask.portmask=((1<<RTK_RG_MAC_PORT0)|(1<<RTK_RG_MAC_PORT1)|
  (1<<RTK_RG_MAC_PORT2)|(1<<RTK_RG_MAC_PORT_CPU)); //untagset hasn't ext port
if(rtk_rg_lanInterface_add(&lanIntf,&lanIdx)) return -1;
```

Table8. Example: rtk_rg_lanInterface_add

lan_info. ip_version is used to decide whether to enabling ipv4, or ipv6 respectively, or to enabling both. by setting ip_version to ipv4 or ipv6 respectively, it means port and protocol based VLAN decision is setup, the different protocol from the same interface will be treat differently. For example, if the interface set ip_version to 0, means ipv4_only, ipv6 packets from this interface will be blocked. If ip_version set to 2, we will set up port based VLAN decision, that means untag packets will follow the interface's VLAN filtering setting. Besides, no matter *lan_info. ip_version* is IPVER_V6ONLY or IPVER_V4V6, only one IPv6 lan interface can be supported because the hardware limitation.

lan_info.gmac must be unicast address, if you entering a multicast here, the INVALID_PARAM error will be returned.

lan_info. intf_vlan_id indicate the interfcae VLAN, but the value can not be the same as any existing VLAN-Binding or 1Q-VLAN setting.

lan_info.vlan_based_pri_enable is to decide enable VLAN_BASED priority or not. This may effect the internal priority decision. User have to assign the priority value to *lan_info.vlan_based_pri* if enabled *lan_info.vlan_based_pri_enable.*

lan_info.port_mask(data type: rtk_rg_port_idx_t) and *lan_info.untag_mask*(data type: rtk_rg_mac_port_idx_t) are different. *lan_info.port_mask* include Virtual Extension Port, but *lan_info.untag_mask* serves as indication for whether the Physical Port should add tag or not.

lan_info.isIVL is used to decide the behavior of L2-table learning. If user initialized in MAC_BASED VLAN, *then lan_info.isIVL* can only set to SVL. When add LAN interface, it can

not have exact same IP subnet with other added interfaces, otherwise it will return fail.
When add LAN or WAN IPv4 subnet, we reserved C class subnet for each side. For example, if you add a LAN interface with network mask 255.255.255.0, the ARP entries of this subnet will remain in hardware table for L34 forwarding. After that, if you add another LAN interface, no matter how big the subnet is, it's ARP entries will be added to software ARP table and their L34 packets have to forward by fwdEngine. Identically, the policy is same with WAN interfaces. This is compromise for hardware ARP table has 512 entries in total, so we need software ARP mechanism to make up for it.
If the IP address and MAC address are all zero, then this LAN interface will not be added to routing and netif hardware table. It just influences the LAN port's port-based or port-and-protocol-based VLAN settings, if it is the first added interface for that port.

Step3-Add WAN Interface:

rtk_rg_wanInterface_add is an API used to add WAN Interface. Just like rtk_rg_lanInterface_add. There are two parameters, the first one rtk_rg_wanIntfConf_t *wanintf is used to send WAN information to Driver, the second one int *intf_idx is the interface index returning from Driver. Following code shows how to assign Port3 to WAN Interface, and WAN configuration: VID=8, GMAC=00:11:22:33:44:56, without assigned VLAN_BASED priority and egress packet without VLAN Tag, WAN Type is STATIC and default gateway. (only one WAN Interface can be Default gateway).
When add wan_type as RTK_RG_BRIDGE and the egress_vlan_tag_on is set as zero, the port-based VLAN setting of WAN port will be assigned as this WAN's VLAN ID. Otherwise, for other wan_type, they will not infect WAN port's port-based VLAN setting. Only one WAN interface can be set as untag bridge WAN in the system. Besides, if user initialized in MAC_BASED VLAN, the API will add LAN/WAN side VLAN member to each other. But if initialize in VLAN_BASED VLAN, user have to configurate the BRIDGE WAN VLAN as same as LAN VLAN.

```
rtk_rg_wanIntfConf_t wan_info;
int wanIdx;
memset(&wan_info,0,sizeof(wan_info));
wan_info.egress_vlan_id=8;
wan_info.vlan_based_pri_enable= RTK_RG_DISABLED;
wan_info. vlan_based_pri=0;
wan_info.egress_vlan_pri=0;
wan_info.egress_vlan_tag_on=0;
wan_info.gmac.octet[0]=0x00;
wan_info.gmac.octet[1]=0x11;
wan_info.gmac.octet[2]=0x22;
wan_info.gmac.octet[3]=0x33;
wan_info.gmac.octet[4]=0x44;
```

```
wan_info.gmac.octet[5]=0x56;
wan_info.port_binding_mask.portmask=0;
wan_info.wan_port_idx= RTK_RG_MAC_PORT_PON;
wan_info.wan_type= RTK_RG_STATIC;
wan_info.isIVL=0; //0:SVL 1:IVL
if(rtk_rg_wanInterface_add(&wan_info,&wanIdx)) return -1;
```

Table9. Example: rtk_rg_wanInterface_add

If user initialized in VLAN_BASED VLAN , *the wan_info.port_binding_mask.portmask* is limited to 0x0. In the other side, if user initialized in MAC_BASED VLAN , the *wan_info.isIVL* is limited to SVL.

Step4-Add WAN Interface information, such as IP Address/Network Mask/Gateway IP…etc:

If WAN Type is Static, user needs to use rtk_rg_staticInfo_set to set up WAN IP information. When the napt_enable on rtk_rg_ipStaticInfo_t is disabled, the Router just forward Routing Packter when NAPT action is disabled.

When ip_network_mask=0xffffffff is on, this WAN will not be considered as Interface Route. Like LAN interface, WAN interfaces share 256 hardware ARP entries at all. If they need more, we will add their ARP to software table.

gw_mac_auto_learn=1 is not supported in Lite RomeDriver. Thus, the WAN Gateway MAC Address should be assigned by user. And also, WAN Gateway MAC should never be multicast address.

```
rtk_rg_ipStaticInfo_t staticInfo;
memset(&staticInfo,0,sizeof(staticInfo));
staticInfo. ip_version=2;
staticInfo. ipv4_default_gateway_on =1;
staticInfo. gateway_ipv4_addr =0xc0a89675; //wan gateway ip:192.168.150.117
staticInfo.ip_addr=0xc0a89674; //wan ip:192.168.150.116
staticInfo.ip_network_mask=0xfffffffe;
staticInfo. ipv6_default_gateway_on=1;
staticInfo. gateway_ipv6_addr. ipv6_addr[0]=0x20; //ipv6 gateway 2002::2
staticInfo. gateway_ipv6_addr. ipv6_addr[1]=0x02;
staticInfo. gateway_ipv6_addr. ipv6_addr[15]=0x02;
staticInfo. ipv6_addr. ipv6_addr[0]=0x20; //ipv6 2002::1
staticInfo. ipv6_addr. ipv6_addr[0]=0x02;
staticInfo. ipv6_addr. ipv6_addr[0]=0x01;
staticInfo. ipv6_mask_length=64;
staticInfo.mtu=1500;
```

```
staticInfo.napt_enable=1;
staticInfo gw_mac_auto_learn_for_ipv4 =0;
staticInfo .gw_mac_auto_learn_for_ipv6=0
staticInfo.gateway_mac_addr.octet[0]=0x00;
staticInfo.gateway_mac_addr.octet[1]=0x10;
staticInfo.gateway_mac_addr.octet[2]=0x20;
staticInfo.gateway_mac_addr.octet[3]=0x30;
staticInfo.gateway_mac_addr.octet[4]=0x40;
staticInfo.gateway_mac_addr.octet[5]=0x60;
if(rtk_rg_staticInfo_set(wanIdx,&staticInfo)) return -1;
```

Table10. Example: rtk_rg_staticInfo_set(for LiteRomeDriver ARP set by USER)

Forwarding Engine can process ARP Request for getting WAN Gateway MAC in RomeDriver. If user hasn't assigned WAN Gateway MAC, then the value of gw_mac_auto_learn should be 1.

```
rtk_rg_ipStaticInfo_t staticInfo;
memset(&staticInfo,0,sizeof(staticInfo));
staticInfo. ipv4_default_gateway_on =1;
staticInfo.gateway_ip_addr=0xc0a89675; //wan ip:192.168.150.117
staticInfo.ip_addr=0xc0a89674; //wan ip:192.168.150.116
staticInfo.ip_network_mask=0xfffffffe;
staticInfo.mtu=1500;
staticInfo.napt_enable=1;
staticInfo. gw_mac_auto_learn_for_ipv4=1;
staticInfo. ipv6_default_gateway_on=1;
staticInfo. ipv6_addr. ipv6_addr[0]=0x20; //ipv6 2002::1
staticInfo. ipv6_addr. ipv6_addr[0]=0x02;
staticInfo. ipv6_addr. ipv6_addr[0]=0x01;
staticInfo. ipv6_mask_length=64;
staticInfo .gw_mac_auto_learn_for_ipv6=1;



if(rtk_rg_staticInfo_set(wanIdx,&staticInfo)) return -1;
```

Table11. Example: rtk_rg_staticInfo_set(for RomeDriver ARP auto learning)

Step5- Add Gateway MAC:

For LiteRomeDriver users without L2/ARP Autolearning Module, L2/ARP/NAPT should be configured by following APIs. If L2/ARP Autolearning Module is enabled in LiteRomeDriver, only NAPT should be configured. However, RomeDriver users can ignore all following APIs because its

Auto Learning Mechanism for adding all these entries.

Following code shows adding MAC Entry into L2 table by rtk_rg_macEntry_add:

```
rtk_rg_macEntry_t l2;
int l2Idx=0;
memset(&l2,0,sizeof(rtk_rg_macEntry_t));
l2.mac.octet[0]=0x0;
l2.mac.octet[1]=0x10;
l2.mac.octet[2]=0x20;
l2.mac.octet[3]=0x30;
l2.mac.octet[4]=0x40;
l2.mac.octet[5]=0x60;
l2.fid=0;
l2.isIVL=1;
l2.port_idx=RTK_RG_PORT1;
l2.vlan_id=9;
if(rtk_rg_macEntry_add(&l2,&l2Idx)) return -1;
```

Table12. Example: rtk_rg_macEntry_add

Step6-Add Gateway ARP:

After obtaining, l2Idx from rtk_rg_macEntry_add, user should set IP information & l2Idx to ARP table by rtk_rg_arpEntry_add.

```
rtk_rg_arpInfo_t arpInfo;
int arpIdx=0;
memset(&arpInfo,0,sizeof(rtk_rg_arpInfo_t));
arpInfo.arpEntry.ipv4Addr=0xc0a80102; //192.168.1.2
arpInfo.arpEntry.macEntryIdx = l2Idx;
arpInfo.arpEntry.staticEntry = 1;
if(rtk_rg_arpEntry_add(&arpInfo.arpEntry,&arpIdx)) return -1;
```

Table13. Example: rtk_rg_arpEntry_add

Step8-Add NAPT

Use rtk_rg_naptEntry_t napt to configure NAPT connection information and set this configuration to NAPT table by rtk_rg_naptConnection_add.

```
rtk_rg_naptEntry_t napt;
int naptIdx=0;
memset(&napt,0,sizeof(rtk_rg_naptEntry_t));
napt.is_tcp = 1;
```

```
napt.local_ip = 0xc0a80102; //192.168.1.2
napt.local_port = 0x500;
napt.remote_ip = 0xc0a89675;
napt.remote_port = 0x706;
napt.wan_intf_idx = 1;
napt.external_port = 0x555;
if(rtk_rg_naptConnection_add(&napt,&naptIdx)) return -1;
```

Table14. Example: rtk_rg_naptConnection_add

6.2. Gateway Advanced Conifguration

6.2.1. Port Binding

Port Binding is assigned as soon as WAN Interface initialization is established. Each Port can only be assigned to Port Binding PortMask on one WAN Interface. If one Port assigned to WAN Interface twice, the second WAN Interfcae will fail on initialization. Besides, this feature only enabled if init_param.macBasedTagDecision=ENABLED while calling *rtk_rg_initParam_set*. Following code shows how to support Port Binding by fix Step3 Sample code.

```
rtk_rg_wanIntfConf_t wanIntf;
int wanIdx;
memset(&wanIntf,0,sizeof(rtk_rg_wan_Intf_Conf_t));
wanIntf.egress_vlan_id=8;
wanIntf.egress_vlan_pri=0;
wanIntf.egress_vlan_tag_on=0; /* egress c-vlan untag */
wanIntf.gmac.octet[0]=0x00; /* set WAN GMAC=00:11:22:33:44:56 */
wanIntf.gmac.octet[1]=0x11;
wanIntf.gmac.octet[2]=0x22;
wanIntf.gmac.octet[3]=0x33;
wanIntf.gmac.octet[4]=0x44;
wanIntf.gmac.octet[5]=0x56;
wanIntf.wan_port_idx=RTK_RG_MAC_PORT3;
wanIntf.default_gateway_on=1; /* default router to this WAN interface */
wanIntf.port_binding_mask.portmask=(1<< RTK_RG_PORT0)|(1<< RTK_RG_PORT1);
wanIntf.wan_type= RTK_RG_STATIC; /* WAN Type: STATIC */
if(rtk_rg_wanInterface_add(&wanIntf,&wanIdx)) return -1;
```

Table15. Example: rtk_rg_wanInterface_add (with Port Binding)

After setting above Port Binding configuration, Port0 & Port1 will use this WAN Interface as default route, no matter what DIP or what Ingress VLAN Tag VID take from packet, this default routing will transform packet VID as WAN Interface VID.

6.2.2. VLAN Binding

VLAN Binding is used to binding specific VLAN Tag packet on the LAN and assign Wan Interface Egress packet VLAN Tag which will follow WAN Interface setting to decide Tag or not, and Egress VID. In the meantime, Ingress packet VLAN Tag from WAN assigned by LAN will be transformed to original format.

```
rtk_rg_vlanBinding_t vlanBind;
memset(&vlanBind,0,sizeof(rtk_rg_vlanBinding_t));
vlanBind.port_idx=RTK_RG_MAC_PORT0;
vlanBind.ingress_vid=100;
vlanBind.wan_intf_idx=wanIdx;
if(rtk_rg_vlanBinding_add(&vlanBind,&wanIdx)) return -1;
```

Table16. Example: rtk_rg_vlanBinding_add

After VLAN Binding configuration is set, no matter what DIP is, the Host with CTAG VID=100 from Port0 will use this WAN Interface as Default Route, and transform VID to WAN Interface VID. This feature only enabled if user set init_param.macBasedTagDecision=ENABLED while calling *rtk_rg_initParam_set.*
This example is to show how to forward packet from Port0 and CTAG=100 to Port PON, and determind to take VLAN Tag=8 or not by wlanIntf.egress_vlan_tag_on. When packet is sent form WAN side to LAN side specific host, the CTAG will be transformed to VID=100 by MAC and sent to Port0.

6.2.3. ACL Filter / ACL-Based QoS Remarking

rtk_rg_aclFilterAndQos_add can be used to filter under particular ingress/egress condition, it also can trap packet to CPU conducting or do multi-taskings such as: Logging action and do Qos action(CVLAN Priority remarking, DSCP Remarking, IP Precedence Remarking, assign Queue ID, and assign Share Meter Index). User can configure *action_type, and qos_actions field and fwding_type_and_direction* in aclRule to perform different activities.

action_type field supports following activities:

ACL_ACTION_TYPE_DROP	Drop any packet hit the rule by Hardware. If there is no set

	rule on Permit action, the Default Action Policy is Permit.
ACL_ACTION_TYPE_PERMIT	Hardware will keep the packet of Hit Rule, and drop all other packets. When a Permit Rule is added, the Default Action Policy will turn to Drop.
ACL_ACTION_TYPE_TRAP	Trap Hit Rule packet to CPU Port. And the packet will be handeled by Forwarding Engine
ACL_ACTION_TYPE_QOS	All QoS action can be distributed to perform 1P Remarking, IP Precedence Remarking, DSCP Remarking, assign Queue ID, assign Share Meter, assign streamed, assign acl-priority. qos_actions is relevant only when action_type is ACL_ACTION_TYPE_QOS
ACL_ACTION_TYPE_TRAP_TO_PS	Trap Hit Rule packet to CPU Port. But the packet will skip Forwarding Engine process, and be sent to Protocal Stack in Kernal directly.

Table17. List of ACL Action Type

qos_actions field support following activities:

ACL_ACTION_1P_REMARKING_BIT	Transformed CVLAN Prioriy, the value of Priority(0~7) should be assigned in dot1p_remarking_pri field.
ACL_ACTION_IP_PRECEDENCE_REMARKING_BIT	Transformed IP Precedence, the value of IP Precedence (0~7) should be assigned in ip_precedence_remarking_pri.
ACL_ACTION_DSCP_REMARKING_BIT	Transformed DSCP, the value of DSCP (0~63) should be assigned in dscp_remarking_pri.
ACL_ACTION_QUEUE_ID_BIT	Assigned packet to specific Queue
ACL_ACTION_SHARE_METER_BIT	Assigned Share Meter Index.
ACL_ACTION_STREAM_ID_OR_LLID_BIT	Assigned GPON StreanID or EPON LLID
ACL_ACTION_ACL_PRIORITY_BIT	Assigned acl-priority, which may effect internal priority decision.

Table18. List of ACL QoS Actions

qos_actions can handle several actions in the same time by using "|"(OR) operation.

The filtedred conditions and filtered condition glossary are definded as follow:

Filter conditions bits	Related fields
INGRESS_PORT_BIT	ingress_port_mask、ingress_extport_mask

INGRESS_INTF_BIT	ingress_intf_idx
EGRESS_INTF_BIT	egress_intf_idx
INGRESS_ETHERTYPE_BIT	ingress_ethertype
INGRESS_CTAG_PRI_BIT	ingress_ctag_pri
INGRESS_CTAG_VID_BIT	ingress_ctag_vid
INGRESS_SMAC_BIT	ingress_smac
INGRESS_DMAC_BIT	ingress_dmac
INGRESS_DSCP_BIT	ingress_dscp
INGRESS_L4_TCP_BIT	None
INGRESS_L4_UDP_BIT	None
INGRESS_L4_ICMP_BIT	None
INGRESS_IPV6_SIP_RANGE_BIT	ingress_src_ipv6_addr_start、ingress_src_ipv6_addr_end
INGRESS_IPV6_DIP_RANGE_BIT	ingress_dest_ipv6_addr_start、ingress_dest_ipv6_addr_end
INGRESS_IPV4_SIP_RANGE_BIT	ingress_src_ipv4_addr_start、ingress_src_ipv4_addr_end
INGRESS_IPV4_DIP_RANGE_BIT	ingress_dest_ipv4_addr_start、ingress_dest_ipv4_addr_end
INGRESS_L4_SPORT_RANGE_BIT	ingress_src_l4_port_start、ingress_src_l4_port_end
INGRESS_L4_DPORT_RANGE_BIT	ingress_dest_l4_port_start、ingress_dest_l4_port_end
EGRESS_IPV4_SIP_RANGE_BIT	egress_src_ipv4_addr_start、egress_src_ipv4_addr_end
EGRESS_IPV4_DIP_RANGE_BIT	egress_dest_ipv4_addr_start、egress_dest_ipv4_addr_end
EGRESS_L4_SPORT_RANGE_BIT	egress_src_l4_port_start、egress_src_l4_port_end
EGRESS_L4_DPORT_RANGE_BIT	egress_dest_l4_port_start、egress_dest_l4_port_end
EGRESS_CTAG_PRI_BIT	egress_ctag_pri
EGRESS_CTAG_VID_BIT	egress_ctag_vid
INGRESS_IPV6_DSCP_BIT	ingress_ipv6_dscp

Table19. List of ACL Patterns

fwding_type_and_direction supports following activities:

ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET	This type can support all actions, however only ingress patterns can be supported.
ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_UP_DROP	This type is used as upstream filter for dropping any ingress/egress combination patterns packet.
ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_DOWN_DROP	This type is used as downstream filter for dropping any ingress/egress combination patterns packet, however egress_ctag_vid and egress_ctag_pri pattern are not supported.
ACL_FWD_TYPE_DIR_INGRESS_OR_EGRESS_L34_UP_STREAMID	This type is used for assigning upstream PON streamID and must assign egress_ctag_vid and egress_ctag_pri

	pattern.

Table20. List of ACL fwding type

Below is a typical senario of setting an ACL Qos Rule to filter all UDP packet in PPPoE, and transform DSCP to 0x20 and assigned QueueID to 1. The filtered conditions are distributed to perform INGRESS_ETHERTYPE(0x8864) and INGRESS_L4_UDP. The action_type is assigned to ACL_ACTION_TYPE_QOS, In the same time, enabled qos_actions bit of ACL_ACTION_DSCP_REMARKING_BIT & ACL_ACTION_QUEUE_ID_BIT.

```
rtk_rg_aclFilterAndQos_t aclRule;
int aclIdx;
memset(&aclRule,0,sizeof(rtk_rg_aclFilterAndQos_t);
aclRule.fwding_type_and_direction= ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET;
aclRule.filter_fields=INGRESS_ETHERTYPE_BIT | INGRESS_L4_UDP_BIT;
aclRule.ingress_ethertype=0x8864;
aclRule.action_type= ACL_ACTION_TYPE_QOS;
aclRule.qos_actions= ACL_ACTION_DSCP_REMARKING_BIT |
                     ACL_ACTION_QUEUE_ID_BIT;
aclRule.action_dscp_remarking_pri=0x20;
aclRule.action_queue_id=0x7;
if(rtk_rg_aclFilterAndQos_add(&aclRule,&aclIdx)) return -1;
```

Table21. Example: rtk_rg_aclFilterAndQos_add

6.2.4. Layer2 packet classify filter and assign streamID

rtk_rg_classifyEntry_add can be use to assign layer2 packet streamed or LLID in PON environment. It can also use to filter some egress patterns. User can configure *index, filter_fields , direction, us_action_field or ds_action_field* in classifyRule to perform different activities

The *index* should be in range of 64~511. The former the rule is, the higher priotiy the rule is. The classifyRule will be checked in sequence. Once a classifyRule is hit, the related action will be excute and skip checking the rest classifyRules.

The filter_fields conditions and filtered condition glossary are definded as follow:

Filter conditions bits	Related fields
EGRESS_ETHERTYPR_BIT	etherType, etherTypeMask
EGRESS_GEMIDX_BIT	gemidx, gemidxMask

EGRESS_LLID_BIT	llid, llidMask
EGRESS_TAGVID_BIT	outterTagVid, outterTagVidMask
EGRESS_TAGPRI_BIT	outterTagPri, outterTagPriMask
EGRESS_INTERNALPRI_BIT	internalPri, internalPriMask
EGRESS_STAGIF_BIT	stagIf, stagIfMask
EGRESS_CTAGIF_BIT	ctagIf, ctagIfMask
EGRESS_UNI_BIT	uni, uniMask

direction field support following activities:

RTK_RG_CLASSIFY_DIRECTION_UPSTREAM	actions should assign in *us_action_field*
RTK_RG_CLASSIFY_DIRECTION_DOWNSTREAM	actions should assign in *ds_action_field*

us_action_field support following actions:

CF_US_ACTION_SID_BIT	assign streamed by field *sid.*
CF_US_ACTION_DROP_BIT	Drop the packet that hit the classifyRule.

ds_action_field support following actions:

CF_DS_ACTION_UNI_FORCE_BIT	*uniPortMask* must be 0x0. Drop the packet that hit the classifyRule.

Below is a typical senario of setting an classifyRule to assign VLAN=100 and Dot1p priority=1 packet to streameId 10.

```
rtk_rg_classifyEntry_t classifyRule;
memset(&classifyRule,0,sizeof(rtk_rg_classifyEntry_t));
classifyRule.index=64;
classifyRule.direction = RTK_RG_CLASSIFY_DIRECTION_UPSTREAM;
classifyRule.filter_fields |= (EGRESS_TAGVID_BIT | EGRESS_TAGPRI_BIT);
classifyRule.outterTagVid =100;
classifyRule.outterTagVidMask=0xfff;
classifyRule.outterTagPri=1;
classifyRule.outterTagPriMask=0x7;
classifyRule.us_action_field = (CF_US_ACTION_SID_BIT);
classifyRule.sid=10;
if(rtk_rg_classifyEntry_add(&classifyRule)) return -1;
```

Table22. Example: rtk_rg_classifyEntry_add

6.2.5. Rate Limit

6.2.5.1. ACL-Based Share Meter

When user wants to limit the flow rate of specific packet, the requirement can be supported by the ACL QoS action type- ACL_ACTION_SHARE_METER_BIT providing both ACL Rule is and Share Meter bandwidth are configurated.
In order to do so, Share Meter bandwidth should be set by rtk_rg_shareMeter_set. There are three parameters in API;
The first parameter is the index of Share Meter(0~31). The second parameter is the max flow rate ,and the uint is 1Kbps, However, the actual Hardware unit is 8Kbps. If user fills in 15 to the second parameter, it will be rounded down to 8Kbps. So, the minimal value is liminted to 8. If user fills in 1048576 to this parameter which means unlimit flow rate. The third parameter represents inter-frame gap+frame preamble(total 20bytes) into the second parameter count.
However, Rate Limit in this way doesn't include BandWidth Control. The packets which exceeding the rate limitation will be dropped.

The following cod shows how to limit all UDP packet to 80Kbps.

```
rtk_rg_aclFilterAndQos_t aclRule;
int aclIdx
//set meter index 1 rate=80Kbps, and include inter-frame gap+frame preamble;
rtk_rg_shareMeter_set (1,80,ENABLED);
memset(&aclRule,0,sizeof(rtk_rg_aclFilterAndQos_t);
aclRule.fwding_type_and_direction= ACL_FWD_TYPE_DIR_INGRESS_ALL_PACKET;
aclRule.filter_fields= INGRESS_L4_UDP_BIT;
aclRule.action_type=ACL_ACTION_TYPE_QOS;
aclRule.qos_actions=ACL_ACTION_SHARE_METER_BIT;
aclRule.action_share_meter=1;
if(rtk_rg_aclFilterAndQos_add(&aclRule,&aclIdx)) return -1;
```

Table23. Example: rtk_rate_shareMeter_set

6.2.5.2. Egress Port -Based Rate Limit

rtk_rg_portEgrBandwidthCtrlRate_set set is used to set per-egress port limitation bandwidth. In case if the per-egress port limitation bandwith and per-egress port & queue share meter bandwidth

are all set for the same purpose in the same time, the smallest one will be refered as queue egress bandwidth. The parameter Port Number can only be set as MAC Port (RTK_RG_MAC_PORT0 ~ RTK_RG_MAC_PORT_CPU).

The following code shows how to limit the packet which egress set to Port0 to 80Kbps

```
if(rtk_rg_portEgrBandwidthCtrlRate_set (RTK_RG_MAC_PORT0,80)) return -1;
```

Table24. Example: rtk_rate_portEgrBandwidthCtrlRate_set

6.2.5.3. Ingress Port-Based Rate Limit

rtk_rg_portIgrBandwidthCtrlRate_set is used to set limitation bandwidth of ingress direction, the parameter is same as rtk_rate_portEgrBandwidthCtrlRate_set.

The following code shows how to limit the packet of ingress from Port0 to 80Kbps.

```
if(rtk_rg_portIgrBandwidthCtrlRate_set (RTK_RG_MAC_PORT0,80)) return -1;
```

Table25. Example: rtk_rate_portIgrBandwidthCtrlRate_set

6.2.6. QoS Internal Priority & Egress Queue Decision

Egress Queue is mapping from Internal Priority. Users must understand how to decide the internal priority if they want to determine the egress queue.
In Our HW, we support 9 kinds of priority source. Users are able to determine each weight of priority source by their own.

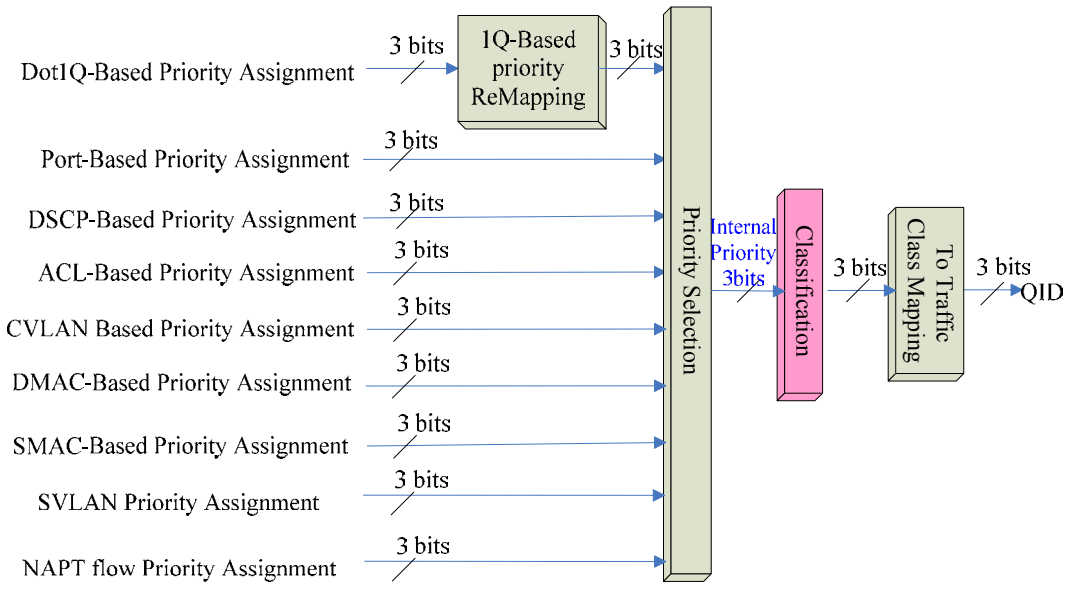


Figure9. Internal Priority decision flow chart

The following table show the corresponding RG APIs of each kind of internal priority source.

Dot1Q-Based Priority	rtk_rg_qosDot1pPriRemapToInternalPri_set
Port-Based Priority	rtk_rg_lanInterface_add rtk_rg_wanInterface_add rtk_rg_qosPortBasedPriority_set
DSCP-Based	rtk_rg_qosDscpRemarkByInternalPri_set
ACL-Based	rtk_rg_aclFilterAndQos_add
CVLAN-Based	rtk_rg_lanInterface_add rtk_rg_wanInterface_add rtk_rg_cvlan_add
DMAC-Based	X(Not support yet)
SMAC-Based	X(Not support yet)
SVLAN-Based	X(Not support yet)
NAPT-Based	rtk_rg_naptConnection_add

Table26. The corresponding RG APIs of each kind of priority source.

QoS Internal Priority Decision	rtk_rg_qosDot1pPriRemapToInternalPri_get	CVLAN-Priority remapping table get
	rtk_rg_qosDot1pPriRemapToInternalPri_set	CVLAN-Priority remapping table set
	rtk_rg_qosDscpRemapToInternalPri_get	DSCP remapping table get.
	rtk_rg_qosDscpRemapToInternalPri_set	DSCP remapping table set
	rtk_rg_qosInternalPriDecisionByWeight_get	Get the weight of the 9 kinds of priority decision source.
	rtk_rg_qosInternalPriDecisionByWeight_set	Set the weight of the 9 kinds of priority decision source.
	rtk_rg_qosPortBasedPriority_get	Get port based priority
	rtk_rg_qosPortBasedPriority_set	Set port based priority
QoS Egress Queue Decision	rtk_rg_qosInternalPriMapToQueueId_get	Get the “Internal priority to Queue” table.
	rtk_rg_qosInternalPriMapToQueueId_set	Set the “Internal priority to Queue” table.

Table27. The QoS internal priority & egress queue decision RG APIs.

The following sample code shows how to config the weight of priority source. If users’ priority decision sequence is ACL > DMAC > DSCP > NAPT > CVLAN > SVLAN > 1Q > PORT >

SMAC.

```
rtk_rg_qos_priSelWeight_t weight;
memset(&weight,0,sizeof(rtk_rg_qos_priSelWeight_t));
weight.weight_of_portBased=1;
weight.weight_of_dot1q=2;
weight.weight_of_dscp=12;
weight.weight_of_acl=15;
weight.weight_of_ lutFwd =13;
weight.weight_of_saBaed=0;
weight.weight_of_vlanBased=10;
weight.weight_of_svlanBased=9;
weight.weight_of_l4Based=11;
if(rtk_rg_qosInternalPriDecisionByWeight_set (weight)) return -1;
```

Table28. Example: rtk_rg_qosInternalPriDecisionByWeight_set

Harware has a C-Priority Remapping Table. When ingress packet has C-tag, the Dot1Q based priority is one-to-one mapping from ingress C-tag priority by this remapping table.
The following sample code shows how to remap C-Priority to Internal priority.

```
rtk_rg_qosDot1pPriRemapToInternalPri_set(0,0);
rtk_rg_qosDot1pPriRemapToInternalPri_set(1,0);
rtk_rg_qosDot1pPriRemapToInternalPri_set(2,1);
rtk_rg_qosDot1pPriRemapToInternalPri_set(3,1);
rtk_rg_qosDot1pPriRemapToInternalPri_set(4,2);
rtk_rg_qosDot1pPriRemapToInternalPri_set(5,2);
rtk_rg_qosDot1pPriRemapToInternalPri_set(6,3);
rtk_rg_qosDot1pPriRemapToInternalPri_set(7,3); //ingress c-pri 7 -->internal priority 3
```

Table29. Example: rtk_rg_qosDot1pPriRemapToInternalPri_set

Harware has a DSCP Remapping Table. When ingress packet has DSCP, the DSCP based priority is multple-to-one mapping from ingress DSCP priority by this remapping table.
The following sample code shows how to remap DSCP to Internal priority.

```
// config internal priority decison by chapter: 6.2.6
...

rtk_rg_qosDot1pPriRemapToInternalPri_set(0,0);
rtk_rg_qosDot1pPriRemapToInternalPri_set(1,0);
...
```

```
rtk_rg_qosDscpRemapToInternalPri_set(62,7);
rtk_rg_qosDscpRemapToInternalPri_set(63,7); // DSCP 63 --> internal priority 7
```

Table30. Example: rtk_rg_qosDscpRemapToInternalPri_set

Harware has a Internal-Priority to Egress-Queue Remapping Table. The Egress Queue is one-to-one remapping from internal priority.

The following sample code shows how to config this remap table.

```
// config internal priority decison by chapter: 6.2.6
...
rtk_rg_qosInternalPriMapToQueueId_set(0,0);
rtk_rg_qosInternalPriMapToQueueId_set(1,0);
rtk_rg_qosInternalPriMapToQueueId_set(2,1);
rtk_rg_qosInternalPriMapToQueueId_set(3,1);
rtk_rg_qosInternalPriMapToQueueId_set(4,2);
rtk_rg_qosInternalPriMapToQueueId_set(5,2);
rtk_rg_qosInternalPriMapToQueueId_set(6,3);
rtk_rg_qosInternalPriMapToQueueId_set(7,3); //internal priority 7 --> egress queue 3
```

Table31. Example: rtk_rg_qosInternalPriMapToQueueId_set

6.2.7. QoS 1P Remarking

The egress priority of 1P remarking is refer from internal priority.

QoS 1Q Remarking	rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get	CVLAN-Priority remarking egress port get.
	rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set	CVLAN-Priority remarking egress port set.
	rtk_rg_qosDot1pPriRemarkByInternalPri_get	CVLAN-Priority remarking table get
	rtk_rg_qosDot1pPriRemarkByInternalPri_set	CVLAN-Priority remarking table set

Table32. The QoS 1Q Reamrking RG APIs.

The following sample code shows how to remark the priority of egress vlan tag by internal priority.

```
// config internal priority decison by chapter: 6.2.6
...
rtk_rg_qosDot1pPriRemarkByInternalPri_set(0,0);
```

```
rtk_rg_qosDot1pPriRemarkByInternalPri_set(1,0);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(2,1);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(3,1);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(4,2);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(5,2);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(6,3);
rtk_rg_qosDot1pPriRemarkByInternalPri_set(7,3); //internal priority 7 --> egress C-pri 3
rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set(RTK_RG_MAC_PORT_PON,
RTK_RG_ENABLED); // Enable PON port egress 1P remarking.
```

Table33. Example: QoS 1P remarking

6.2.8. QoS DSCP Remarking

The egress DSCP is refer from internal priority or ingress DSCP. Users are able to choise one which they need.

QoS DSCP Remarking	rtk_rg_qosDscpRemarkByDscp_get	Get the “DSCP remarking by DSCP” mapping table.
	rtk_rg_qosDscpRemarkByDscp_set	Set the “DSCP remarking by DSCP” mapping table.
	rtk_rg_qosDscpRemarkByInternalPri_get	Get the “DSCP remarking by Internal Priority” mapping table.
	rtk_rg_qosDscpRemarkByInternalPri_set	Set the “DSCP remarking by Internal Priority” mapping table.
	rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get	Get DSCP remarking egress port.
	rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set	Set DSCP remarking egress port.

Table34. The QoS DSCP Remarking RG APIs.

The following sample code shows how to remark egress DSCP by internal priority.

```
// config internal priority decison by chapter: 6.2.6
...
rtk_rg_qosDscpRemarkByInternalPri_set(0,0);
rtk_rg_qosDscpRemarkByInternalPri_set(1,8);
rtk_rg_qosDscpRemarkByInternalPri_set(2,16);
rtk_rg_qosDscpRemarkByInternalPri_set(3,24);
```

```
rtk_rg_qosDscpRemarkByInternalPri_set(4,32);
rtk_rg_qosDscpRemarkByInternalPri_set(5,40);
rtk_rg_qosDscpRemarkByInternalPri_set(6,48);
rtk_rg_qosDscpRemarkByInternalPri_set(7,56); //internal priority 7 --> egress DSCP 56

rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set(
                RTK_RG_MAC_PORT_PON, RTK_RG_ENABLED,
                RTK_RG_DSCP_RMK_SRC_INT_PRI);
                // Enable PON port egress DSCP remarking by internal priority.
```

Table35. Example: QoS DSCP remarking

6.2.9. QoS Strict Priority / WFQ

The RG API: rtk_rg_qosStrictPriorityOrWeightFairQueue_set is used to achieve this kind of requirement. This function has two parameters. The first parameter is Port number(range of 0~6,0~3:LAN Port, 4: PON, 5:RGMII, 6:CPU). The second parameter is per-queue weight of related port. Weight equal 0 represents the Strict Priority. When weight falls in the range of 1~128 represents WFQ Weights. If there is more than one Strict Priority, the higher the queue number has the first priority to be handled. After all Strict Priority Queues are handled, the WFQ Queues then will be handled. Each Port has the capacity for 8 Queues.

To prioritize any specific packets into Strict Priority Queue or WFQ, can done from ACL rule by Setting QoS action type- ACL_ACTION_QUEUE_ID_BIT with assigned queue ID and assigning per Port & Queue.

Here is the scenery of setting code for Port4~7 on Strict Priority, while Port0~3 to WFQ. Consequently, packets transmit priority will put Queue7 packets as first priority to send out, and then following with the sequencial order of Queue6, Queue5, and Queue4. After all packets in Queue7~4 are exhausted, the Queue3~0 will start to send, and the bandwidth will be in order of 4:3:2:1. In ACL, all TCP packets are assigned to Queue7 with top priority.

```
rtk_rg_aclFilterAndQos_t aclRule;
int aclIdx,i;
rtk_rg_qos_queue_weights_t q_weight;
q_weight.weights[7]=0; //Queue4~7:Strict Priority
q_weight.weights[6]=0;
q_weight.weights[5]=0;
```

```
q_weight.weights[4]=0;
q_weight.weights[3]=4; //Queue3~0 4:3:2:1
q_weight.weights[2]=3;
q_weight.weights[1]=2;
q_weight.weights[0]=1;
for(i=0;i<7;i++)
{
    rtk_rg_qosStrictPriorityOrWeightFairQueue_set (i,&q_weight);
}
memset(&aclRule,0,sizeof(rtk_rg_aclFilterAndQos_t));
aclRule.filter_fields=INGRESS_L4_TCP_BIT;
aclRule.action_type=ACL_ACTION_TYPE_QOS;
aclRule.qos_actions=ACL_ACTION_QUEUE_ID_BIT;
aclRule.action_queue_id=0x7;
if(rtk_rg_aclFilterAndQos_add(&aclRule,&aclIdx)) return -1;
```

Table36. Example: rtk_qos_schedulingQueue_set

6.2.9.1. DSCP to Internal Priority

If user wants to limit the qos of DHCP, not only above-mentioned (Sec. QoS Strict Priority / WFQ) configuration needs to set, but also the Internal priority decision & the mapping of dhcp-to-internal priority needs to set.

For Example, let dscp10: dscp20: dscp30: dscp40 sending ratio to 1:2:3:4 , we have to use *rtk_rg_qosInternalPriDecisionByWeight_set* to make the DSCP priority higher while Internal-Priority decision.(DSCP weight=12, just lower than ACL and lutFwd). And then using *rtk_rg_qosDscpRemapToInternalPri_set* to set dscp10 remapping to Priority0, dscp20 remapping to Priority1, dscp30 remapping to Priority2, and dscp40 remapping to Priority3

```
int ret;
rtk_rg_qos_priSelWeight_t weight;
memset(&weight,0,sizeof(rtk_rg_qos_priSelWeight_t));
weight.weight_of_portBased=1;
weight.weight_of_dot1q=2;
weight.weight_of_dscp=12;
weight.weight_of_acl=15;
weight.weight_of_ lutFwd =13;
weight.weight_of_saBaed=0;
weight.weight_of_vlanBased=10;
weight.weight_of_svlanBased=9;
weight.weight_of_l4Based=11;
```

```
if(rtk_rg_qosInternalPriDecisionByWeight_set (weight)) return -1;

if(rtk_rg_qosDscpRemapToInternalPri_set (10,0)) return -1;
if(rtk_rg_qosDscpRemapToInternalPri_set (20,1)) return -1;
if(rtk_rg_qosDscpRemapToInternalPri_set (30,2)) return -1;
if(rtk_rg_qosDscpRemapToInternalPri_set (40,3)) return -1;
```

Table37. Example: rtk_rg_qosDscpRemapToInternalPri_set

6.2.9.2. Port to Internal Priority

If user wants to limit the qos of each Port, not only above-mentioned (Sec. QoS Strict Priority / WFQ) configuration needs to set, but also the Internal priority decision & the mapping of port-to-internal priority needs to set.

For Example, let packets of Port1: Port2: Port3: Port4 sending ratio to 1:2:3:4 , we have to use *rtk_rg_qosInternalPriDecisionByWeight_set* to make the port-based priority higher while Internal-Priority decision.(port-based priority just lower than ACL and lutFwd). And then using *rtk_rg_qosPortBasedPriority_set* to set Port0 mapping to Priority0, Port 2 mapping to Priority1, Port 3 mapping to Priority2, and Port 4 remapping to Priority3.

```
int ret;
rtk_rg_qos_priSelWeight_t weight;
memset(&weight,0,sizeof(rtk_rg_qos_priSelWeight_t));
weight.weight_of_portBased=12;
weight.weight_of_dot1q=2;
weight.weight_of_dscp=0;
weight.weight_of_acl=15;
weight.weight_of_ lutFwd =13;
weight.weight_of_saBaed=0;
weight.weight_of_vlanBased=10;
weight.weight_of_svlanBased=9;
weight.weight_of_l4Based=11;
if(rtk_rg_qosInternalPriDecisionByWeight_set (weight)) return -1;

if(rtk_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT0,0)) return -1;
if(rtk_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT1,1)) return -1;
if(rtk_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT2,2)) return -1;
if(rtk_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT3,3)) return -1;
```

Table38. Example: rtk_rg_qosPortBasedPriority_set

6.2.10. L2 per-Port learning count limit

There are 2048 entries available on L2 Table without limitation on auto-learning volume in per-Port in default setting. If user wants to limit the auto-learning on per port should use rtk_l2_portLimitLearningCnt_set. The first parameter is PortNumber, and the second parameter is auto learning volume. If user enter 0 on the second parameter which will disable the auto learning mechanism to that specific Port, and the maximum number to fill in the second parameter should be 2048.

```
if(rtk_l2_portLimitLearningCnt_set(RTK_RG_MAC_PORT0,10)) return -1;
```

Table39. Example: rtk_l2_portLimitLearningCnt_set

6.2.11. MacFilter

For saving ACL resources, Macfilter is provided by L2 Tables currently. The following example shows how to drop packet with source mac 00:e0:4c:86:70:99 by rtk_rg_macFilter_add.

```
int macfilterIdx;
rtk_rg_macFilterEntry_t macFilterEntry;
memset(&macFilterEntry,0,sizeof(rtk_rg_macFilterEntry_t));
macFilterEntry.mac.octet[0]=0x00;
macFilterEntry.mac.octet[1]=0xe0;
macFilterEntry.mac.octet[2]=0x4c;
macFilterEntry.mac.octet[3]=0x86;
macFilterEntry.mac.octet[4]=0x70;
macFilterEntry.mac.octet[5]=0x99;
macFilterEntry.direct=RTK_RG_MACFILTER_FILTER_SRC_MAC_ONLY;
if(rtk_rg_macFilter_add(&macFilterEntry,&macfilterIdx))
    return -1;
```

Table40. Example: rtk_rg_macFilter_add

6.2.12. Application-Level-Gateway(ALG)

Up to now, PPTP/L2TP/FTP are supported in ALG. The following example shows how to enable FTP by rtk_rg_virtualServer_add.

```
rtk_rg_alg_type_t alg_app;
alg_app = RTK_RG_ALG_FTP_TCP_BIT | RTK_RG_ALG_FTP_UDP_BIT;
if(rtk_rg_algApps_set(alg_app))
```

```
    return -1;
```

Table41. Example: rtk_rg_algApps_set

6.2.13. Virtual Server

The following Example shows how to transmit tcp packets from Wan Interface, which wanIdx is 1, and destination port is 1000~1099 to Lan Interfcae Server which IP is 192.168.1.2 and mapping destination port to 2000~2099 by rtk_rg_virtualServer_add.

```
int virtualServerIdx;
rtk_rg_virtualServer_t virtual_server;
memset(&virtual_server,0,sizeof(rtk_rg_virtualServer_t));
virtual_server.valid=ENABLED;
virtual_server.is_tcp=ENABLED;
virtual_server.wan_intf_idx=1;
virtual_server.gateway_port_start = 1000;
virtual_server.local_port_start = 2000;
virtual_server.mappingPortRangeCnt = 100;
virtual_server.local_ip=0xc0a80102;     //192.168.1.2
if(rtk_rg_virtualServer_add(&virtual_server,&virtualServerIdx))
    return -1;
```

Table42. Example: rtk_rg_virtualServer_add

6.2.14. DmzHost

The following example shows how to enable dmzHost, and transmit all packets from Wan Interface, which wanIdx is 1, to the Lan Interfcae Host which IP is 192.168.1.3 by rtk_rg_dmzHost_set.

```
int wan_intf_idx;
rtk_rg_dmzInfo_t dmz_info;
memset(&dmz_info,0,sizeof(rtk_rg_dmzInfo_t));
wan_intf_idx = 1;
dmz_info.enabled=ENABLED;
dmz_info.private_ip=0xc0a80103; //192.168.1.3
if(rtk_rg_dmzHost_set(wan_intf_idx,&dmz_info))
    return -1;
```

Table43. Example: rtk_rg_dmzHost_set

6.2.15. UpnpConnection

The following example shows how to transmit tcp packets from Wan Interface, which wanIdx is 1, destination IP is 192.168.10.4, source port is 3000, and destination port is 3000 to Lan Interfcae Host whoch IP is 192.168.1.4, and Port is 4000. If upnp.limit_remote_ip=DISABLED means don't care source IP of originally packet. Similarly, upnp.limit_remote_port = DISABLED means don't care source port of originally packet.

```
int upnpIdx;
rtk_rg_upnpConnection_t upnp;
memset(&upnp,0,sizeof(rtk_rg_upnpConnection_t));
upnp.valid=ENABLED;
upnp.is_tcp=ENABLED;
upnp.wan_intf_idx=1;
upnp.gateway_port=3000;
upnp.local_ip=0xc0a80104; //192.168.1.4
upnp.local_port=4000;
upnp.remote_ip=0xc0a80a04;
upnp.remote_port=5000;
upnp.limit_remote_ip = 100; //192.168.10.4
upnp.limit_remote_port = 100;
upnp.type=UPNP_TYPE_PERSIST;
upnp.timeout=0; //0:disable auto-delete
if(rtk_rg_upnpConnection_add(&upnp,&upnpIdx))
    return -1;
```

Table44. Example: rtk_rg_upnpConnection_add

6.2.16. MulticastFlow

The following example shows how to let port0, port1 join an multicast flow which IP is 224.1.2.3 by rtk_rg_multicastFlow_add.

```
int flowIdx;
rtk_rg_multicastFlow_t mcFlow;
memset(&mcFlow,0,sizeof(rtk_rg_multicastFlow_t));
mcFlow.multicast_ipv4_addr = 0xe0010203; //224.1.2.3
mcFlow.isIPv6 = DISABLED;
mcFlow.port_mask.portmask = (1<<RTK_RG_MAC_PORT0)|(1<<RTK_RG_MAC_PORT1);
if(rtk_rg_multicastFlow_add(&mcFlow,&flowIdx))
    return -1;
```

Table45. Example: rtk_rg_multicastFlow_add

6.2.17. StormControl

StormControl can limit packet rate by sharemeter. We can use rtk_rg_shareMeter_set for setting sharemeters, and the parameters can reference to Sec.ACL-Based Share Mete. The following example shows how to use sharemeter[1] to limit multicast packets from port0 in 80Kbps by rtk_rg_stormControl_add.

```
int stormInfoIdx;
rtk_rg_stormControlInfo_t stormInfo;

if(rtk_rg_shareMeter_set(1, 80000, RTK_RG_DISABLED))
	return -1;

memset(&stormInfo,0,sizeof(rtk_rg_stormControlInfo_t));
stormInfo.valid=ENABLED;
stormInfo.port=RTK_RG_MAC_PORT0;
stormInfo.stormType=RTK_RG_STORM_TYPE_MULTICAST;
stormInfo.meterIdx=1;
if(rtk_rg_stormControl_add(&stormInfo,&stormInfoIdx))
	return -1;
```

Table46. Example: rtk_rg_stormControl_add

6.2.18. PortMirror

The following example shows how to set port2 as monitor port, and copy all received packets from port0 and port1 to port2 by rtk_rg_portMirror_set. Attention! We suggest the host in monitor port should not respond or redirect any packet that it received. Sometimes the response or redirection will make the packet direct looping in gateway until the ttl(time-to-live) decrease to 0.

```
rtk_rg_portMirrorInfo_t portMirrorInfo;
memset(&portMirrorInfo,0,sizeof(rtk_rg_portMirrorInfo_t));
portMirrorInfo.monitorPort=RTK_RG_MAC_PORT2;
portMirrorInfo.enabledPortMask.portmask=(1<<RTK_RG_MAC_PORT0)|
                                        (1<<RTK_RG_MAC_PORT1);
portMirrorInfo.direct=RTK_RG_MIRROR_RX_ONLY;
if(rtk_rg_portMirror_set(portMirrorInfo))
```

```
    return -1;
```

Table47. Example: rtk_rg_portMirror_set

6.2.19. Physical Port Ability

rtk_rg_phyPortForceAbility_set is used to force control the behavior of physical port. If ability.force_disable_phy set to ENABLED that can force link-down physical port. If ability.force_disable_phy set to DISABLED, and ability.valid set to ENABLED, then we can control the link speed, half-duplex/full-duplex, and flowControl or not. In the other side, if ability.valid set to DISABLED, the physical port will do auto-negotiation automatically. Following example shows how to force set physical port to link-speed 100M, full-duplex, and enable flowControl.

```
rtk_rg_mac_port_idx_t port;
rtk_rg_phyPortAbilityInfo_t ability;
memset(&ability,0,sizeof(rtk_rg_phyPortAbilityInfo_t));
port = RTK_RG_MAC_PORT0;
ability.force_disable_phy=DISABLED;
ability.valid=ENABLED;
ability.speed=RTK_RG_PORT_SPEED_100M;
ability.duplex=RTK_RG_PORT_FULL_DUPLEX;
ability.flowCtrl=RTK_RG_ENABLED;
if(rtk_rg_phyPortForceAbility_set(port,ability))
    return -1;
```

Table48. Example: rtk_rg_phyPortForceAbility_set

6.2.20. Denial-of-Service (DoS)

rtk_rg_dosPortMaskEnable_set is used to enable the DoS feature for each port. After that, we can use rtk_rg_dosType_set to determind trap or drop packets of different deny types. Or, we can use rtk_rg_dosFloodType_set to limit the packet rate of different flood types. The unit of threshold is (1K packets/sec). The packets exceed the threshold will br dropped.

```
rtk_rg_mac_portmask_t dos_port_mask;
dos_port_mask.portmask = (1<<RTK_RG_MAC_PORT0)|(1<<RTK_RG_MAC_PORT1)
|(1<<RTK_RG_MAC_PORT2)|(1<<RTK_RG_MAC_PORT3);
if(rtk_rg_dosPortMaskEnable_set(dos_port_mask))
    return -1;
```

Table49. Example: rtk_rg_dosPortMaskEnable_set

The following example shows how to trap all land deny packets to CPU port by rtk_rg_dosType_set.

```
if(rtk_rg_dosType_set(RTK_RG_DOS_LAND_DENY,ENABLED,RTK_RG_DOS_ACTION_TRAP))
    return -1;
```

Table50. Example: rtk_rg_dosType_set

The following example shows how to limit syn flood packets in 1K/sec.

```
if(rtk_rg_dosFloodType_set(RTK_RG_DOS_SYNFLOOD_DENY,ENABLED,
RTK_RG_DOS_ACTION_DROP,1))
    return -1;
```

Table51. Example: rtk_rg_dosFloodType_set

7. Wireless LAN Hardware Forward

To enable the Hardware Offload Forwarding (bridging/Routing/NAPT) capability with Wireless LAN, user should connect Extension Port and Switch. Extension Ports is a Virtual Port, which used to correspond with WLAN and SSID (ex. ROOT Interface and VAP Interface). There are 5 Extension Ports supported by Hardware Asic, and all of them trap packets to CPU Port with a specific identified ID for Hardware/Software.

CPU Tag is needed in order for Switch to identify Extension Port. However, packet form OS Protocol Stack doesn't have CPU Tag, and doesn't have enough Header Buffery to copy CPU Tag. Even if the Header Buffery can support the need sufficiently, some Memory Copy action, filling CPU Tag between MAC Address and Payload, can not be avoided. Solving this problem will take NIC TX/RX Descriptor to have enough capacity holding the CPU tag information. During TX procedure, TX Descriptor content would be transformed to CPU Tag format so that Hardware can identify. During RX procedure, Hardware would parsing CPU Tag from packet to RX Descriptor content, then remove CPU Tag, following with DMA the packet to memory to complete the task.

OWN=1	EOR	FS	LS	IPCS	L4CS	KEEP	BLU	CRC	vSEl	DISLRn	Cputag_ipcs	Cputag_l4cs	Cputag_psel	RSVD(1bit)	Data_Length (17 bits)	Offset 0
TX_BUFFER_ADDRESS (32 bits)																Offset 4

CPU Tag	aspri	Cputag_PRI (3 bits)	Tx VLAN action (2 bits)	Tx PPPoE action (2 bits)	Tx PPPoE Idx (3 bits)	efid	Enhance_fid [2:0]	VLAN_TAG	Offset 8
								VIDL / PRIO / CFI / VIDH	
Source Extension Port(3 bits) (extspa)	Tx destination port number (6 bits) (Tx portmask)			Tx destination stream ID (7 bits) (PON sream ID)				Reserved(13bits) / L34KEEP / PTP	Offset 12
LGSEN	Large-Send MTU value (11 bits)					RSVD (20 bits)			Offset 16

Figure10. NIC TX Descriptor Format

Above figure show NIC TX Descriptor fields. The fields relate to Extension Port are OWN, EOR, FS, LS, Cputag_ipcs, Cputag_l4cs, Data Length, CPU Tag, Tx PPPoE action, Tx PPPoE Idx, Source Extension Port, Tx destination port number, L34KEEP. There are options for the packet to sent form CPU Port to Destination Port directly((Direct TX) or forward by Switch(Hardware Lookup).

Using Hardway Lookup must fill followimg fields: OWN=1, EOR, FS=1, LS=1, Data Length, CPU Tag=1, Source Extension Port. On the other hand, using Direct TX must fill following fields: OWN=1, EOR, FS=1, LS=1, Cputag_ipcs=1, Cputag_l4cs=1, Data Length, CPU Tag=1, Tx PPPoE action, Tx PPPoE Idx, Source Extension Port, Tx destination port number=CPU decision Port, and L34KEEP=1.

OWN	EOR	FS	LS	CRCErr	IPV4CSF	L4CSF	RCDF	IP FRAG	PPPoE tag	RWT	PktType (4bits)	L3 routing	Orig Format	PCTRL	RSVD (2bits)	Data_Length (12bits) (max: 0x800)	Offset 0
RxBuff (32 bits)																	Offset 4
Cputag	PTP in CPU Tag exist	SVLAN Tag exist		RSVD (2bits)		PON stream id(7bits)		RSVD (3bits)		CTAVA	CVLAN_TAG (16 bits)						Offset 8
Source Port Number (5 bits)		Destination port mask (6 bits) (extension port mask)			Reason (8 bits)			Internal priority (3 bits)		Extenstion port TTL-1 (5 bits)		RSVD (5 bits)					Offset 12

Figure11. NIC RX Descriptor Format

Upon receipt of a packet from Switch in NIC, the CPU Tag information will be parsed and port to NIC RX Descriptor fields as above figure shown, After that CPU Tag will be removed. The Software can acquire informations from Switch by RX Descriptor fields: L3 Routing, Orig Format, Cputag, Source Port Number, Destination port mask, Reason, Internal Priority, Extension Port TTL-1.

L3 Routing	It means the packet has been modified by Hardware Asic, and Software doesn't need do any thing about the packet.

Orig Format	The packet is the original format (doesn't modify by Hareware).
Cputag	The RX Descriptor includes parsed CPU Tag information.
Source Port Number	The Port Number which the packet is originated from.
Destination port mask	The Ports that the is about to be ported to (include CPU Port & EXT0~4).
Reason	It is a normal forwarding function if 0 is the value of Reason . On the other hand, if the value is not zero, it means that user needs to trap CPU because the Switch can not handle the packet, and requires assistance from Software.
Internal Priority	Decision of NIC RX Ring ID (the value is decided by Internal Priority in Switch)
Extension Port TTL-1	While Multicasting and WAN_SA =1 in LUT, Hardward will determine whethere to include such Ext Port in TTL-1 list if the egress Ext Port across different VLAN. This field is valid only when Orig Format =1.

Table52. NIC RX Descriptor for CPU Tag

Once the NIC, CPU Port and Switch are connected, the Wireless LAN packets can be forwarded by Switch (Hardware Forwarding). The architechure is as following figure:

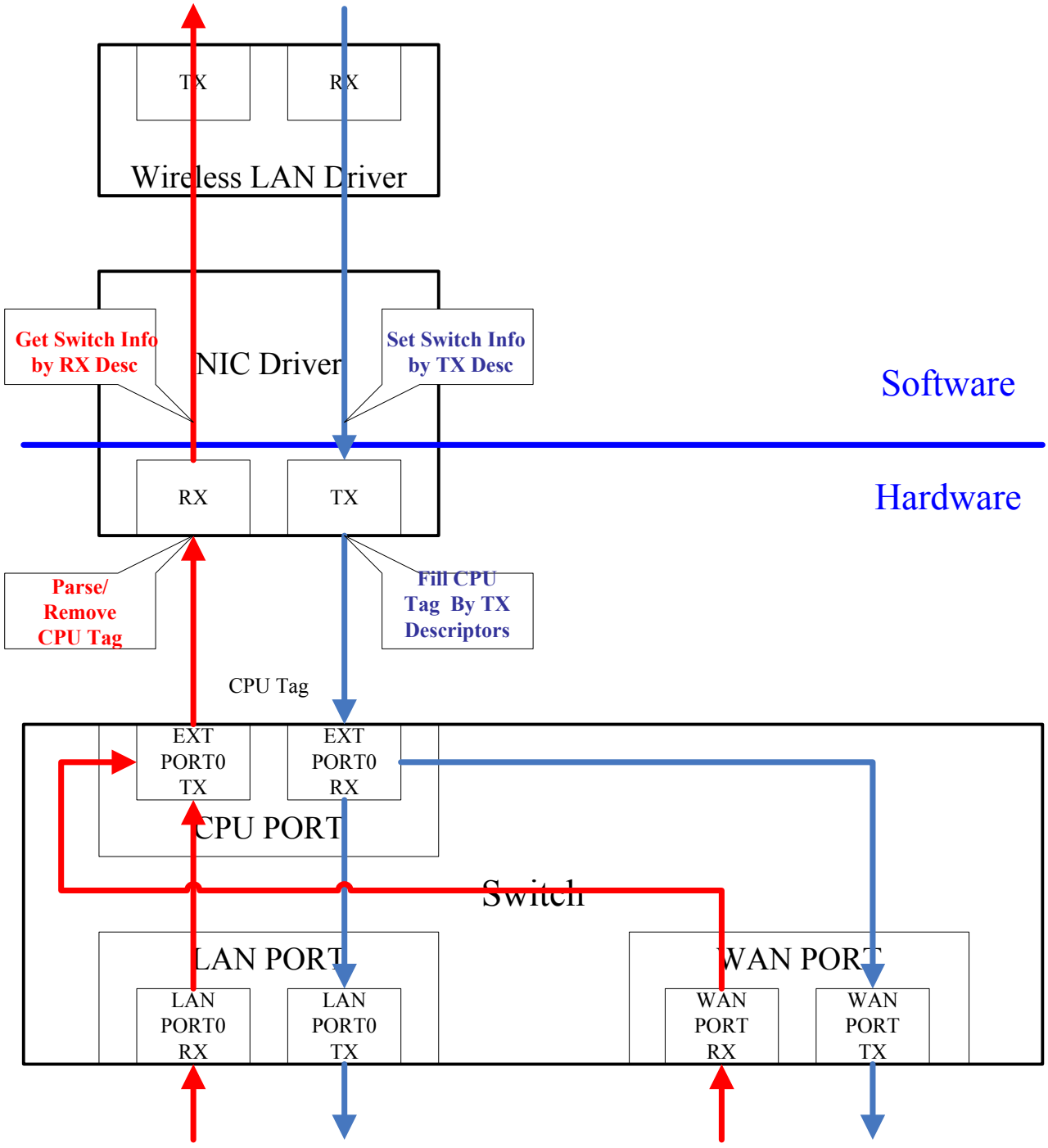


Figure12. Wireless LAN Hardware Forward

8. Test by Web UI configuration

the machine is turned on , the machine will take the last UI confirguration from Flash, and Linux Kernal will automatically set the Network Interface by the configuration. It will sequentially call RTK RG APIs, rtk_rg_initParam_set, rtk_rg_lanInterface_add, rtk_rg_wanInterface_add, and rtk_rg_staticInfo_set to enable NAPT capability in Hardware.

8.1. Configure LAN by Web UI

Use the Browser to connect Web UI and LAN Host (default LAN Gatwway IP is 192.168.1.1).
Select the LAN item: fill the IP Address and Subnet Mask.
Select [Apply Changes] to reflash the setting and write the configuration to MIB Flash.

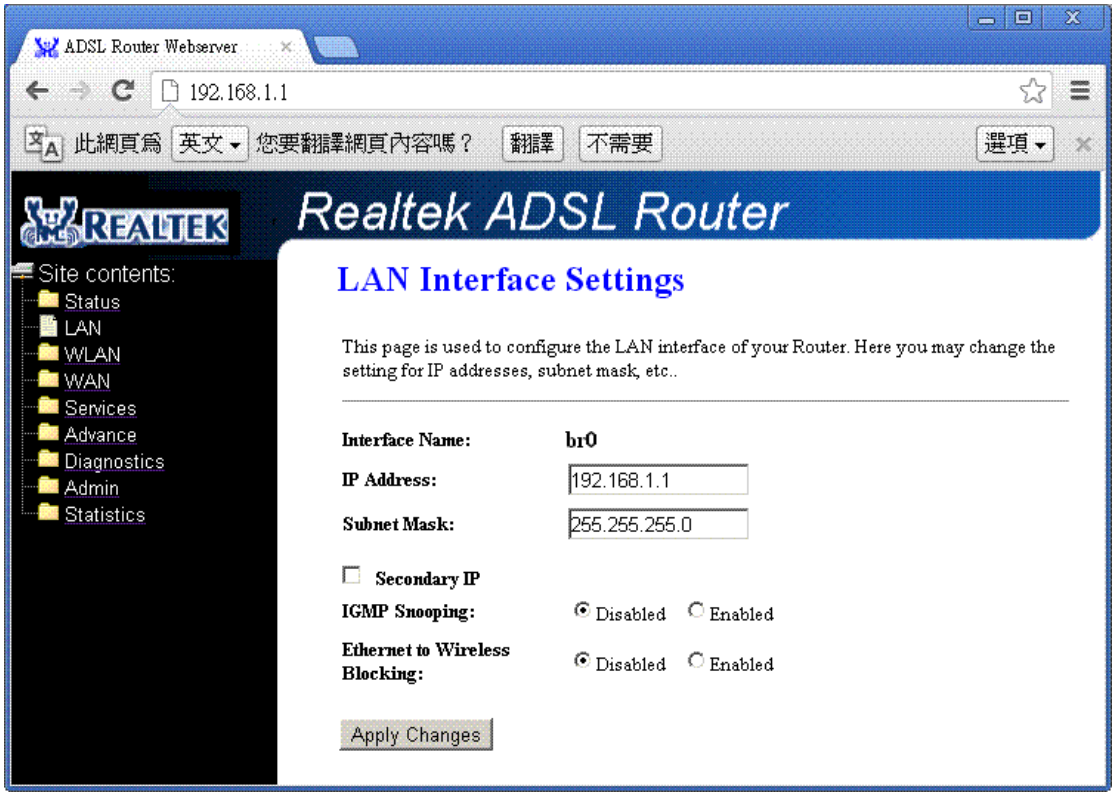


Figure13. Web UI:LAN Interface Setting

The MAC Address and Wan Physical Port can not be modified on Web UI. What can be done is to modity the MAC address and Wan Physical Port on MIB Flash, use following command, and then reboot the system.

```
# flash set ELAN_MAC_ADDR 00E04C867001     <assign Lan gateway mac 00:E0:4C:86:70:01:>
# flash set WAN_PHY_PORT 4     <assign PON Port as Wan Physical Port >
```

8.2. Configure WAN by Web UI

Select WAN item: fill Channel Mode=IPoE, Enable NAPT, IP Protocol=IPv4, Type=Fixed IP, Local IP Address: 192.168.150.116, Remote IP Address=192.168.150.117, Subnet Mask=255.255.255.0. Select [Apply Changes] to reflash the setting and write the configuration on MIB Flash.

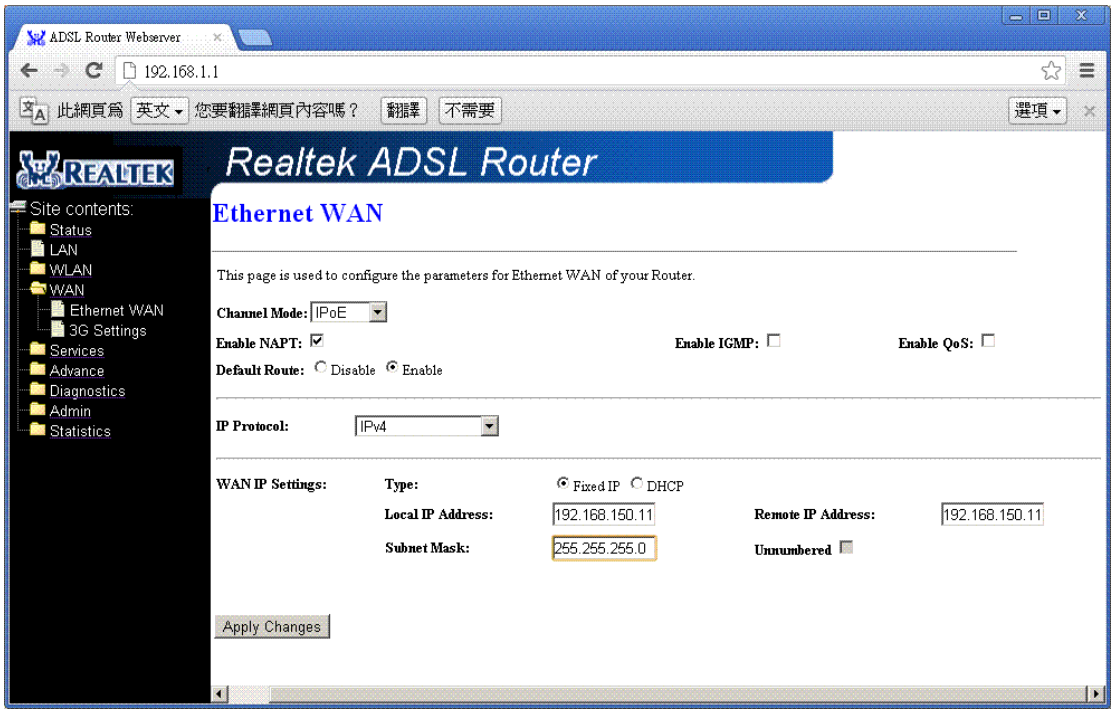


Figure14. Web UI:Ethernet WAN Interface Setting

9. Test by Diagnostics Shell Command

9.1. Diag Shell Introduction

DiagShell is an application in RG RomeDriver for debug and access RG APIs in run time. It provides an interface to typing diag commands to access RG APIs dynamically. Each RG API can mapping to 1~3 diag commands depends to the RG API complexity, and the relationship between RG API and diag command is introduced in *Sec. Mapping for RG API & Diag Cmd*

9.1.1. Startup DiagShell

Turn on the sytem, and log in with account: **admin** and password: **system**.

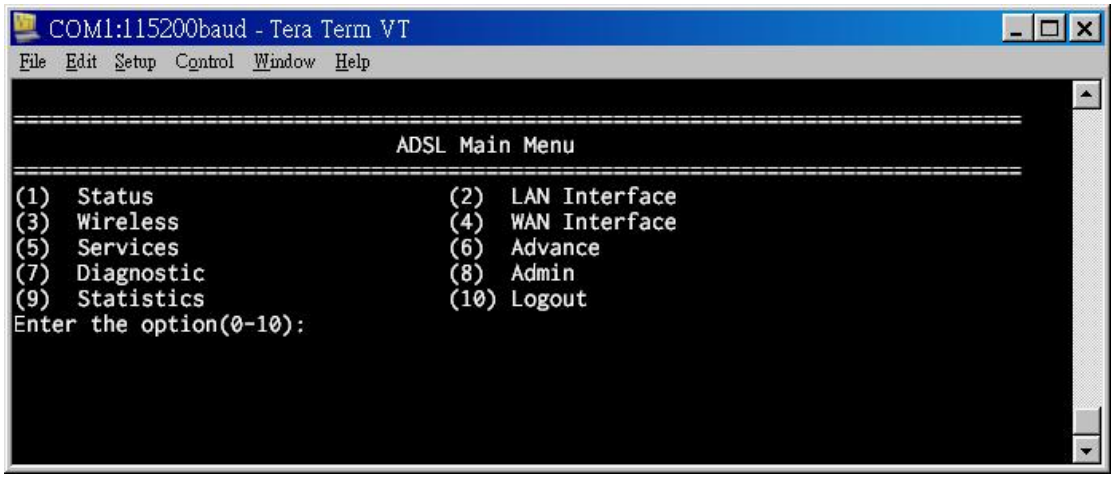


Figure15. UART: ADSL Main Menu

After login, enter the option **0** to skip ADSL Main Menu.
Enter **diag** for startup diagshell application.

9.1.2. Command-Line-Completion for Diag Cmd

DiagShell provides Command-Line-Completion capability by [Tab].

1. If current keyword is complete, it will give tips for next possible keyword.

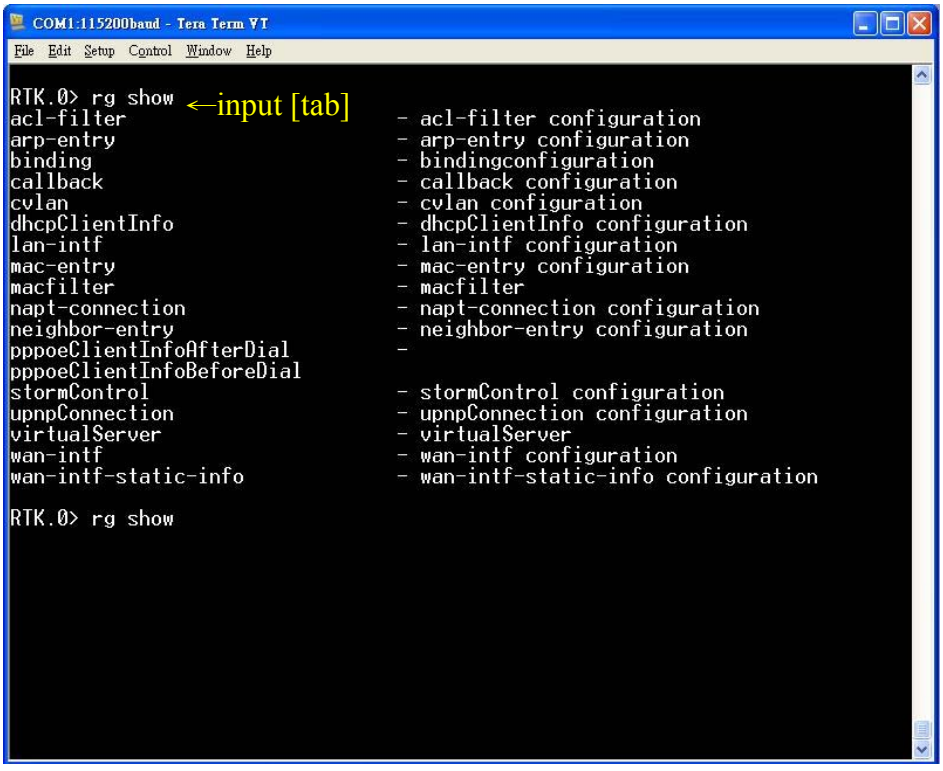


Figure16. DiagShell tips for next keyword

2. If current keyword is incomplete, it will complete current keyword:

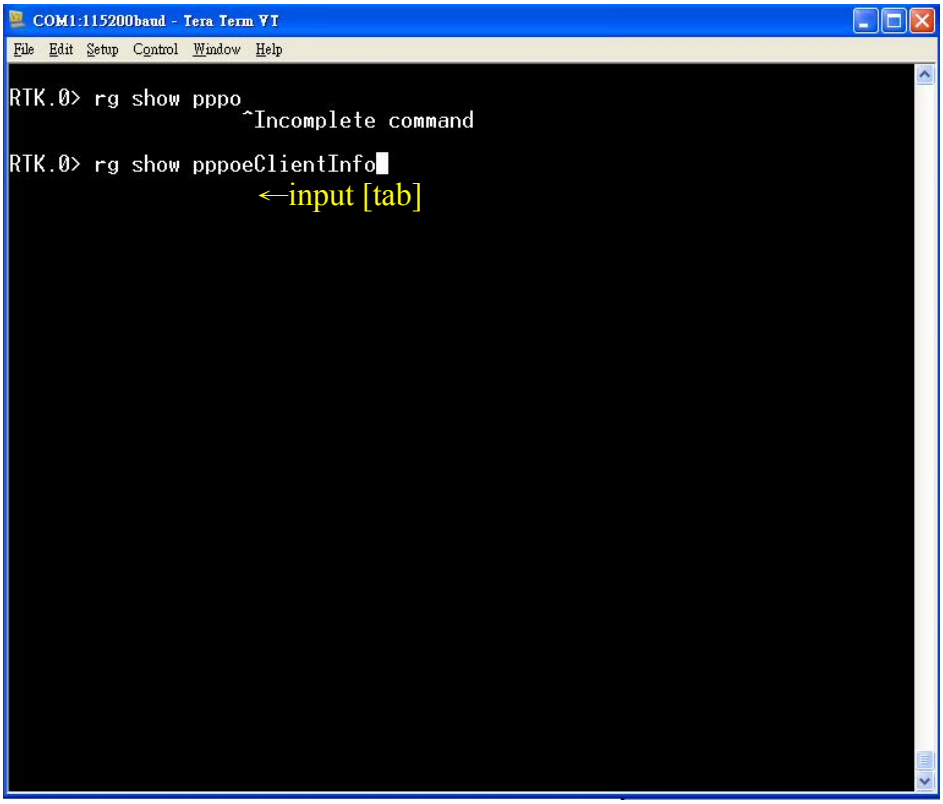


Figure17. DiagShell command-line-completion

9.1.3. Mapping for RG API & Diag Cmd

All RG API are related to the diag command with first keyword "rg". the second keyword indicate the action (such as add, delete, get…etc) of API. And the third keyword indicate the feature of the API (such as Interface, macEntry, ACL…etc).

For complex RG API suct as rtk_rg_XXXX_add( ), DiagShell separate these kind of RG API into two or three diag commands. We use command "***rg set XXXX ….***" for preparing the parameters of the API, and use command "***rg add XXXX entry***" for actually call to the rtk_rg_XXXX_add( ).

Following table shows all action of the command(Second keyword):

add	Call to RG API rtk_rg_XXXX_add( )
del	Call to RG API rtk_rg_XXXX_del( )
get	Call to RG API rtk_rg_XXXX_find( )
init	Especial call to RG API rtk_rg_initParam_set( )
set	Set the parameters before call to RG API rtk_rg_XXXX_add( )

clear	Clear the parameters
show	Show the current setting parameters

Table53. List of Diag Command second keywords

Following table shows the feature and API mapping of the command(Third keyword):

acl-filter	rtk_rg_aclFilterAndQos_add/del/find
algApps	rtk_rg_algApps_set/get
arp-entry	rtk_rg_arpEntry_add/del /find
binding	rtk_rg_vlanBinding_add/del/ find
cvlan	rtk_rg_cvlan_add/del
dhcpClientInfo	rtk_rg_dhcpClientInfo_set
dhcpRequest	rtk_rg_dhcpRequest_set
dmzHost	rtk_rg_dmzHost_set/get
dosFloodType	rtk_rg_dosFloodType_set/get
dosPortMaskEnable	rtk_rg_dosPortMaskEnable_set/get
dosType	rtk_rg_dosType_set/get
lan-intf	rtk_rg_lanInterface_add
mac-entry	rtk_rg_macEntry_add/del/find
macfilter	rtk_rg_macFilter_add/del/find
napt-connection	rtk_rg_naptConnection_add/del/find
neighbor-entry	rtk_rg_neighborEntry_add/del/find
phyPortForceAbility	rtk_rg_phyPortForceAbility_set/get
portEgrBandwidthCtrlRate	rtk_rg_portEgrBandwidthCtrlRate_set/get
portIgrBandwidthCtrlRate	rtk_rg_portIgrBandwidthCtrlRate_set/get
portMirror	rtk_rg_portMirror_set/get
pppoeClientInfoAfterDial	rtk_rg_pppoeClientInfoAfterDial_set
pppoeClientInfoBeforeDial	rtk_rg_pppoeClientInfoBeforeDial_set
qosDot1pPriRemapToInternalPri	rtk_rg_qosDot1pPriRemapToInternalPri_set/get
qosDot1pPriRemarkByInternalPri	rtk_rg_qosDot1pPriRemarkByInternalPri_set/get
qosDot1pPriRemarkByInternalPriEgressPortEnable	rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set/get
qosDscpRemapToInternalPri	rtk_rg_qosDscpRemapToInternalPri_set/get
qosDscpRemarkByDscp	rtk_rg_qosDscpRemarkByDscp_set/get
qosDscpRemarkByInternalPri	rtk_rg_qosDscpRemarkByInternalPri_set/get
qosDscpRemarkEgressPortEnableAndSrcSelect	rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set/get
qosInternalPriDecisionByWeight	rtk_rg_qosInternalPriDecisionByWeight_set/get

qosInternalPriMapToQueueId	rtk_rg_qosInternalPriMapToQueueId_set/get
qosPortBasedPriority	rtk_rg_qosPortBasedPriority_set/get
qosStrictPriorityOrWeightFairQueue	rtk_rg_qosStrictPriorityOrWeightFairQueue_set/get
serverInLanAppsIpAddr	rtk_rg_algServerInLanAppsIpAddr_add/del
shareMeter	rtk_rg_shareMeter_set/get
softwareSourceAddrLearningLimit	rtk_rg_softwareSourceAddrLearningLimit_set/get
stormControl	rtk_rg_stormControl_add/del/find
upnpConnection	rtk_rg_upnpConnection_add/del/find
url-filter	rtk_rg_urlFilterString_add/del/find
virtualServer	rtk_rg_virtualServer_add/del/find
wan-intf	rtk_rg_wanInterface_add
wan-intf-static-info	rtk_rg_staticInfo_set

Table54. List of Diag Command Third keywords

9.2. NIC Driver network device name & HW Port Index mapping

Lan Port mapping is eth0(PortIdx=0), eth0.2(PortIdx=1), eth0.3(PortIdx=2), eth0.4(PortIdx=3), and nas0(PortIdx-4) is used for PON WAN Port while (PortIdx=5) is used for RGMII WAN Port.

```
# echo 0 eth0 > /proc/rtl8686gmac/dev_port_mapping
# echo 1 eth0.2 > /proc/rtl8686gmac/dev_port_mapping
# echo 2 eth0.3 > /proc/rtl8686gmac/dev_port_mapping
# echo 3 eth0.4 > /proc/rtl8686gmac/dev_port_mapping
# echo 4 nas0 > /proc/rtl8686gmac/dev_port_mapping <4:PON, 5:RGMII>
```

9.3. Diag Shell for Realtek RG APIs

Enter diag shell by command **diag**:

9.3.1. Initial RG API

User can use following command for simply resetting all hardware tables. And then configurate Lan/Wan Interfcae or other features of the gateway. However, without callback functions for automatically synchronizing hardware tables and protocol stack, user should configurate the state of protocol stack by himself. The basic commands for configurating LAN/WAN Interface of protocol stack can be referenced by Sec. Set Lan/Wan Configuration of Protocal Stack.

```
RTK.0>rg init
```

9.3.2. Initial RG API and register callback function

LiteRomeDriver/RomeDriver reserves some hook points in RTK RG APIs for registering callback functions. Callback functions is used to synchronize Hardware Tables and Protocal Stack, such as sync Rouing Table. Diag Shell supports a set of default callback functions, and following command shows how to register all of them from the start. Besides, IgmpSnooping Module and macBasedTagDecision can be enable/sidable by user-self by following command, too. Enter parameter 1 as enable, and 0 as disable.

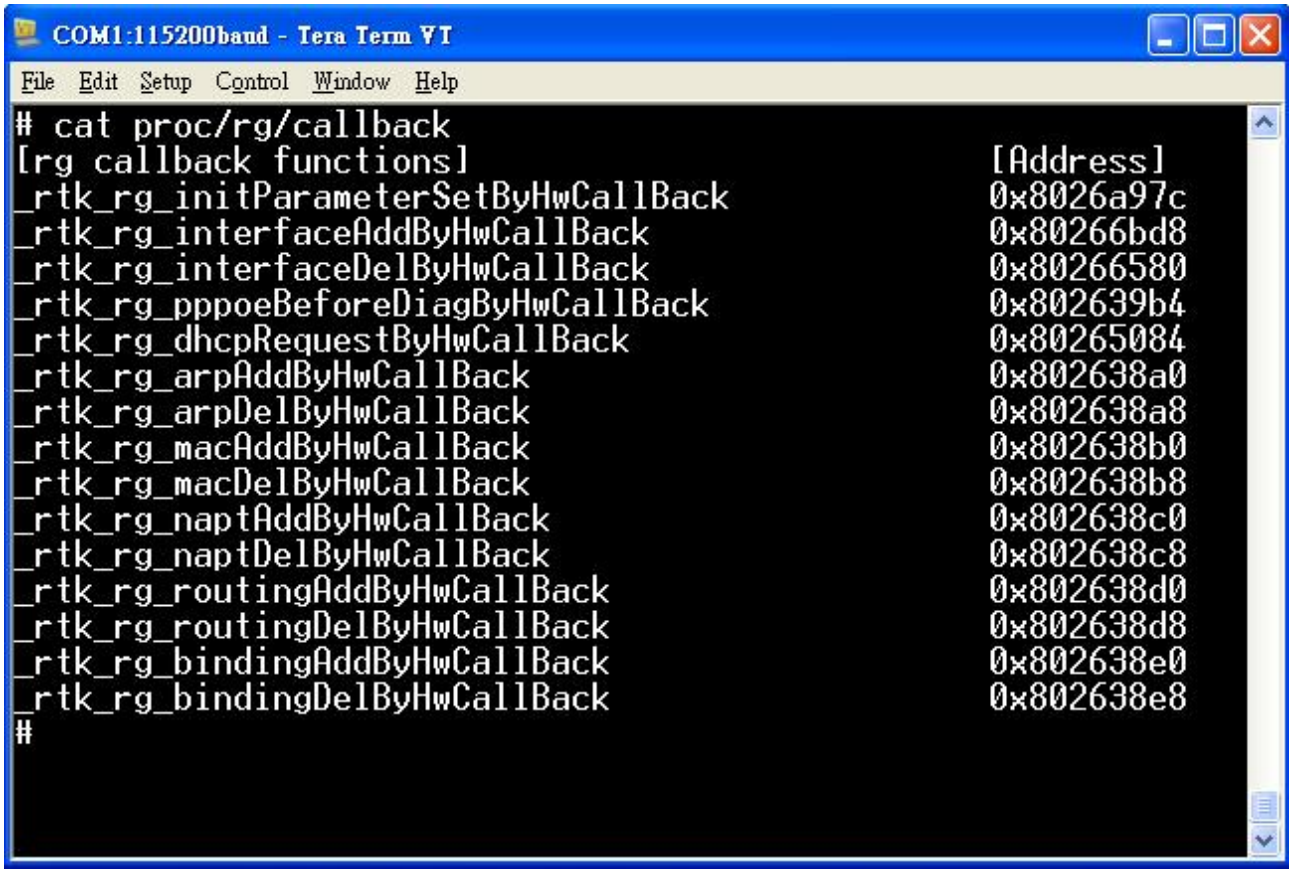
```
RTK.0> rg init callback default igmpSnoopingEnable 1 macBasedTagDecision 1
wanPortGponMode 0
```

User can register his own callback functions without using default callback functions. but need to remember where he stores it in Kernal memory (ex, oxffffffff). Following command shows how to register callback function (The memory address just a sample, it should be assigned based on real environment.):

```
RTK.0>rg set callback arpAddByHwCallBack 0xffffff00
RTK.0>rg set callback arpDelByHwCallBack 0xffffff01
RTK.0>rg set callback bindingAddByHwCallBack 0xffffff02
RTK.0>rg set callback bindingDelByHwCallBack 0xffffff03
RTK.0>rg set callback initByHwCallBack 0xffffff04
RTK.0>rg set callback interfaceAddByHwCallBack 0xffffff05
RTK.0>rg set callback interfaceDelByHwCallBack 0xffffff06
RTK.0>rg set callback macAddByHwCallBack 0xffffff07
RTK.0>rg set callback macDelByHwCallBack 0xffffff08
RTK.0>rg set callback naptAddByHwCallBack 0xffffff09
RTK.0>rg set callback naptDelByHwCallBack 0xffffff0a
RTK.0>rg set callback neighborAddByHwCallBack 0xffffff0b
RTK.0>rg set callback neighborDelByHwCallBack 0xffffff0c
RTK.0>rg set callback pppoeBeforeDiagByHwCallBack 0xffffff0d
RTK.0>rg set callback routingAddByHwCallBack 0xffffff0e
RTK.0>rg set callback routingDelByHwCallBack 0xffffff0f
RTK.0>rg set callback v6RoutingAddByHwCallBack 0xffffff10
RTK.0>rg set callback v6RoutingDelByHwCallBack 0xffffff11
RTK.0>rg init callback igmpSnoopingEnable 1 macBasedTagDecision 1 wanPortGponMode 0
```

Default callback function is located in memory that can be found by following command:

```
# cat proc/rg/callback
```



9.3.3. Add LAN Interface

RTK.0> **rg set lan-intf ip-version 2 gateway-mac 00:e0:4c:86:70:01 ip-addr 192.168.1.1 ip-mask 255.255.255.0 ipv6-addr 0::0 ipv6_network_mask_length 0 port-mask 0x5f untag-mask 0x5f intf-vlan_id 9 vlan-based-pri-enable disable mtu 1500 isIVL 0**

RTK.0>**rg add lan-intf entry**

add lan-intf[0] success. <This is the corresponding message.>

9.3.4. Add WAN Interface

RTK.0>**rg set wan-intf wan-type 0 gateway-mac 00:e0:4c:86:70:02 wan-port 4 port-binding-mask 0x0 egress-vlan-tag-on 0 egress-vlan-id 8 vlan-based-pri-enable disable isIVL 0**

RTK.0>**rg add wan-intf entry**

add wan-intf[1] success. <This is the corresponding message.>

9.3.5. Set Static WAN Configuration

(1)Using following command will allow Router get gateway_mac_addr found by gateway_ip_addr automatically(Not supported in Lite RomeDriver).

```
RTK.0>rg set wan-intf-static-info ip-version 2 napt_enable 1 ip_addr 192.168.150.116
ip_network_mask 255.255.255.248 ipv4_default_gateway_on 1 gateway_ipv4_addr
192.168.150.117 mtu 1500 gw_mac_auto_learn_for_ipv4 1 gateway_mac_addr_for_ipv4
00:00:00:00:00:00
RTK.0>rg set wan-intf-static-info-ipv6 ipv6_addr 2002::1 ipv6_mask_length 64
ipv6_default_gateway_on 1 gateway_ipv6_addr 2002::2 mtu 1500
gw_mac_auto_learn_for_ipv6 1 gateway_mac_addr_for_ipv6 00:00:00:00:00:00
RTK.0>rg add wan-intf-static-info intf-index 1 <1 is the corresponding value of Add WAN Intf >
add static info to interface[1] success. <This is the corresponding message.>
```

(2) Using following command to Assign gateway_mac_addr manually:

```
RTK.0>rg set wan-intf-static-info ip-version 2 napt_enable 1 ip_addr 192.168.150.116
ip_network_mask 255.255.255.248 ipv4_default_gateway_on 1 gateway_ipv4_addr
192.168.150.117 mtu 1500 gw_mac_auto_learn_for_ipv4 0 gateway_mac_addr_for_ipv4
00:e0:01:02:03:04
RTK.0>rg set wan-intf-static-info-ipv6 ipv6_addr 2002::1 ipv6_mask_length 64
ipv6_default_gateway_on 1 gateway_ipv6_addr 2002::2 mtu 1500
gw_mac_auto_learn_for_ipv6 0 gateway_mac_addr_for_ipv6    00:e0:01:02:03:04
RTK.0>rg add wan-intf-static-info intf-index 1 <1 is the corresponding value of Add WAN Intf >
add static info to interface[1] success. <This is the corresponding message.>
```

9.4. Diag Shell for Realtek RG Runtime API

9.4.1. Get RG API Version Info

```
RTK.0>rg get version
Lunar:1049 Switch:42382 RG:865 User:55269 <This is the corresponding message.>
```

Once NAPT, ARP Auto learning is enabled, there is no need to command following Diag Shell.

9.4.2. Add MAC Entry

```
RTK.0>rg set mac-entry mac-address 00:e0:01:02:03:44 isIVL 0 fid 0 vlan_id 9 port_idx 1
```

static_entry 1
RTK.0>**rg add mac-entry entry**
add macEntry[44] success. <This is corresponding message.>

Implying add MAC to MAC Table Index 45.

9.4.3. Add ARP Entry

RTK.0>**rg set arp-entry macEntryIdx 44 ip_addr 192.168.1.2 static_entry 1 valid 1**
RTK.0>**rg add arp-entry entry**
add arpEntry[2] success. <This is corresponding message.>

Implying add ARP to ARP Table Index 2.

9.4.4. Add NAPT Entry

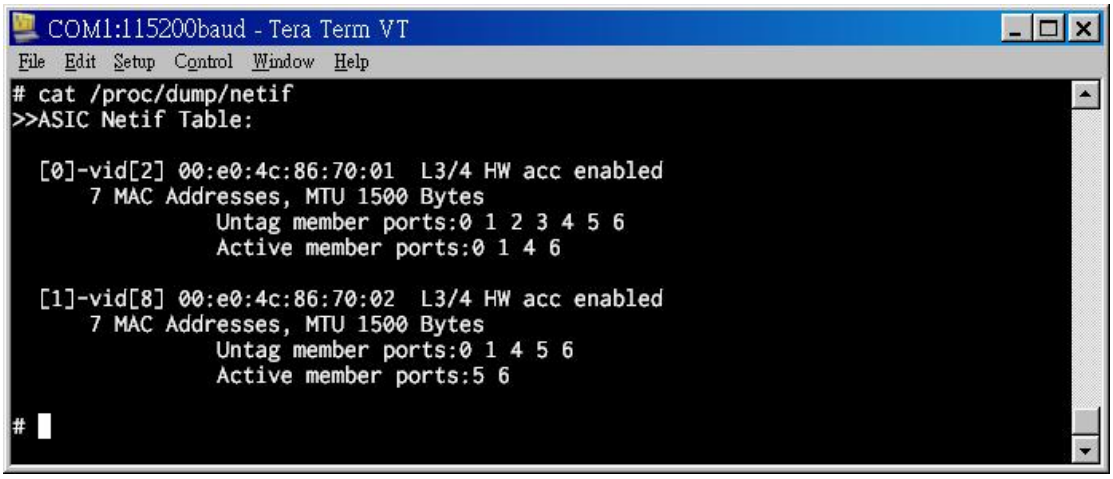
RTK.0>**rg set napt-connection is_tcp 1 local_ip 192.168.1.2 remote_ip 10.10.10.10 wan_intf_idx 1 local_port 16811 remote_port 1010 external_port 1000 outbound_pri_valid 0 outbound_priority 0 inbound_pri_valid 0 inbound_priority 0**
RTK.0>**rg add napt-connection entry**
add naptConn[320] success. <This is corresponding message.>

Implying add NAPT to NAPT Table Index 320.

10. Realtek RG Debug Tools

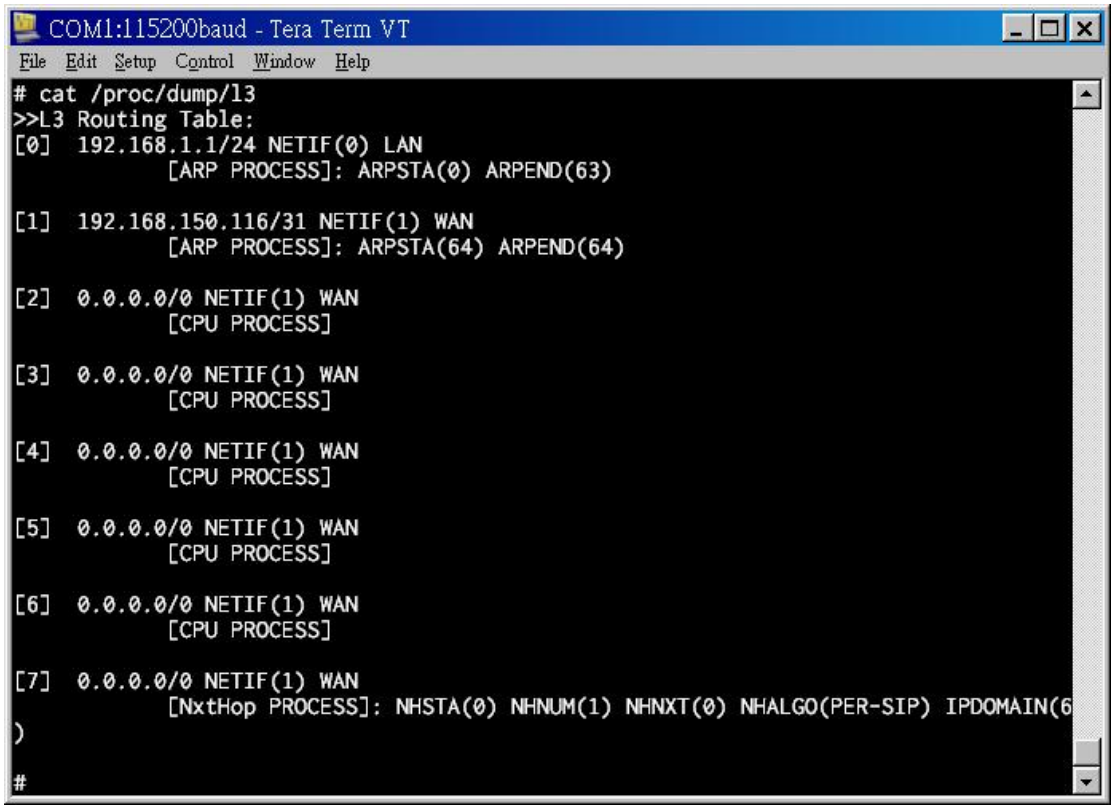
If Realtek RG Debug Tools is enabled in Kernel menuconfig, the following debug tools can be used. Each Hardware Table can be dumped by following commands:

cat /proc/dump/netif <Show Network Interface Table >



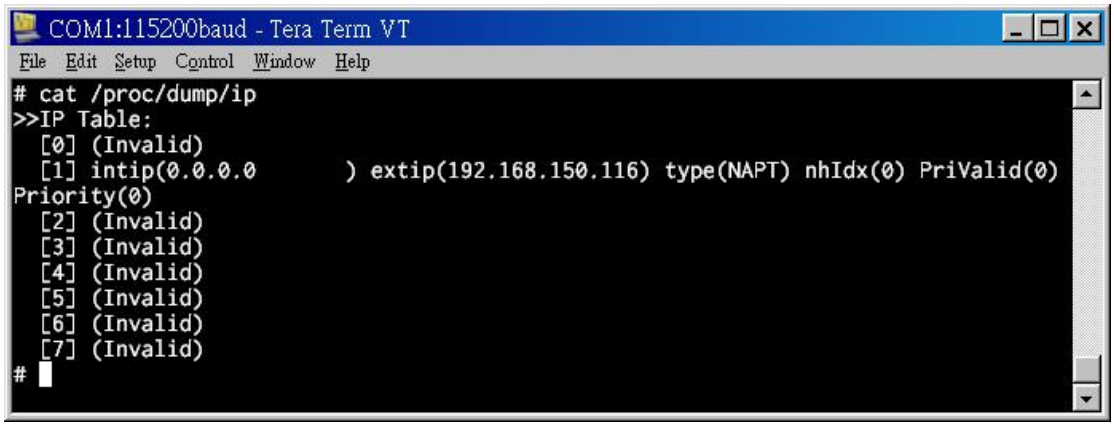
After LAN/WAN is Set up, two Network Interfaces should be showed.

cat /proc/dump/l3 <Show Network Routing Table >



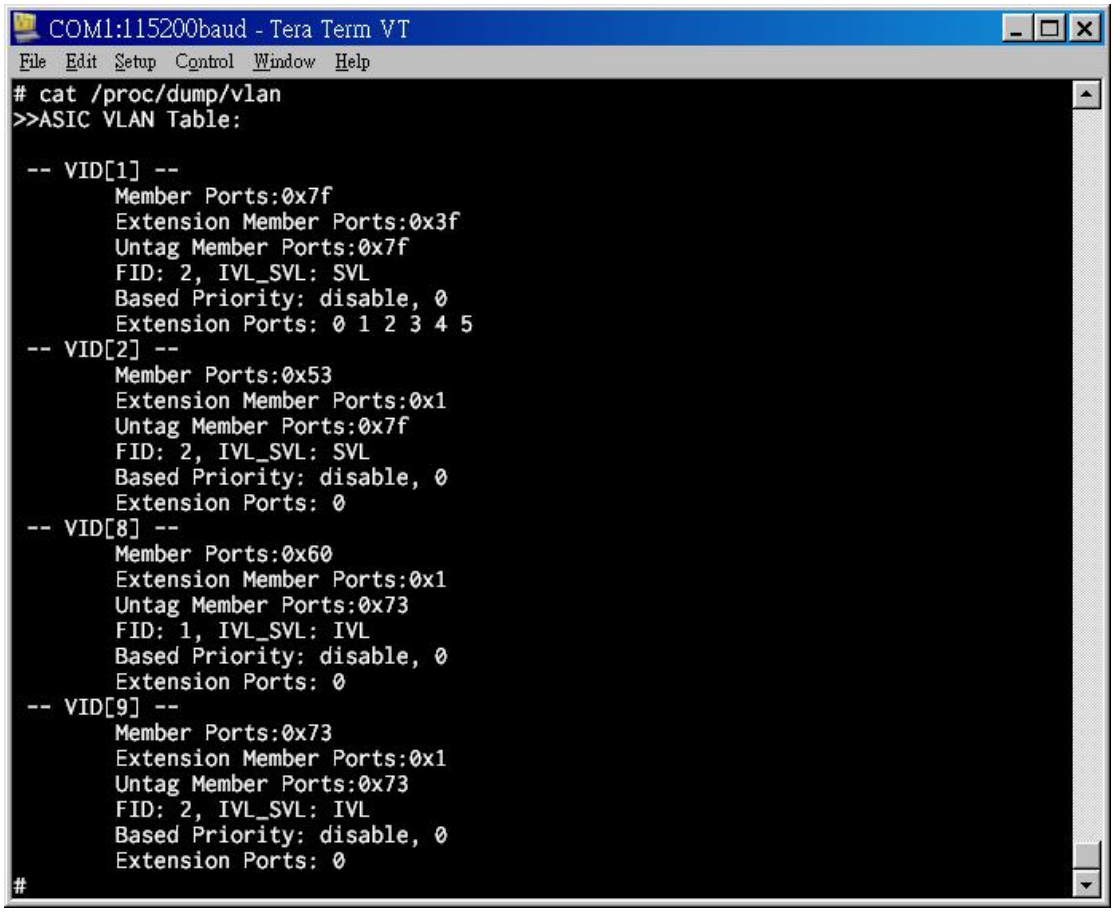
Once LAN/WAN is set up, two Interface Route[0],[1] and one Default Route[7] should be showed.

cat /proc/dump/ip <Show Network Interface Table >



Once LAN/WAN is set up, one set of WAN IP should be set in NAPT, and IP Table Index should refer to Network Interface Table Index.

cat /proc/dump/vlan <Show VLAN Interface Table >



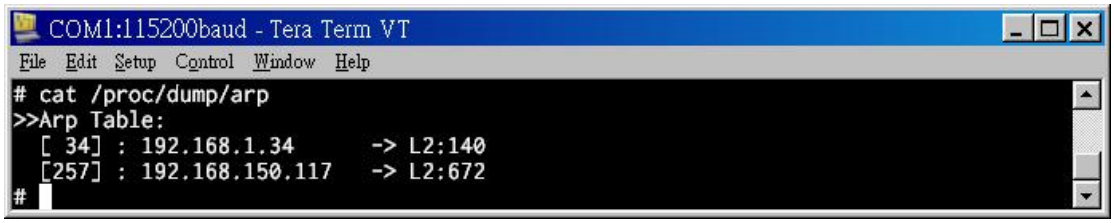
VID1 is all VLAN member port of LAN & WAN.

VID2 is all VLAN member port of LAN Interface.

VID8 is USER added VLAN member port of WAN Interface.

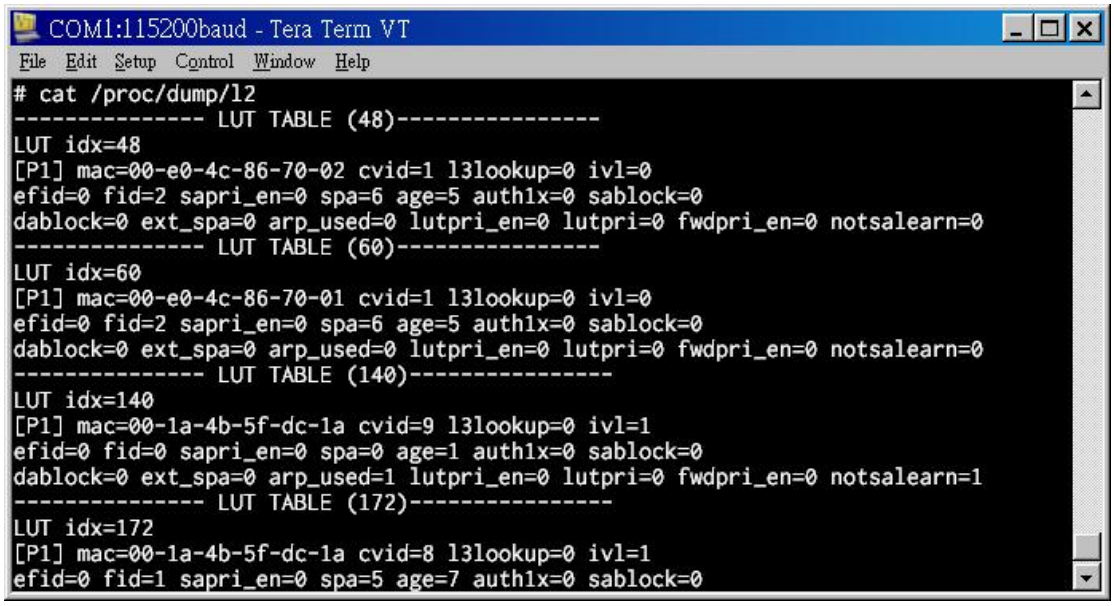
VID9 is USER added VLAN member port of LAN Interface.

cat /proc/dump/arp <Show ARP Table >

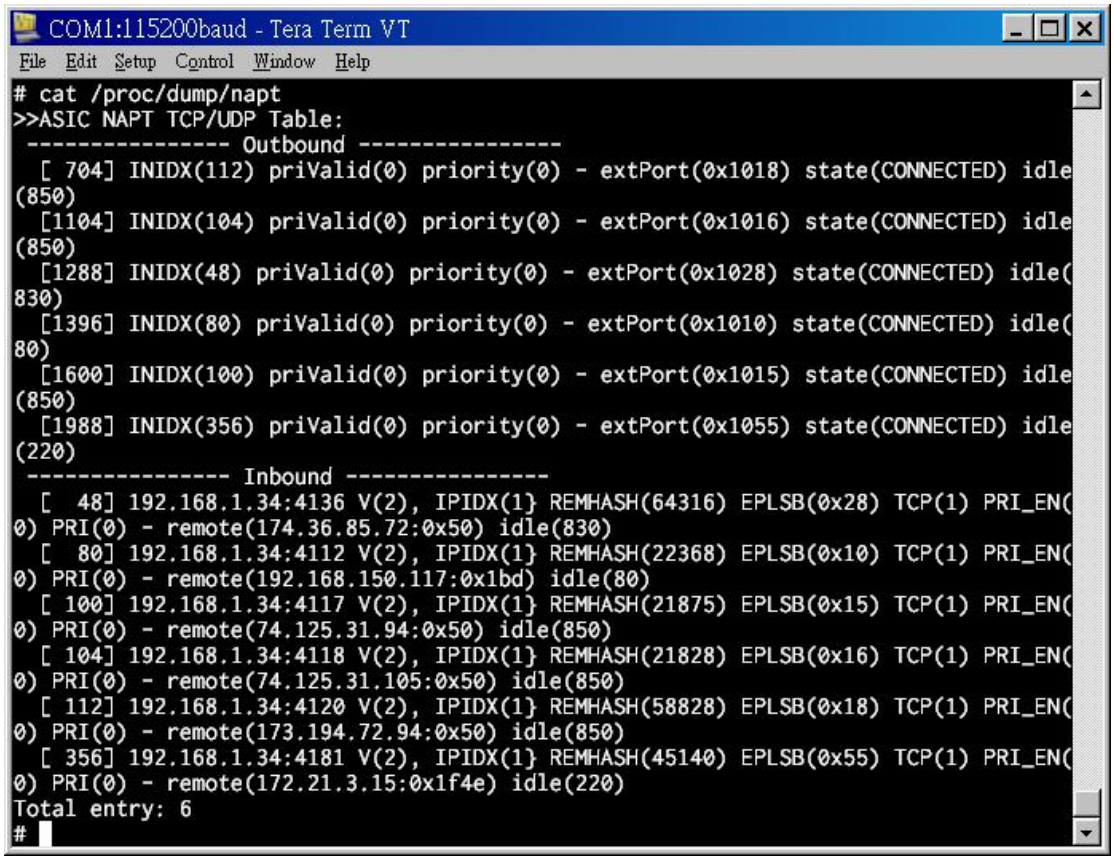


cat /proc/dump/nh <Show NextHop Table >

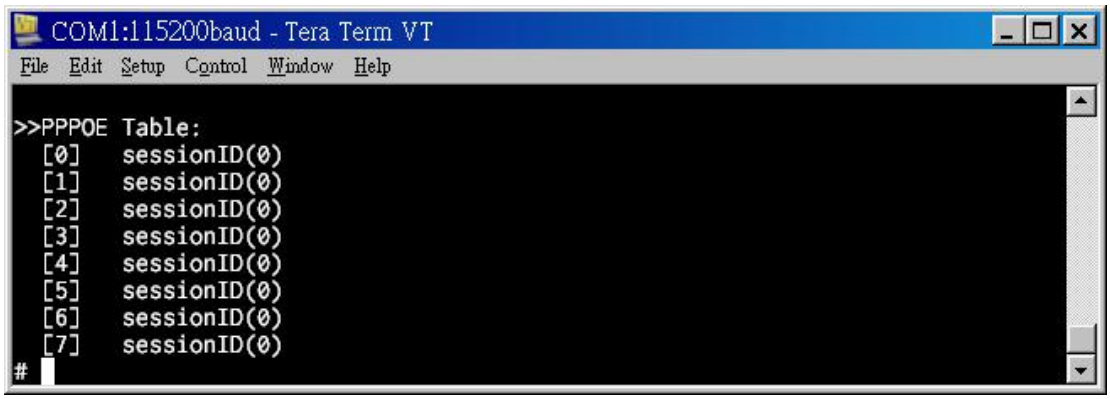
cat /proc/dump/l2 <Show MAC Table >



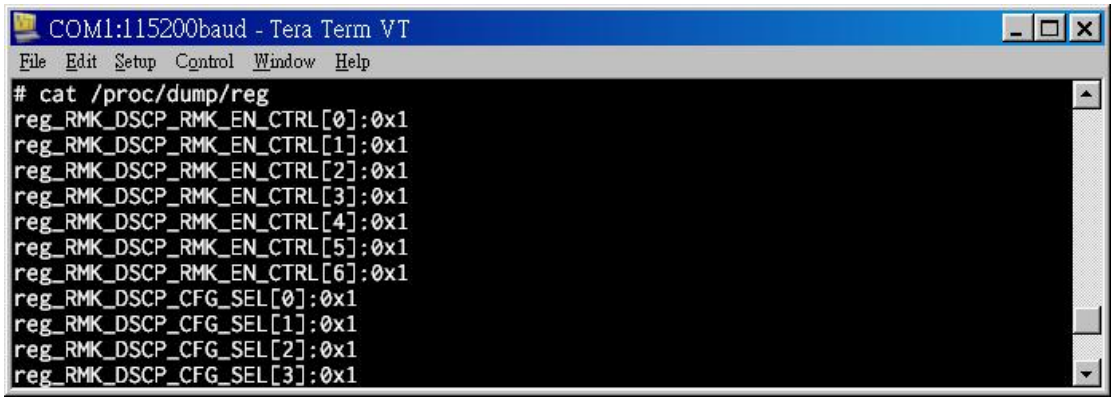
cat /proc/dump/napt <Show NAPT Table >



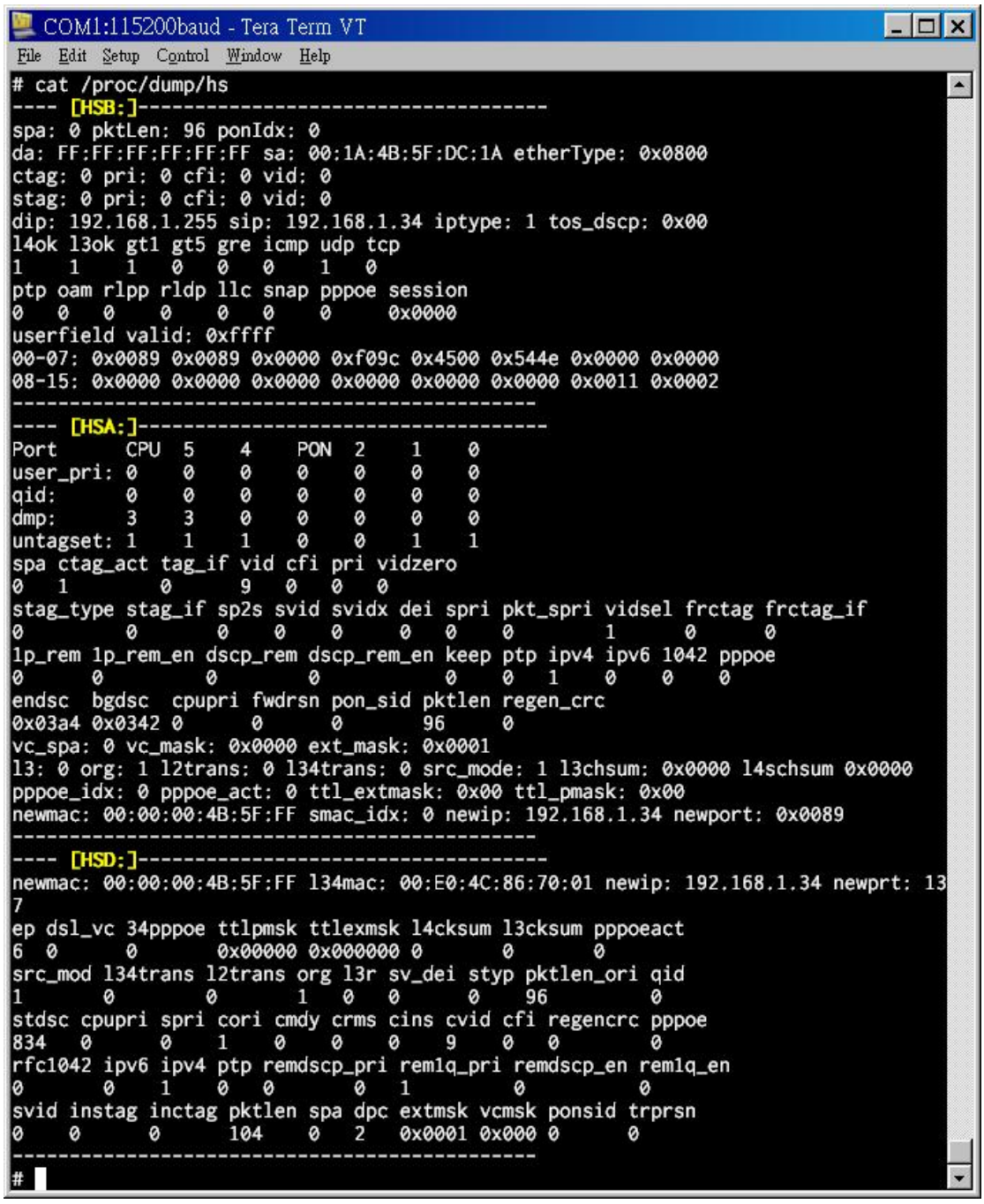
cat /proc/dump/pppoe <Show PPPoE Table >



cat /proc/dump/reg <Show Register >



cat /proc/dump/hs <Show Hardware Detail Information for Bug Report>



11. Test Report

11.1. Chariot-LAN to LAN Bridge Mode-TCP Throughput

Using High_Performance_Throughput.scr Script, 2 LAN to LAN TCP connections speed is at

902Mbps. Because the speed of Chariot related to Testing Computer efficiency, this test result should be treated as reference for LAN to WAN speed and only.

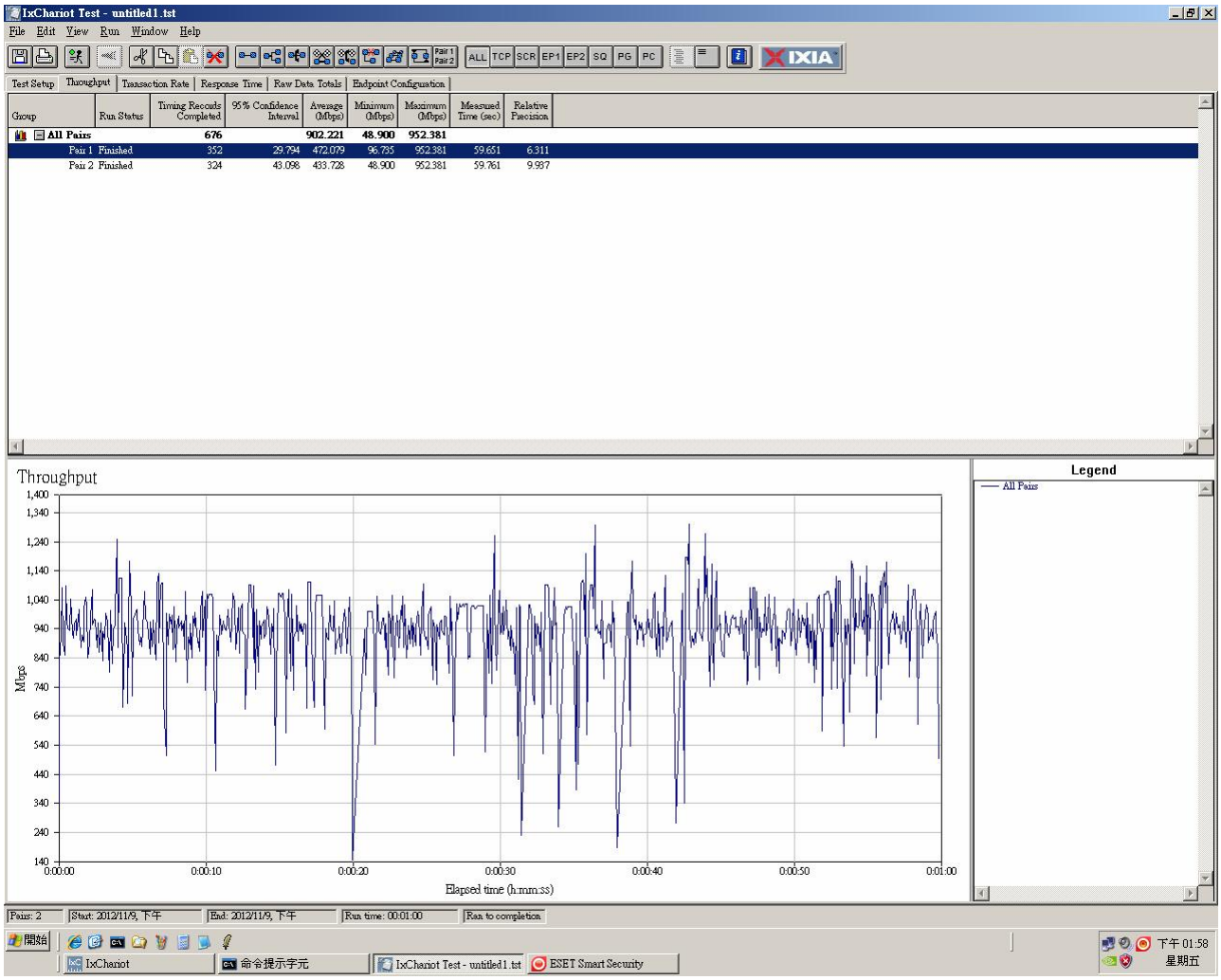


Figure18. Chariot LAN to LAN TCP connection

11.2. Chariot-LAN to WAN NAPT Mode-TCP Throughput

Using High_Performance_Throughput.scr Script, 2 LAN to WAN TCP connections speed is at 905Mbps. Because the speed of Chariot related to Testing Computer efficiency, this test result should be treated as reference for LAN to LAN speed and only.

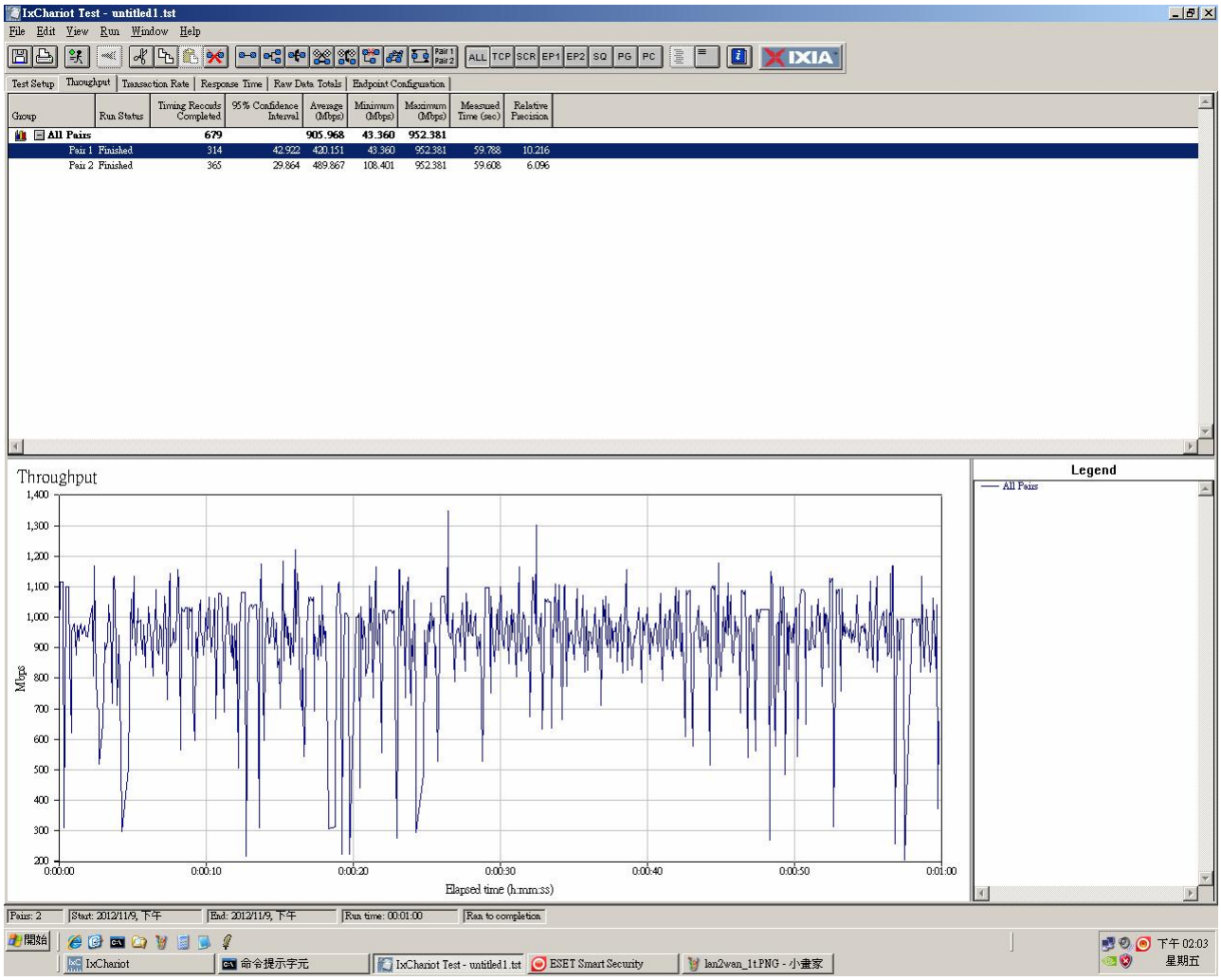


Figure19. Chariot LAN to WAN TCP connection

11.3. IXIA-LAN to LAN Bridge Mode-TCP(Packet Size:1518Bytes)

Data Bit Rate: **986.962**Mbps

Total Packets/Sec:**81486.31**fps

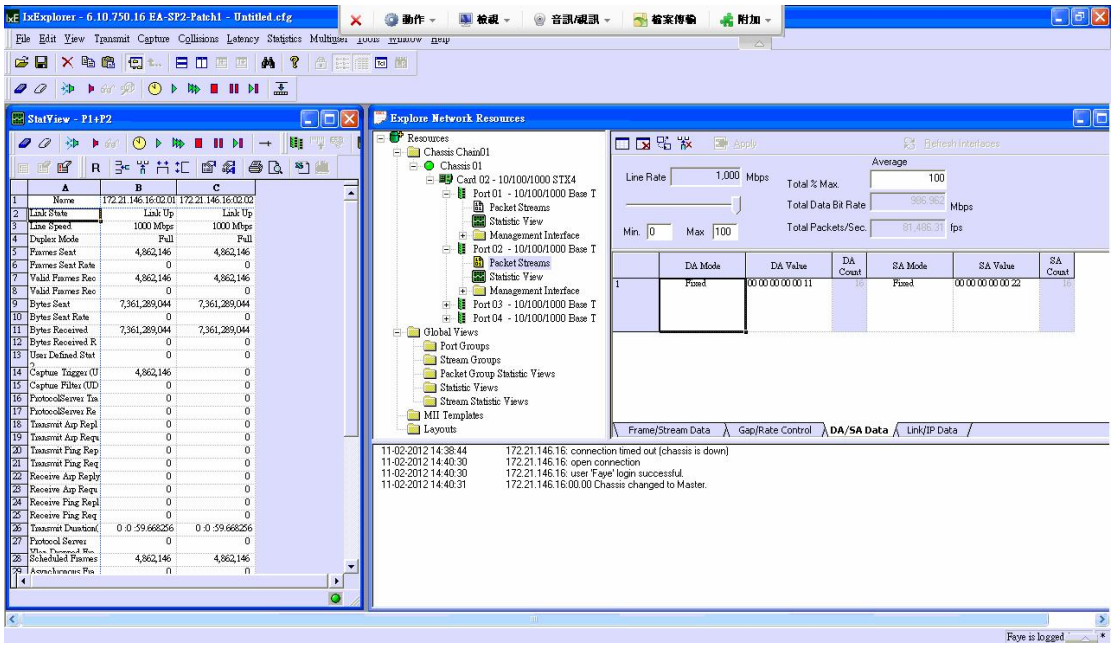


Figure20. IXIA LAN to LAN TCP connection(Large Packet)

11.4. IXIA-LAN to LAN Bridge Mode-TCP(Packet Size:64Bytes)

Data Bit Rate: **761.905**Mbps

Total Packets/Sec:**1488095.2**fps

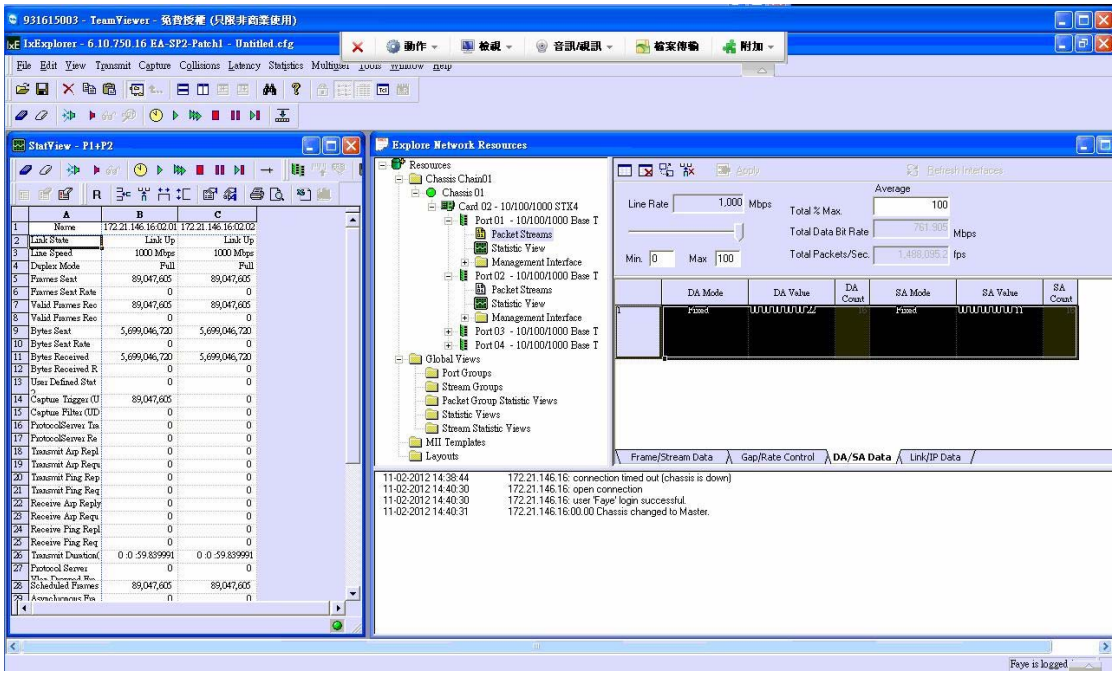


Figure21. IXIA LAN to LAN TCP connection(Small Packet)

11.5. IXIA-LAN to UTP WAN NAPT Mode-TCP(Packet Size:1518Bytes)

Data Bit Rate: **986.962**Mbps

Total Packets/Sec:**81486.31**fps

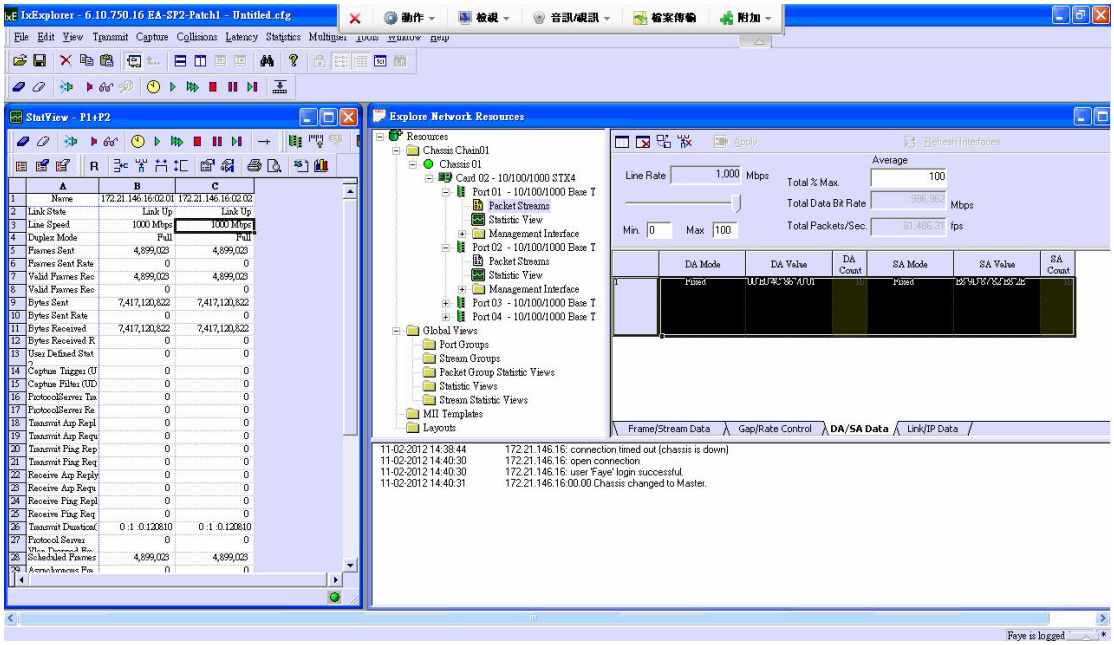


Figure22. IXIA LAN to WAN TCP connection(Large Packet)

11.6. IXIA-LAN to UTP WAN NAPT Mode-TCP(Packet Size:64Bytes)

Data Bit Rate: **761.905**Mbps

Total Packets/Sec:**1488095.2**fps

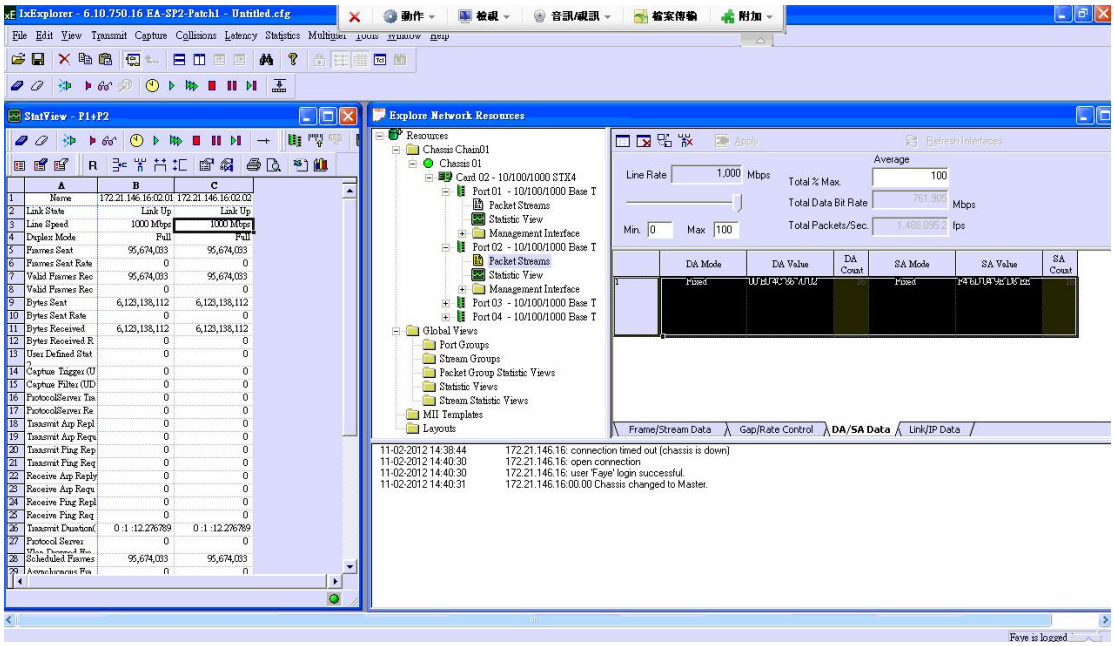


Figure23. IXIA LAN to WAN TCP connection(Small Packet)

11.7. IXIA-LAN to GPON WAN NAPT Mode-UDP(Packet Size:1518Bytes)

Up-stream Data Bit Rate: **986.996**Mbps

Total Packets/Sec: **81,274,382**ps

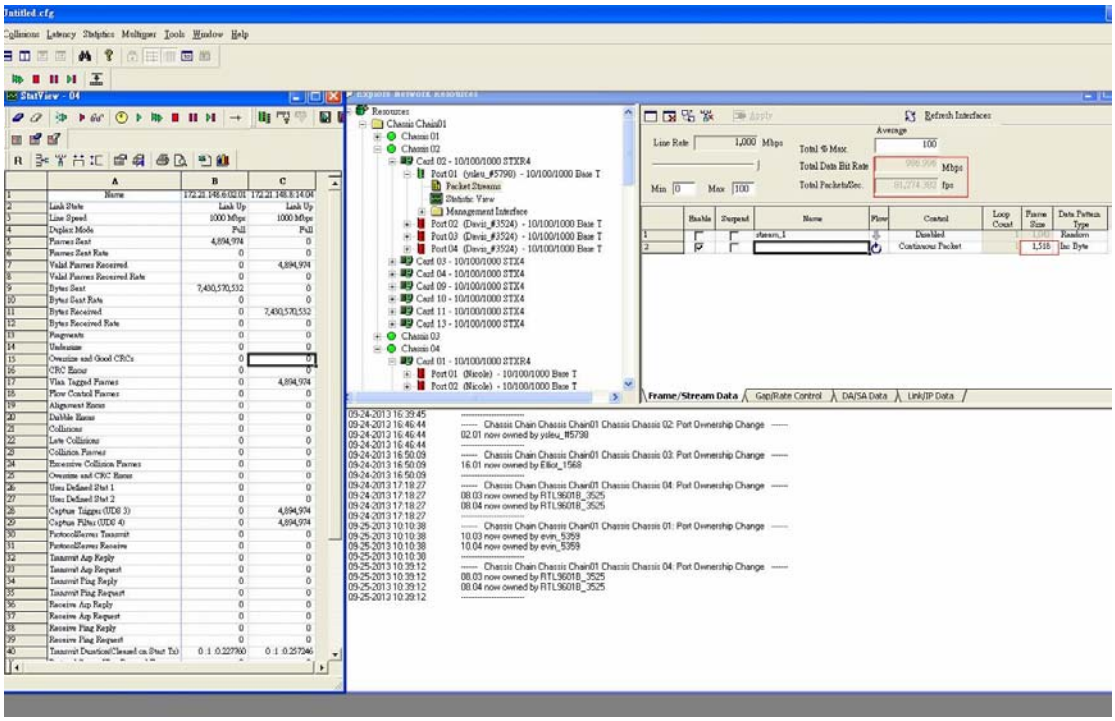


Figure24. IXIA LAN to GPON WAN UDP connection (Large Packet)

11.8. IXIA-GPON WAN to LAN NAPT Mode-UDP(Packet Size:1518Bytes)

Down-stream Data Bit Rate: **986.996**Mbps

Total Packets/Sec: **81,274,382**ps

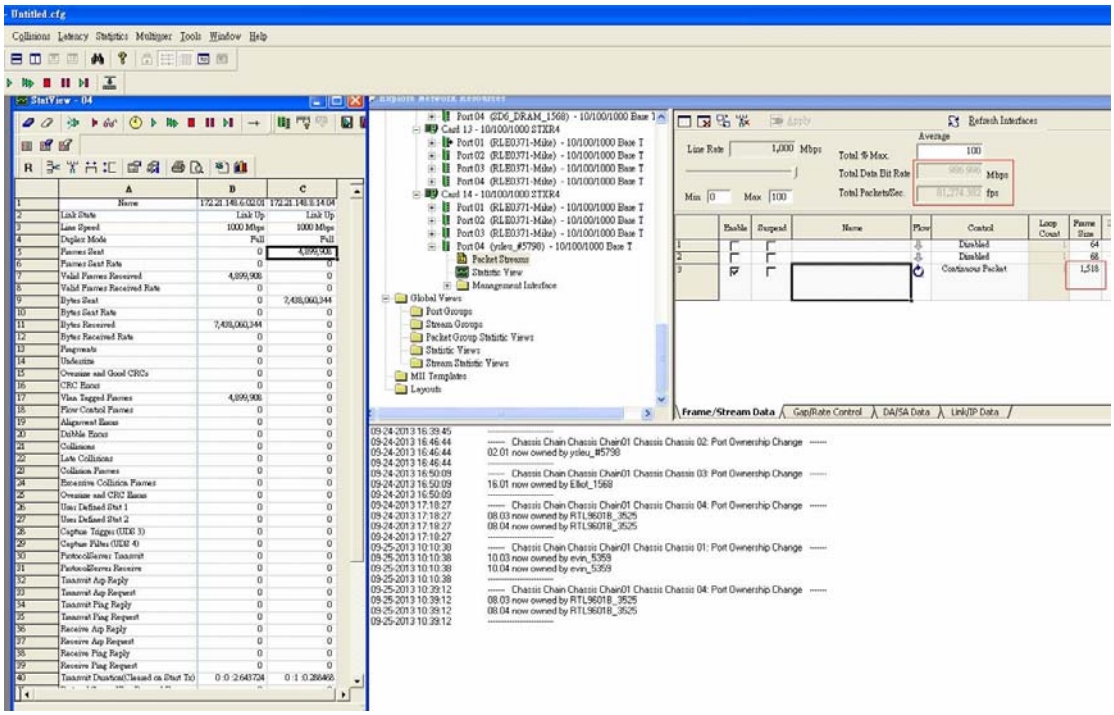


Figure25. IXIA GPON WAN to LAN UDP connection (Large Packet)

11.9. IXIA-LAN to GPON WAN NAPT Mode-UDP(Packet Size:64Bytes)

Up-stream Data Bit Rate: **761.905**Mbps

Total Packets/Sec: **1,488,095.2**ps

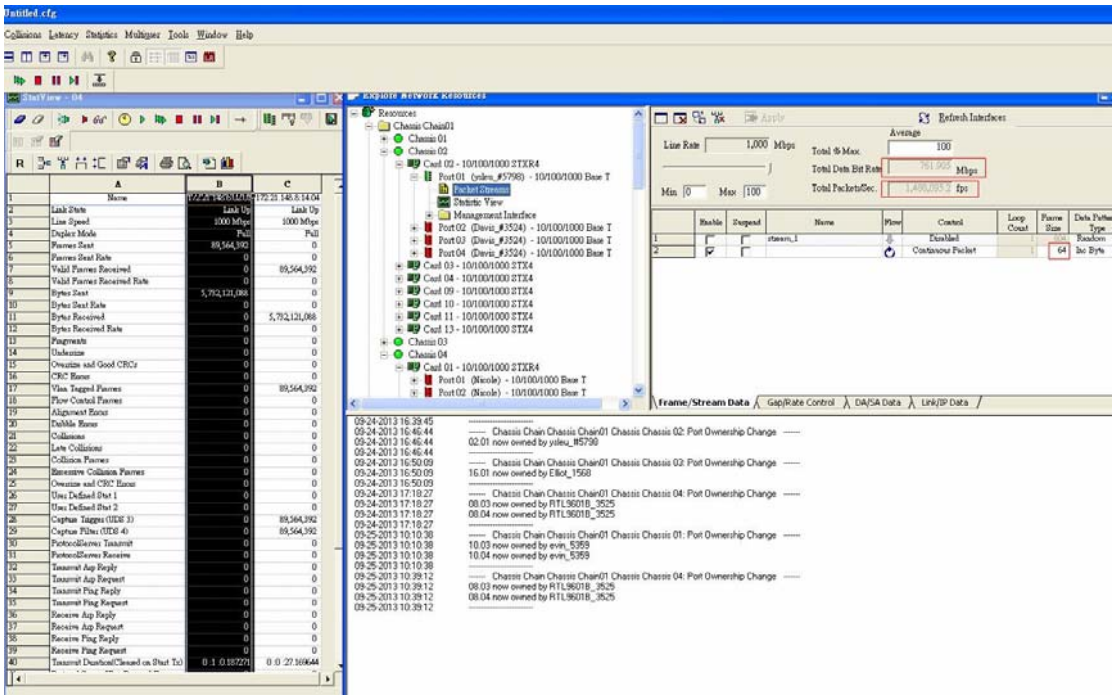


Figure26. IXIA LAN to GPON WAN UDP connection(Small Packet)

11.10. IXIA-GPON WAN to LAN NAPT Mode-UDP(Packet Size:64Bytes)

Down-stream Data Bit Rate: **761.905**Mbps

Total Packets/Sec: **1,488,095.2**ps

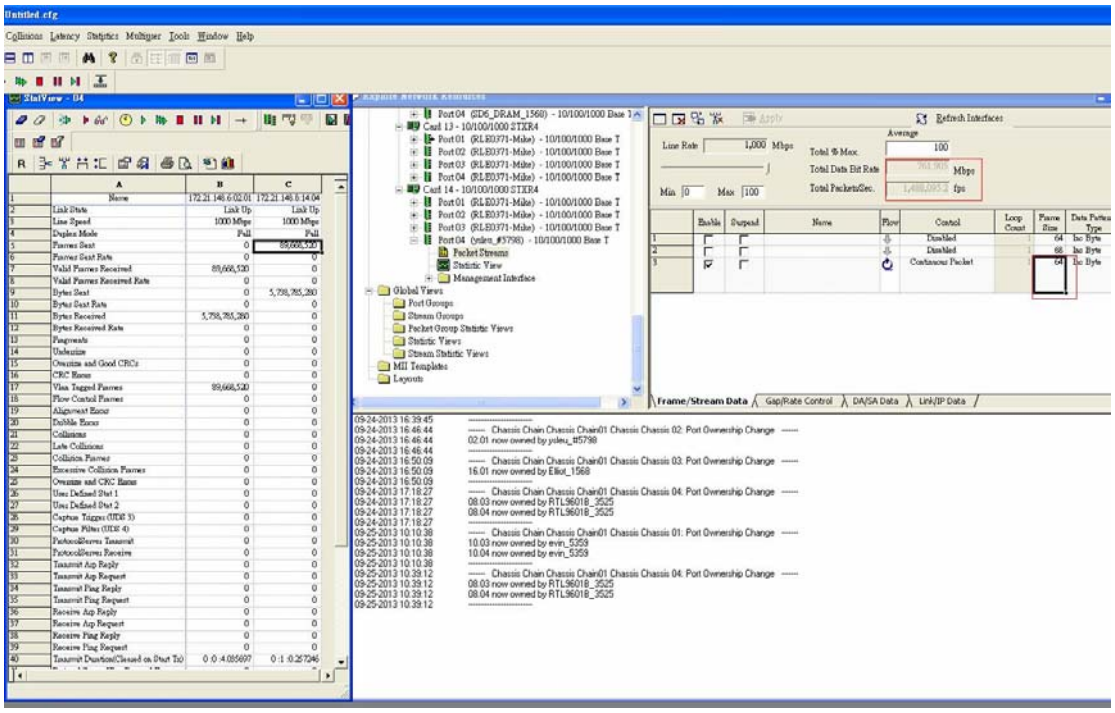


Figure27. IXIA GPON WAN to LAN UDP connection(Small Packet)

11.11. IXIA(veriwave)-WIFI to WAN Bridge Mode-UDP

Veriwave Configuration:

Learning Time	2 sec
Achieved Transmit Time	10 sec
Settle Time	2 sec
Aging Time	5 sec
Acceptable Loss	1.00%
Protocol	UDP

Wifi Configuration:

Protocol	802.11n
MCS Rate	15
Channel Width	40 MHz
AMPDU	on
DUT Configuration	
GMAC Ring Size	512
Wifi IC	8192ER
Channel	11

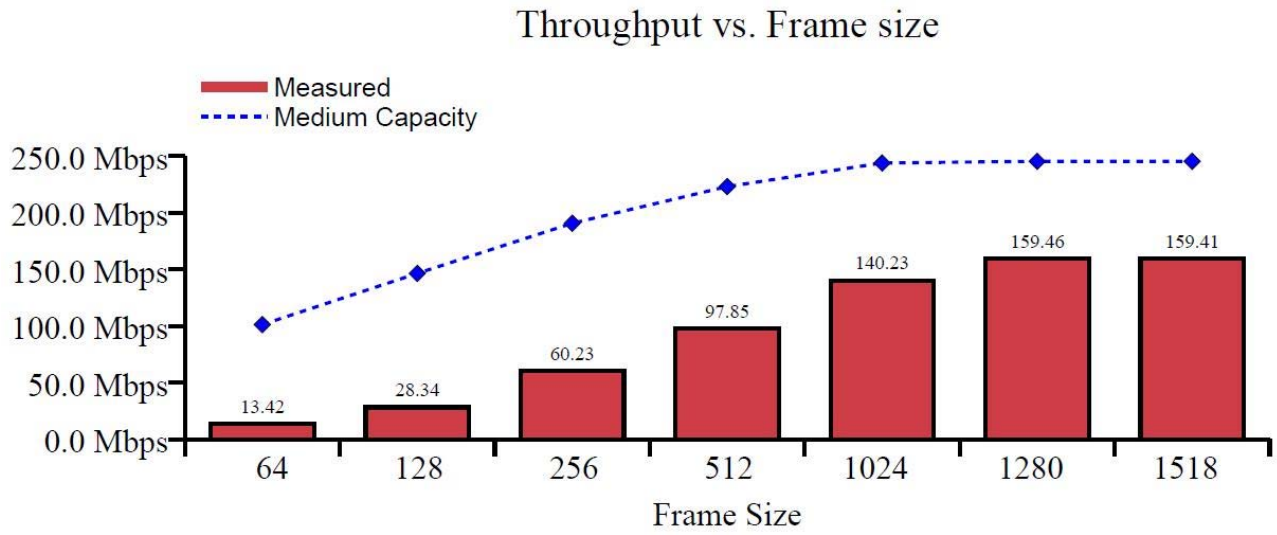


Figure28. IXIA WIFI to WAN(Bridge) UDP connection

11.12. IXIA(veriwave)-WIFI to WAN NAPT Mode-UDP

Veriwave Configuration:

Learning Time	2 sec
Achieved Transmit Time	10 sec
Settle Time	2 sec
Aging Time	5 sec
Acceptable Loss	1.00%
Protocol	UDP

Wifi Configuration:

Protocol	802.11n
MCS Rate	15
Channel Width	40 MHz
AMPDU	on
DUT Configuration	
GMAC Ring Size	512
Wifi IC	8192ER
Channel	11

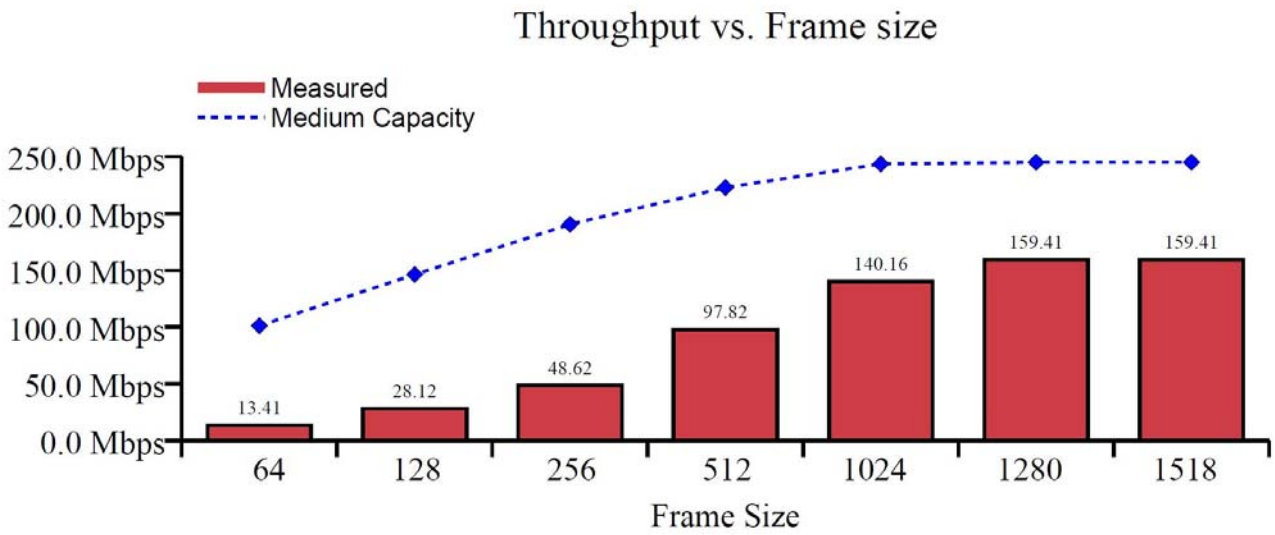


Figure29. IXIA WIFI to WAN(NAPT) UDP connection

12. RTK RG APIs Prototype

12.1. rtk_rg_driverVersion_get

int32 rtk_rg_driverVersion_get(rtk_rg_VersionString_t **version_string*)
Get the RG rome driver version number.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*version_string
[in] The pointer of version structure.
[out] The pointer to the stucture with version string.

Comments
version_string.version_string - the char string of version, at most 20 characters.

Return Codes
RT_ERR_RG_OK
RT_ERR_RG_NULL_POINTER the buffer parameter may be NULL

12.2. rtk_rg_initParam_get

int32 rtk_rg_initParam_get(rtk_rg_initParams_t **init_param*)
Get the initialized call-back functions.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*init_param
[in] The instance of rtk_rg_initParams_t.
[out] The instance of rtk_rg_initParams_t with saved call

Comments
igmpSnoopingEnable - control IGMP snooping should on or off.
macBasedTagDecision - control tag decision based on MAC should on or off.
wanPortGponMode - indicate if the switch configure as GPON mode or not.
init_param.arpAddByHwCallBack - this call-back function pointer should be called when ARP entry added.
init_param.arpDelByHwCallBack - this call-back function pointer should be called when ARP entry deleted.
init_param.macAddByHwCallBack - this call-back function pointer should be called when MAC entry added.
init_param.macDelByHwCallBack - this call-back function pointer should be called when MAC entry deleted.
init_param.routingAddByHwCallBack - this call-back function pointer should be called when Routing entry added.
init_param.routingDelByHwCallBack - this call-back function pointer should be called when Routing entry deleted.
init_param.naptAddByHwCallBack - this call-back function pointer should be called when NAPT entry added.
init_param.naptDelByHwCallBack - this call-back function pointer should be called when NAPT entry deleted. init_param.bindingAddByHwCallBack - this call-back function pointer should be called when Binding entry added.
init_param.bindingDelByHwCallBack - this call-back function pointer should be called when Binding entry added.

Return Codes
RT_ERR_RG_OK
RT_ERR_RG_NULL_POINTER the buffer parameter may be NULL
RT_ERR_RG_INITPM_UNINIT init parameter structure are all NULL pointers

12.3. rtk_rg_initParam_set

int32 rtk_rg_initParam_set(rtk_rg_initParams_t *_init_param_)
Set the initialized call-back functions, and reset all Global variables.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*init_param
[in] The instance of rtk_rg_initParams_t with saved call

Comments
This function should be called before any other RG APIs.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	The RG module is failed to initialize
RT_ERR_RG_VLAN_SET_FAIL	Set up default CPU VLAN or LAN VLAN failed

12.4. rtk_rg_lanInterface_add

int32 rtk_rg_lanInterface_add(rtk_rg_lanIntfConf_t *_lan_info_, int *_intf_idx_)
Create one new LAN interface, add related entries into HW tables and Global variables.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*lan_info
[in] LAN interface configuration structure.
*intf_idx
[out] Return the index of new created LAN interface.

Comments
lan_info.gmac - the gateway MAC address of this LAN interface.
lan_info.ip_addr - the IP address of this LAN interface.
lan_info.ip_network_mask - the mask decides how many hosts in this LAN.
lan_info.port_mask - which ports belong to this LAN.
lan_info.untag_mask - which ports in this LAN should be egress untag.
lan_info.extport_mask - which extension ports belong to this LAN.
lan_info.intf_vlan_id - the default VLAN ID of this LAN.
lan_info.vlan_based_pri - indicate what VLAN priority this WAN interface should carry.
lan_info.vlan_based_pri_enable - indicate VLAN-based priority enable or not.
lan_info.mtu - the maximum transmission unit of this LAN interface.
lan_info.isIVL - how to learning layer2 record, IVL or SVL.
This function should be called after rtk_rg_initParam_set and before any WAN interface creation.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameter may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_NOT_INIT	the RG module didn't init
RT_ERR_RG_ENTRY_FULL	the interface number is beyond predefined limitation
RT_ERR_RG_MODIFY_LAN_AT_WAN_EXIST	the LAN interface can not add after WAN interface exist
RT_ERR_RG_PORT_USED	the interface has port overlap with other interface
RT_ERR_RG_ARP_FULL	ARP table is not available for this new interface
RT_ERR_RG_INTF_GET_FAIL	
RT_ERR_RG_VLAN_SET_FAIL	
RT_ERR_RG_INTF_SET_FAIL	
RT_ERR_RG_ROUTE_SET_FAIL	

12.5. rtk_rg_wanInterface_add

int32 rtk_rg_wanInterface_add(rtk_rg_wanIntfConf_t **wanintf***, int** **wan_intf_idx***)**
Add WAN interface.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*wanintf
[in] The configuration structure of WAN interface.
*wan_intf_idx
[out] The created new WAN interface index.

Comments
wanintf.wan_type - this WAN interface type, can be Bridge mode, DHCP mode, PPPoE mode, etc.
wanintf.gmac - the gateway MAC address of this WAN interface.
wanintf.wan_port_idx - indicate which port this WAN interface belong to.
wanintf.port_binding_mask - indicate which ports and extension ports should be bound to this WAN interface.
wanintf.egress_vlan_tag_on - indicate the egress packet should carry VLAN CTAG or not.
wanintf.egress_vlan_id - indicate what VLAN ID this WAN interface should carry.
wanintf.vlan_based_pri_enable - indicate VLAN-based priority enable or not.
wanintf.vlan_based_pri - indicate what VLAN priority this WAN interface should carry.
When adding WAN interface except Bridge mode, this function should be called before each sub function like rtk_rg_staticInfo_set, rtk_rg_dhcpClientInfo_set, rtk_rg_pppoeClientInfoBeforeDial_set, and rtk_rg_pppoeClientInfoAfterDial_set.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_NOT_INIT	the RG module didn't init
RT_ERR_RG_ENTRY_FULL	the table is full
RT_ERR_RG_DEF_ROUTE_EXIST	the default internet WAN is exist
RT_ERR_RG_LAN_NOT_EXIST	WAN interface should be add after LAN interface created
RT_ERR_RG_PON_INVALID	for chip A1, the PON port can not be set as WAN port
RT_ERR_RG_CHIP_NOT_SUPPORT	the function is not support for the chip
RT_ERR_RG_UNBIND_BDWAN_SHOULD_EQUAL_LAN_VLAN	unbind bridge WAN can't assign VLAN id different than LAN interface
RT_ERR_RG_BIND_WITH_UNBIND_WAN	the global switch macBasedTagDecision didn't turn on
RT_ERR_RG_PORT_BIND_GET_FAIL	
RT_ERR_RG_ROUTE_GET_FAIL	
RT_ERR_RG_INTF_GET_FAIL	
RT_ERR_RG_INTF_SET_FAIL	
RT_ERR_RG_VLAN_GET_FAIL	
RT_ERR_RG_VLAN_SET_FAIL	
RT_ERR_RG_PPPOE_SET_FAIL	
RT_ERR_RG_NXP_SET_FAIL	
RT_ERR_RG_WANTYPE_SET_FAIL	
RT_ERR_RG_PORT_BIND_SET_FAIL	

12.6. rtk_rg_staticInfo_set

int32 rtk_rg_staticInfo_set(int *wan_intf_idx***, rtk_rg_ipStaticInfo_t** **static_info***)**
Set static mode information to the indexed WAN interface.
Defined in: rtk_rg_liteRomeDriver.h

Parameters wan_intf_idx
[in] The interface index of previous setup for static mode.
*static_info
[in] The configuration related to static mode.

Comments static_info.napt_enable - indicate this WAN interface should turn on napt or not.
static_info.ip_addr - the IP address of this WAN interface.
static_info.ip_network_mask - the mask decides how many hosts in this WAN.
static_info.default_gateway_on - indicate this WAN interface is default internet interface or not.
static_info.gateway_ip_addr - indicate which gateway this WAN interface is heading to.
static_info.mtu - the maximum transmission unit of this LAN interface.
static_info.gw_mac_auto_learn - switch to turn on auto-ARP request for gateway MAC.
static_info.gateway_mac_addr - pointer to gateway's mac if the auto_learn switch is off.
The interface index should point to a valid WAN interface with matched type.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ENTRY_NOT_EXIST	the wan interface was not added before
RT_ERR_RG_ARP_FULL	ARP table is not available for this new interface
RT_ERR_RG_ARP_NOT_FOUND	the remote gateway is not found or time
RT_ERR_RG_GW_MAC_NOT_SET	gateway mac should be set for Lite RomeDriver
RT_ERR_RG_ENTRY_FULL	for chip A1, the PON port can not be set as WAN port
RT_ERR_RG_ROUTE_GET_FAIL	the function is not support for the chip
RT_ERR_RG_ROUTE_SET_FAIL	unbind bridge WAN can't assign VLAN id different than LAN interface
RT_ERR_RG_EXTIP_GET_FAIL	the global switch macBasedTagDecision didn't turn on
RT_ERR_RG_EXTIP_SET_FAIL	

12.7. rtk_rg_dhcpRequest_set

int32 rtk_rg_dhcpRequest_set(int *wan_intf_idx***)**
Set DHCP request for the indexed WAN interface.
Defined in: rtk_rg_liteRomeDriver.h

Parameters wan_intf_idx
[in] The interface index of previous setup for DHCP mode.

Comments None

Return Codes RT_ERR_RG_OK

12.8. rtk_rg_dhcpClientInfo_set

int32 rtk_rg_dhcpClientInfo_set(int *wan_intf_idx***, rtk_rg_ipDhcpClientInfo_t** ***dhcpClient_info*****)**
Set DHCP mode information to the indexed WAN interface.
Defined in: rtk_rg_liteRomeDriver.h

Parameters wan_intf_idx
[in] The interface index of previous setup for DHCP mode.

*dhcpClient_info
[in] The configuration related to the interface.

Comments dhcpClient_info.hw_info - the static_info sturcture of this DHCP mode WAN interface.
dhcpClient_info.status - the status of DHCP mode WAN interface as client, leased or released.
The client_info contains the Hardware information which contains static mode settings, and the interface index should point to a WAN interface with matched type.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ENTRY_NOT_EXIST	the wan interface was not added before
RT_ERR_RG_ARP_FULL	ARP table is not available for this new interface
RT_ERR_RG_ARP_NOT_FOUND	the remote gateway is not found or time
RT_ERR_RG_GW_MAC_NOT_SET	gateway mac should be set for Lite RomeDriver
RT_ERR_RG_INTF_GET_FAIL	for chip A1, the PON port can not be set as WAN port
RT_ERR_RG_ROUTE_GET_FAIL	the function is not support for the chip
RT_ERR_RG_ENTRY_FULL	unbind bridge WAN can't assign VLAN id different than LAN interface
RT_ERR_RG_ROUTE_SET_FAIL	the global switch macBasedTagDecision didn't turn on
RT_ERR_RG_EXTIP_GET_FAIL	
RT_ERR_RG_EXTIP_SET_FAIL	

12.9. rtk_rg_pppoeClientInfoBeforeDial_set

int32 rtk_rg_pppoeClientInfoBeforeDial_set(int *wan_intf_idx*,
rtk_rg_pppoeClientInfoBeforeDial_t **app_info*)
Set PPPoE mode information to the indexed WAN interface before dial.
Defined in: rtk_rg_liteRomeDriver.h

Parameters wan_intf_idx
[in] The interface index of previous setup for PPPoE mode.
*app_info
[in] The configuration related to PPPoE mode.

Comments app_info.username - saved user account information for PPPoE server.
app_info.password - saved account password for PPPoE server.
app_info.auth_type - indicate the PPPoE server use PAP or CHAP.
app_info.pppoe_proxy_enable - indicate the proxy of PPPoE server enable or not.
app_info.max_pppoe_proxy_num - how many PPPoE proxy at most.
app_info.auto_reconnect - indicate that we reconnect automatically no matter what.
app_info.dial_on_demond - indicate that we connect only on demand.
app_info.idle_timeout_secs - indicate how long we should turn off connection after idle.
app_info.stauts - indicate the status of connection.
app_info.dialOnDemondCallBack - this function would be called when the interface has traffic flow.
app_info.idleTimeOutCallBack - this function would be called when the interface didn't have traffic for indicated timeout period.
The required account information is kept and this function should be called before rtk_rg_pppoeClientInfoAfterDial_set.
The interface index should point to a WAN interface with matched type.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.10. rtk_rg_pppoeClientInfoAfterDial_set

int32 rtk_rg_pppoeClientInfoAfterDial_set(int *wan_intf_idx***,**
rtk_rg_pppoeClientInfoAfterDial_t **clientPppoe_info***)*
Set PPPoE mode information to the indexed WAN interface after dial.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
wan_intf_idx
[in] The interface index of previous setup for PPPoE mode.
*clientPppoe_info
[in] The configuration related to PPPoE mode.

Comments
clientPppoe_info.hw_info - the static_info sturcture of this PPPoE mode WAN interface.
clientPppoe_info.sessionId - stored PPPoE session ID currently used.
The client_info contains the Hardware information which contains static mode settings, and the interface index should point to a WAN interface with matched type.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ENTRY_NOT_EXIST	the wan interface was not added before
RT_ERR_RG_ARP_FULL	ARP table is not available for this new interface
RT_ERR_RG_ARP_NOT_FOUND	the remote gateway is not found or time
RT_ERR_RG_GW_MAC_NOT_SET	gateway mac should be set for Lite RomeDriver
RT_ERR_RG_ROUTE_GET_FAIL	for chip A1, the PON port can not be set as WAN port
RT_ERR_RG_ENTRY_FULL	the function is not support for the chip
RT_ERR_RG_ROUTE_SET_FAIL	unbind bridge WAN can't assign VLAN id different than LAN interface
RT_ERR_RG_EXTIP_GET_FAIL	the global switch macBasedTagDecision didn't turn on
RT_ERR_RG_EXTIP_SET_FAIL	

12.11. rtk_rg_interface_del

int32 rtk_rg_interface_del(int *lan_or_wan_intf_idx***)**
Delete the indicated interface, may be LAN or WAN interface.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
lan_or_wan_intf_idx
[in] The index of interface.

Comments
Before deleting any interface, it should be created first.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_MODIFY_LAN_AT_WAN_EXIST	LAN interface should not be deleted when WAN interface existed
RT_ERR_RG_INTF_GET_FAIL	the wan interface was not added before
RT_ERR_RG_ROUTE_GET_FAIL	ARP table is not available for this new interface
RT_ERR_RG_ROUTE_SET_FAIL	the remote gateway is not found or time
RT_ERR_RG_NXP_GET_FAIL	gateway mac should be set for Lite RomeDriver
RT_ERR_RG_PPPOE_SET_FAIL	for chip A1, the PON port can not be set as WAN port
RT_ERR_RG_NXP_SET_FAIL	the function is not support for the chip
RT_ERR_RG_ROUTE_SET_FAIL	unbind bridge WAN can't assign VLAN id different than LAN interface
RT_ERR_RG_EXTIP_GET_FAIL	the global switch macBasedTagDecision didn't turn on

RT_ERR_RG_EXTIP_SET_FAIL
RT_ERR_RG_WANTYPE_GET_FAIL
RT_ERR_RG_WANTYPE_SET_FAIL
RT_ERR_RG_INTF_SET_FAIL
RT_ERR_RG_PORT_BIND_GET_FAI
L
RT_ERR_RG_PORT_BIND_SET_FAIL

12.12. rtk_rg_intfInfo_find

int32 rtk_rg_intfInfo_find(rtk_rg_intfInfo_t **intf_info*, **int** **valid_lan_or_wan_intf_idx*)
Return the information structure of interface, eithor LAN or WAN.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*intf_info
[in] An empty buffer for storing the structure.
[out] Returned valid interface information structure, eithor LAN or Wint
*valid_lan_or_wan_intf_idx
[in] The index of interface to find.
[out] Return the index of the valid record.

Comments
intf_info.intf_name - the name of found interface.
intf_info.is_wan - indicate this interface is WAN interface or LAN interface.
intf_info.lan_intf - if is_wan is 0, this structure contain all information about the LAN interface.
intf_info.wan_intf - if is_wan is 1, this structure contain all information about the WAN interface.
intf_info.ingress_packet_count - the count of how many packet had passed into this interface.
intf_info.ingress_byte_count - the count of how many data had passed into this interface in bytes.
intf_info.egress_packet_count - the count of how many packet had passed through this interface out.
intf_info.egress_byte_count - the count of how many data had passed through this interface out in bytes.
If the input idx interface is not exist, it will auto incress to next index, until return a valid one or meet end.
Besides, if the input idx is -1, intf_info.lan_intf.ip_addr or intf_info.lan_intf.ipv6_addr is given,
the matching interface information and it's index will be return. Bridge WAN interface could not be found
through this mode.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ENTRY_NOT_EXIST	the wan interface was not added before

12.13. rtk_rg_cvlan_add

int32 rtk_rg_cvlan_add(rtk_rg_cvlan_info_t **cvlan_info*)
Add customer VLAN setting.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*cvlan_info

[in] The VLAN configuration.

Comments cvlan_info.vlanId - which VLAN identifier need to create, between 0 and 4095.
cvlan_info.isIVL - how to learning layer2 record, by SVL or by IVL setting.
cvlan_info.memberPortMask - which port contained in this VLAN identifier.
cvlan_info.untagPortMask - which port contained in this VLAN identifier should be untag.
cvlan_info.vlan_based_pri_enable - indicate VLAN-based priority enable or not.
cvlan_info.vlan_based_pri - indicate what VLAN priority this VLAN should carry.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	the input parameters may be NULL
RT_ERR_RG_NOT_INIT	the input parameter contains illegal information or range
RT_ERR_RG_VLAN_USED_BY_INTERFACE	the prefered VLAN ID had been used as interface VLAN ID
RT_ERR_RG_VLAN_USED_BY_VLANBINDING	the prefered VLAN ID had been used as VLAN
RT_ERR_RG_CVLAN_CREATED	the VLAN ID had been created as customer VLAN before
RT_ERR_RG_CVLAN_RESERVED	the VLAN ID is reserved for system use
RT_ERR_RG_VLAN_SET_FAIL	for chip A1, the PON port can not be set as WAN port

12.14. rtk_rg_cvlan_del

int32 rtk_rg_cvlan_del(int *cvlan_id***)**
Delete customer VLAN setting.
Defined in: rtk_rg_liteRomeDriver.h

Parameters cvlan_id
[in] The VLAN identifier needed to be deleted.

Comments Here can not delete VLAN identifier used for interface or VLAN binding. Only the VLAN identifier created by rtk_rg_1qVlan_add can be deleted this way.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_VLAN_NOT_CREATED_BY_CVLAN	the deleting VLAN ID was not added as customer VLAN ID

12.15. rtk_rg_vlanBinding_add

int32 rtk_rg_vlanBinding_add(rtk_rg_vlanBinding_t **vlan_binding_info***, int** **vlan_binding_idx***)**
Add Port-VLAN binding rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters *vlan_binding_info
[in] The VLAN binding configuration about port index, interface, and VLAN ID
*vlan_binding_idx
[out] The index of added VLAN binding rule.

Comments vlan_binding_idx.port_idx - each VLAN binding rule can only assign one single port or extension port (not CPU port).
vlan_binding_idx.ingress_vid - the VLAN ID used to compare for binding.
vlan_binding_idx.wan_intf_idx - which WAN interface the matched packet should go.
The VLAN-binding rule can add only after the binding WAN interface is created at first.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL

RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_NOT_INIT	the RG module is not init
RT_ERR_RG_ENTRY_FULL	the prefered VLAN ID had been used as VLAN
RT_ERR_RG_VLAN_USED_BY_INTERFACE	the VLAN id can't used by interface
RT_ERR_RG_VLAN_USED_BY_1QVLAN	the VLAN id can't used by 1QVLAN api
RT_ERR_RG_BIND_WITH_UNBIND_WAN	the global switch macBasedTagDecision didn't turn on

12.16. rtk_rg_vlanBinding_del

int32 rtk_rg_vlanBinding_del(int *vlan_binding_idx***)**
Delete Port-VLAN binding rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
vlan_binding_idx
[in] The index of VLAN binding rule.

Comments
None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_VLAN_BIND_UNINIT	there is no vlan

12.17. rtk_rg_vlanBinding_find

int32 rtk_rg_vlanBinding_find(rtk_rg_vlanBinding_t **vlan_binding_info***, int** **valid_idx***)**
Find Port-VLAN binding rule if any.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*vlan_binding_info
[in] The binding configuration of port and VLAN ID.
*valid_idx
[in] The index of the VLAN binding entry.
[out] The index of first valid VLAN binding entry.

Comments
If the input parameter do not point to a valid record, it will continue to look for next valid one until end, and return the valid index by the passed input pointer.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_VLAN_BIND_UNINIT	there is no vlan

12.18. rtk_rg_algServerInLanAppsIpAddr_add

int32 rtk_rg_algServerInLanAppsIpAddr_add(rtk_rg_alg_serverIpMapping_t **srvIpMapping***)**
Add ServerInLan ALG function IP mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters *srvIpMapping
[in] The mapping of bitmask of ALG ServerInLan functions and server IP address.
Comments srvIpMapping.algType - indicate which ALG service should assign to the serverAddress.
srvIpMapping.serverAddress - indicate the server IP address.
Before call rtk_rg_algApps_set to setup Server In Lan service, this IP mapping should be enter at first, otherwise rtk_rg_algApps_set will return failure.
Return Codes RT_ERR_RG_OK
RT_ERR_RG_INVALID_PARAM the input parameters may be NULL
RT_ERR_RG_ALG_SRV_IN_LAN_EXIST the server ip address had been assigned

12.19. rtk_rg_algServerInLanAppsIpAddr_del

int32 rtk_rg_algServerInLanAppsIpAddr_del(rtk_rg_alg_type_t *delServerMapping***)**
Delete ServerInLan ALG function IP mapping.
Defined in: rtk_rg_liteRomeDriver.h
Parameters delServerMapping
[in] Delete the server IP address mapping.
Comments None
Return Codes RT_ERR_RG_OK
RT_ERR_RG_INVALID_PARAM the input parameters may be NULL

12.20. rtk_rg_algApps_set

int32 rtk_rg_algApps_set(rtk_rg_alg_type_t *alg_app***)**
Set ALG functions by bitmask.
Defined in: rtk_rg_liteRomeDriver.h
Parameters alg_app
[in] The bitmask setting for all ALG functions.
Comments Althrough the bitmask list all ALG functions here, the implemented functions depend on romeDriver's version. Please refer user document.
Return Codes RT_ERR_RG_OK
RT_ERR_RG_ALG_SRV_IN_LAN_NO_IP before turn on Server In Lan services, the server ip has to be added by rtk_rg_algServerInLanAppsIpAddr_add

12.21. rtk_rg_algApps_get

int32 rtk_rg_algApps_get(rtk_rg_alg_type_t **alg_app***)**
Get ALG functions by bitmask.
Defined in: rtk_rg_liteRomeDriver.h
Parameters *alg_app
[out] Return the bitmask setting for ALG functions.
Comments Althrough the bitmask list all ALG functions here, the implemented functions depend on romeDriver's version. Please refer user document.
Return Codes RT_ERR_RG_OK

12.22. rtk_rg_dmzHost_set

int32 rtk_rg_dmzHost_set(int *wan_intf_idx*, **rtk_rg_dmzInfo_t** **dmz_info*)
Add DMZ connection rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
wan_intf_idx
[in] the DMZ enabled wan interface index.
*dmz_info
[in] the DMZ setting.

Comments
dmz_info->enabled - Enable/disable DMZ rule.
dmz_info->mac_mapping_enabled - The DMZ rule is using MAC mapping or not.
dmz_info->private_ip - The redirected DMZ internal host IP.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.23. rtk_rg_dmzHost_get

int32 rtk_rg_dmzHost_get(int *wan_intf_idx*, **rtk_rg_dmzInfo_t** **dmz_info*)
Get configured DMZ rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
wan_intf_idx
[in] the DMZ enabled wan interface index.
*dmz_info
[in] the DMZ rule setting.

Comments
dmz_info->enabled - Enable/disable DMZ rule.
dmz_info->mac_mapping_enabled - The DMZ rule is using MAC mapping or not.
dmz_info->private_ip - The redirected DMZ internal host IP.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.24. rtk_rg_virtualServer_add

int32 rtk_rg_virtualServer_add(rtk_rg_virtualServer_t **virtual_server*, **int** **virtual_server_idx*)
Add virtual server connection rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*virtual_server
[in] the svirtual server data structure.
*virtual_server_idx
[in] the index of virtual server table.

Comments
virtual_server.is_tcp - Layer 4 protocol is TCP or not.
virtual_server.wan_intf_idx - Wan interface index of server.
virtual_server.gateway_port_start - Gateway external port mapping start number.

virtual_server.local_ip - The translating local IP address.
virtual_server.local_port_start - The translating internal port mapping start number.
virtual_server.mappingRangeCnt - The port mapping range count.
virtual_server.valid - This entry is valid or not.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INITPM_UNINIT	the virtual server table uninitialized.
RT_ERR_RG_ENTRY_FULL	the virtual server table is full.

12.25. rtk_rg_virtualServer_del

int32 rtk_rg_virtualServer_del(int *virtual_server_idx***)**
Delete one server port connection rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
virtual_server_idx
[in] the index of virtual server entry for deleting.

Comments
None.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	illegal server port rule index.

12.26. rtk_rg_virtualServer_find

int32 rtk_rg_virtualServer_find(rtk_rg_virtualServer_t **virtual_server***, int** **valid_idx***)**
Find entire virtual server table from valid_idx till valid one.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*virtual_server
[in] An empty buffer for storing the virtual server entry data structure.
[out] The data structure of found virtual sint
*valid_idx
[in] The index which find from.
[out] The existing entry index.

Comments
virtual_server.is_tcp - Layer 4 protocol is TCP or not.
virtual_server.wan_intf_idx - Wan interface index of server.
virtual_server.gateway_port_start - Gateway external port mapping start number.
virtual_server.local_ip - The translating local IP address.
virtual_server.local_port_start - The translating internal port mapping start number.
virtual_server.mappingRangeCnt - The port mapping range count.
virtual_server.valid - This entry is valid or not.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INITPM_UNINIT	the virtual server table uninitialized.
RT_ERR_RG_SVRPORT_SW_ENTRY_NOT_FOUND	can't find entry in virtual server table.

12.27. rtk_rg_aclFilterAndQos_add

int32 rtk_rg_aclFilterAndQos_add(rtk_rg_aclFilterAndQos_t **acl_filter***, int** **acl_filter_idx***)**

Add acl rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters

*acl_filter
[in] assign the patterns which need to be filtered, and assign the related action(include drop, permit, Qos, and Trap to CPUint
*acl_filter_idx
[out] the index of the added acl rule.

Comments

acl_filter.filter_fields - use to enable the filtered patterns. Each pattern bit should be "or" together.
acl_filter.ingress_port_mask - assign the packet ingress physical port pattern(valid when INGRESS_PORT_BIT is enable)
acl_filter.ingress_dscp - assign the packet ingress dscp(valid when INGRESS_DSCP_BIT is enable)
acl_filter.ingress_intf_idx - assign the packet ingress interface index(valid when INGRESS_INTF_BIT is enable)
acl_filter.egress_intf_idx - assign the packet egress interface index(valid when EGRESS_INTF_BIT is enable)
acl_filter.ingress_ethertype - assign the packet ingress ethertype pattern(valid when INGRESS_ETHERTYPE_BIT is enable)
acl_filter.ingress_ctag_vid - assign the packet ingress vlan id pattern(valid when INGRESS_CTAG_VID_BIT is enable)
acl_filter.ingress_ctag_pri - assign the packet ingress vlan priority pattern(valid when INGRESS_CTAG_PRI_BIT is enable)
acl_filter.ingress_smac - assign the packet ingress source mac pattern(valid when INGRESS_SMAC_BIT is enable)
acl_filter.ingress_dmac - assign the packet ingress destination mac pattern(valid when INGRESS_DMAC_BIT is enable)
acl_filter.ingress_src_ipv4_addr_start - assign the packet ingress source ipv4 lower bound(valid when INGRESS_IPV4_SIP_RANGE_BIT is enable)
acl_filter.ingress_src_ipv4_addr_end - assign the packet ingress source ipv4 upper bound(valid when INGRESS_IPV4_SIP_RANGE_BIT is enable)
acl_filter.ingress_dest_ipv4_addr_start - assign the packet ingress destination ipv4 lower bound(valid when INGRESS_IPV4_DIP_RANGE_BIT is enable)
acl_filter.ingress_dest_ipv4_addr_end - assign the packet ingress destination ipv4 upper bound(valid when INGRESS_IPV4_DIP_RANGE_BIT is enable)
acl_filter.ingress_src_ipv6_addr_start - assign the packet ingress source ipv6 lower bound(valid when INGRESS_IPV6_SIP_RANGE_BIT is enable)
acl_filter.ingress_src_ipv6_addr_end - assign the packet ingress source ipv6 upper bound(valid when INGRESS_IPV6_SIP_RANGE_BIT is enable)
acl_filter.ingress_dest_ipv6_addr_start - assign the packet ingress destination ipv6 lower bound(valid when INGRESS_IPV6_DIP_RANGE_BIT is enable)
acl_filter.ingress_dest_ipv6_addr_end - assign the packet ingress destination ipv6 upper bound(valid when INGRESS_IPV6_DIP_RANGE_BIT is enable)
acl_filter.ingress_src_l4_port_start - assign the packet ingress layer4 source port lower bound(valid when INGRESS_L4_SPORT_RANGE_BIT is enable)
acl_filter.ingress_src_l4_port_end - assign the packet ingress layer4 source port upper bound(valid when INGRESS_L4_SPORT_RANGE_BIT is enable)
acl_filter.ingress_dest_l4_port_start - assign the packet ingress layer4 destination port lower bound(valid when INGRESS_L4_DPORT_RANGE_BIT is enable)
acl_filter.ingress_dest_l4_port_end - assign the packet ingress layer4 destination port upper bound(valid when INGRESS_L4_DPORT_RANGE_BIT is enable)
acl_filter.egress_src_ipv4_addr_start - assign the packet egress source ipv4 lower bound(valid when EGRESS_IPV4_SIP_RANGE_BIT is enable)
acl_filter.egress_src_ipv4_addr_end - assign the packet egress source ipv4 upper bound(valid when EGRESS_IPV4_SIP_RANGE_BIT is enable)
acl_filter.egress_dest_ipv4_addr_start - assign the packet egress destination ipv4 lower

bound(valid when EGRESS_IPV4_DIP_RANGE_BIT is enable)
acl_filter.egress_dest_ipv4_addr_end - assign the packet egress destination ipv4 pattern(upper bound)(valid when EGRESS_IPV4_DIP_RANGE_BIT is enable)
acl_filter.egress_src_l4_port_start - assign the packet egress layer4 source port lower bound(valid when EGRESS_L4_SPORT_RANGE_BIT is enable)
acl_filter.egress_src_l4_port_end - assign the packet egress layer4 source port upper bound(valid when EGRESS_L4_SPORT_RANGE_BIT is enable)
acl_filter.egress_dest_l4_port_start - assign the packet egress layer4 destination port lower bound(valid when EGRESS_L4_DPORT_RANGE_BIT is enable)
acl_filter.egress_dest_l4_port_end - assign the packet egress layer4 destination port upper bound(valid when EGRESS_L4_DPORT_RANGE_BIT is enable)
acl_filter.action_type - assign the action to the packets which satisfy the assgned patterns
acl_filter.qos_actions - assign the qos action. Each action bit should be "or" together (triggered while action_type==ACL_ACTION_TYPE_QOS)
acl_filter.action_dot1p_remarking_pri - assign the vlan priority value for remarking(vlaid when while ACL_ACTION_1P_REMARKING_BIT is enable)
acl_filter.action_ip_precedence_remarking_pri - assign the ip precedence value for remarking(vlaid when ACL_ACTION_IP_PRECEDENCE_REMARKING_BIT is enable)
acl_filter.action_dscp_remarking_pri - assign the dscp value for remarking(vlaid when ACL_ACTION_DSCP_REMARKING_BIT is enable)
acl_filter.action_queue_id - assign the qid(vlaid when ACL_ACTION_QUEUE_ID_BIT is enable)
acl_filter.action_share_meter - assign the sharemeter(vlaid when ACL_ACTION_SHARE_METER_BIT is enable)

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ACL_CF_FIELD_CONFLICT	acl_filter assigned some conflict patterns
RT_ERR_RG_ACL_CF_FLOW_DIRECTION_ERROR	the assigned ingress interface and egress interface are not upstream or downstream
RT_ERR_RG_ACL_ENTRY_FULL	the hardware ACL asic entry is full
RT_ERR_RG_ACL_ENTRY_ACCESS_FAILED	access the hardware ACL asic entry failed
RT_ERR_RG_ACL_IPTABLE_FULL	the hardware asic ACL IP_RANGE_TABLE entry is full
RT_ERR_RG_ACL_IPTABLE_ACCESS_FAILED	access the hardware asic ACL IP_RANGE_TABLE entry failed
RT_ERR_RG_ACL_PORTTABLE_FULL	the hardware asic ACL PORT_RANGE_TABLE entry is full
RT_ERR_RG_ACL_PORTTABLE_ACCESS_FAILED	access the hardware ACL asic PORT_RANGE_TABLE entry failed
RT_ERR_RG_CF_ENTRY_FULL	the hardware Classification asic entry is full
RT_ERR_RG_CF_ENTRY_ACCESS_FAILED	access the hardware Classification asic entry failed
RT_ERR_RG_CF_IPTABLE_FULL	the hardware Classification asic IP_RANGE_TABLE entry is full
RT_ERR_RG_CF_IPTABLE_ACCESS_FAILED	access the hardware Classification asic IP_RANGE_TABLE entry failed
RT_ERR_RG_CF_PORTTABLE_FULL	the hardware Classification PORT_RANGE_TABLE asic entry is full
RT_ERR_RG_CF_PORTTABLE_ACCESS_FAILED	access the hardware Classification asic PORT_RANGE_TABLE entry failed
RT_ERR_RG_CF_DSCPTABLE_FULL	the hardware Classification DSCP_REMARKING_TABLE asic entry is full
RT_ERR_RG_CF_DSCPTABLE_ACCESS_FAILED	access the hardware Classification asic DSCP_REMARKING_TABLE entry failed
RT_ERR_RG_ACL_SW_ENTRY_FUL	the software ACL entry is full

L	
RT_ERR_RG_ACL_SW_ENTRY_ACC ESS_FAILED	access the software ACL entry failed

12.28. rtk_rg_aclFilterAndQos_del

int32 rtk_rg_aclFilterAndQos_del(int *acl_filter_idx***)**

Delete acl rule.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

acl_filter_idx

[in] the index of the acl rule which need to be delete

Comments

None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ACL_ENTRY_ACCESS _FAILED	access the hardware ACL asic entry failed
RT_ERR_RG_ACL_IPTABLE_ACCES S_FAILED	access the hardware asic ACL IP_RANGE_TABLE entry failed
RT_ERR_RG_ACL_PORTTABLE_AC CESS_FAILED	access the hardware ACL asic PORT_RANGE_TABLE entry failed
RT_ERR_RG_CF_ENTRY_ACCESS_F AILED	access the hardware Classification asic entry failed
RT_ERR_RG_CF_IPTABLE_ACCESS _FAILED	access the hardware Classification asic IP_RANGE_TABLE entry failed
RT_ERR_RG_CF_PORTTABLE_ACC ESS_FAILED	access the hardware Classification asic PORT_RANGE_TABLE entry failed
RT_ERR_RG_CF_DSCPTABLE_ACC ESS_FAILED	access the hardware Classification asic DSCP_REMARKING_TABLE entry failed
RT_ERR_RG_ACL_SW_ENTRY_ACC ESS_FAILED	access the software ACL entry failed

12.29. rtk_rg_aclFilterAndQos_find

int32 rtk_rg_aclFilterAndQos_find(rtk_rg_aclFilterAndQos_t **acl_filter***, **int** **valid_idx***)**

fine acl rule.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*acl_filter

the index of the acl rule which start to search

*valid_idx

[out] the acl rule which be found

Comments

this API search the software acl entry start from acl_filter_idx, and find the first exist acl entry. If all entry after the acl_filter_idx are empty, it will return RT_ERR_RG_ACL_SW_ENTRY_NOT_FOUND

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range
RT_ERR_RG_ACL_SW_ENTRY_ACC ESS_FAILED	access the software ACL entry failed

RT_ERR_RG_ACL_SW_ENTRY_NOT_FOUND	can not find the assigned ACL entry

12.30. rtk_rg_macFilter_add

int32 rtk_rg_macFilter_add(rtk_rg_macFilterEntry_t **macFilterEntry*, **int** **mac_filter_idx*)

add mac filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*macFilterEntry

[in] the mac address which need to be add to mac filter rule.

*mac_filter_idx

[in] the mac address which shouuld be filter in SMAC, DMAC, or BOTH.

Comments None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.31. rtk_rg_macFilter_del

int32 rtk_rg_macFilter_del(int *mac_filter_idx*)

delete mac filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

mac_filter_idx

[in] the index of the mac filter rule which need to be deleted.

Comments None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.32. rtk_rg_macFilter_find

int32 rtk_rg_macFilter_find(rtk_rg_macFilterEntry_t **macFilterEntry*, **int** **valid_idx*)

find mac filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*macFilterEntry

[out] the mac address which be found.

*valid_idx

[out] filter for SMAC, DMAC, or BOTH.

Comments None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL

RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.33. rtk_rg_urlFilterString_add

int32 rtk_rg_urlFilterString_add(rtk_rg_urlFilterString_t **filter*, **int** **url_idx*)

add url filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*filter

[in] the url rule which need to be added to url filter rule.

*url_idx

[out] the index of added url filter rule.

Comments

filter.url_filter_string - the string which need to be filtered in url string.(ex: in url "http://www.sample.com/explain", "www.sample.com" is the url string)

filter.path_filter_string - the string which need to be filtered in url path string.(ex: in url "http://www.sample.com/explain", "/explain" is the url path string)

filter.path_exactly_match - the urlFilter will execute even the path_filter_string is part of url path string while path_exactly_match is 0. Else, the path_filter_string must exactly match the url path string to trigger urlFilter execution.

filter.wan_intf - the index of the wan interface which should limited by this urlFilter.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.34. rtk_rg_urlFilterString_del

int32 rtk_rg_urlFilterString_del(int *url_idx*)

delete url filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

url_idx

[in] the index of the url filter rule which need to be deleted.

Comments

None

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.35. rtk_rg_urlFilterString_find

int32 rtk_rg_urlFilterString_find(rtk_rg_urlFilterString_t **filter*, **int** **valid_idx*)

find url filter rule

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*filter

[out] the url filter rule which be found.

*valid_idx
[in] the index fo url filter rule which start to search.
[out] the index of found url filter rule.

Comments filter.url_filter_string - the string which need to be filtered in url string.(ex: in url "http://www.sample.com/explain", "www.sample.com" is the url string)
filter.path_filter_string - the string which need to be filtered in url path string.(ex: in url "http://www.sample.com/explain", "/explain" is the url path string)
filter.path_exactly_match - the urlFilter will execute even the path_filter_string is part of url path string while path_exactly_match is 0. Else, the path_filter_string must exactly match the url path string to trigger urlFilter execution.
filter.wan_intf - the index of the wan interface which should be limited by this urlFilter.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information or range

12.36. rtk_rg_upnpConnection_add

int32 rtk_rg_upnpConnection_add(rtk_rg_upnpConnection_t **upnp*, **int** **upnp_idx*)
Add UPNP connection rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters *upnp
[in] the UPNP connection data structure.
*upnp_idx
[in] the index of UPNP table.

Comments upnp.is_tcp - Layer 4 protocol is TCP or not.
upnp.valid - The UNPN mapping is valid or not.
upnp.wan_intf_idx - Wan interface index.
upnp.gateway_port - Gateway external port number.
upnp.local_ip - Internal ip address.
upnp.local_port - Internal port number.
upnp.limit_remote_ip - The Restricted remote IP address.
upnp.limit_remote_port - The Restricted remote port number.
upnp.remote_ip - Remote IP address. upnp.type - One shot or persist.
upnp.timeout - timeout value for auto-delete. Set 0 to disable auto-delete.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	the rg has not been init. (rtk_rg_initParam_set() should be called first)
RT_ERR_RG_INITPM_UNINIT	the UPNP table uninitialized.
RT_ERR_RG_ENTRY_FULL	the UPNP mapping table is full.

12.37. rtk_rg_upnpConnection_del

int32 rtk_rg_upnpConnection_del(int *upnp_idx*)
Delete UPNP connection rule.
Defined in: rtk_rg_liteRomeDriver.h

Parameters upnp_idx
[in] the index of UPNP table entry to be deleted.

Comments None.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INVALID_PARAM	illegal UPNP rule index.

12.38. rtk_rg_upnpConnection_find

int32 rtk_rg_upnpConnection_find(rtk_rg_upnpConnection_t **upnp*, int **valid_idx*)

Find entire UPNP mapping table from upnp_idx till valid one.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*upnp

[in] An empty buffer for storing the UPNP mapping entry data structure.

[out] The data structure of found UPNP mappinint

*valid_idx

[in] The index which find from.

[out] The existing entry index.

Comments

upnp.is_tcp - Layer 4 protocol is TCP or not.
upnp.valid - The UNPN mapping is valid or not.
upnp.wan_intf_idx - Wan interface index.
upnp.gateway_port - Gateway external port number.
upnp.local_ip - Internal ip address.
upnp.local_port - Internal port number.
upnp.limit_remote_ip - The Restricted remote IP address.
upnp.limit_remote_port - The Restricted remote port number.
upnp.remote_ip - Remote IP address. upnp.type - One shot or persist.
upnp.timeout - timeout value for auto-delete. Set 0 to disable auto-delete. The condition fields will be ignore while it was set to zero.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_INITPM_UNINIT	the UPNP table uninitialized.
RT_ERR_RG_NULL_POINTER	the input parameters may be NULL
RT_ERR_RG_UPNP_SW_ENTRY_NOT_FOUND	can't find entry in UPNP mapping table.

12.39. rtk_rg_naptConnection_add

int32 rtk_rg_naptConnection_add(rtk_rg_naptEntry_t **naptFlow*, int **flow_idx*)

Add NAPT connection flow.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*naptFlow

[in] the NAPT connection flow data structure.

*flow_idx

[out] the NAPT flow index.

Comments

naptFlow.is_tcp - Layer 4 protocol is TCP or not.
naptFlow.local_ip - Internal IP address.
naptFlow.remote_ip - Remote IP address.
naptFlow.wan_intf_idx - Wan interface index.
naptFlow.local_port - Internal port number.
naptFlow.remote_port - Remote port number.
naptFlow.external_port - Gateway external port number.
naptFlow.pri_valid - NAPT priority remarking enable.
naptFlow.priority - NAPT priority remarking value.

Return Codes

RT_ERR_RG_OK

RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INVALID_PARAM	illegal NAPT connection flow index.
RT_ERR_RG_EXTIP_GET_FAIL	illegal wan_intf_idx.
RT_ERR_RG_NAPT_FLOW_DUPLICATE	duplicate NAPT flow.
RT_ERR_RG_NAPTR_OVERFLOW	NAPTR table collision.
RT_ERR_RG_NAPT_OVERFLOW	NAPT table collision.
RT_ERR_RG_NAPTR_SET_FAIL	write to NAPTR table failed.
RT_ERR_RG_NAPT_SET_FAIL	write to NAPT table failed.

12.40. rtk_rg_naptConnection_del

int32 rtk_rg_naptConnection_del(int *flow_idx***)**
Delete NAPT connection flow.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
flow_idx
[in] The index of NAPT connection flow table.

Comments
None.

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INVALID_PARAM	illegal NAPT connection flow index.
RT_ERR_RG_NAPT_SW_ENTRY_NOT_FOUND	the NAPT entry can not be found.
RT_ERR_RG_NAPTR_SET_FAIL	write to NAPTR table failed.
RT_ERR_RG_NAPT_SET_FAIL	write to NAPT table failed.

12.41. rtk_rg_naptConnection_find

int32 rtk_rg_naptConnection_find(rtk_rg_naptInfo_t ***naptInfo*****, int** ***valid_idx*****)**
Find entire NAPT table from index valid_idx till valid one.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*naptInfo
[in] An empty buffer for storing the NAPT entry data structure.
[out] The data structure of found NAPT entry.
*valid_idx
[in] The index which find from.
[out] The existing entry index.

Comments
naptInfo.is_tcp - Layer 4 protocol is TCP or not.
naptInfo.local_ip - Internal IP address.
naptInfo.remote_ip - Remote IP address.
naptInfo.wan_intf_idx - Wan interface index.
naptInfo.local_port - Internal port number.
naptInfo.remote_port - Remote port number.
naptInfo.external_port - Gateway external port number.
naptInfo.pri_valid - NAPT priority remarking enable.
naptInfo.priority - NAPT priority remarking value.
naptInfo.idleSecs - NAPT flow idle seconds.
naptInfo.state - NAPT connection state, 0:INVALID 1:SYN_RECV 2:SYN_ACK_RECV 3:CONNECTED 4:FIN_RECV 5:RST_RECV

Return Codes

RT_ERR_RG_OK	
RT_ERR_RG_NOT_INIT	system is not initiated.

RT_ERR_RG_INVALID_PARAM	illegal NAPT connection flow index.
RT_ERR_RG_NAPT_SW_ENTRY_NOT_FOUND	the NAPT entry can not be found.

12.42. rtk_rg_multicastFlow_add

int32 rtk_rg_multicastFlow_add(rtk_rg_multicastFlow_t **mcFlow*, **int** **flow_idx*)

Add a Multicast flow entry into ASIC.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*mcFlow

[in] Multicast flow entry content structure.

*flow_idx

[out] Returned Multicast flow index.

Comments

mcFlow.multicast_ipv4_addr - The destination IP address of packet, it must be class D address.
mcFlow.port_mask - This MC packet will forward to those MAC ports.
mcFlow.ext_port_mask - This MC packet will forward to those Extension ports.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	System is not initiated.
RT_ERR_RG_INVALID_PARAM	The params is not accpeted.
RT_ERR_RG_ENTRY_FULL	The MC entry is full.
RT_ERR_FAILED	failed
RT_ERR_NOT_INIT	The module is not initial
RT_ERR_IPV4_ADDRESS	Invalid IPv4 address
RT_ERR_VLAN_VID	invalid vlan id
RT_ERR_NULL_POINTER	input parameter may be null pointer
RT_ERR_INPUT	invalid input parameter
	access the software ACL entry failed

12.43. rtk_rg_multicastFlow_del

int32 rtk_rg_multicastFlow_del(int *flow_idx*)

Delete an Multicast flow ASIC entry.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

flow_idx

[in] The Multicast flow entry index for deleting.

Comments

None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_ENTRY_NOT_EXIST	the index is not exist.
RT_ERR_FAILED	failed
RT_ERR_L2_EMPTY_ENTRY	Empty LUT entry.
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_L2_HASH_KEY	invalid L2 Hash key
RT_ERR_L2_EMPTY_ENTRY	the entry is empty(invalid)

12.44. rtk_rg_multicastFlow_find

int32 rtk_rg_multicastFlow_find(rtk_rg_multicastFlow_t **mcFlow*, **int** **valid_idx*)

Find an exist Multicast flow ASIC entry.
Defined in: rtk_rg_liteRomeDriver.h

Parameters

*mcFlow
[in] An empty buffer for storing the structure.
[out] An exist entry structure which index is valid_idx.
*valid_idx
[in] The index which find from.
[out] The exist entry index.

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_NO_MORE_ENTRY_FOUND	no more exist entry is found.
RT_ERR_FAILED	failed
RT_ERR_L2_EMPTY_ENTRY	Empty LUT entry.
RT_ERR_INPUT	Invalid input parameters.

12.45. rtk_rg_macEntry_add

int32 rtk_rg_macEntry_add(rtk_rg_macEntry_t **macEntry*, **int** **entry_idx*)
Add a MAC Entry into ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters

*macEntry
[in] MAC entry content structure.
*entry_idx
[out] this MAC entry index.

Comments

macEntry.mac - The MAC address.
macEntry.isIVL - The MAC is for IVL or SVL.
macEntry.fid - The fid, only used in SVL.
macEntry.vlan_id - The VLAN id.
macEntry.port_idx - The port index,0~5 for phy. port, 6 for CPU port, 7~11 for ext. port.
macEntry.arp_used - The entry is used for NAT/NAPT.
macEntry.static_entry - The MAC is static or not.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.

12.46. rtk_rg_macEntry_del

int32 rtk_rg_macEntry_del(int *entry_idx*)
Delete an ASIC MAC Entry.
Defined in: rtk_rg_liteRomeDriver.h

Parameters

entry_idx
[in] The MAC entry index for deleting.

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_ENTRY_NOT_EXIST	the index is not exist.

12.47. rtk_rg_macEntry_find

int32 rtk_rg_macEntry_find(rtk_rg_macEntry_t *_macEntry_, int *_valid_idx_)
Find an exist ASIC MAC Entry.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*macEntry
[in] An empty buffer for storing the MAC entry data structure.
[out] The data structure of found MAC entry.
*valid_idx
[in] The index which find from.
[out] The existing entry index.

Comments
macEntry.mac - The MAC address.
macEntry.isIVL - The MAC is for IVL or SVL.
macEntry.fid - The fid, only used in SVL.
macEntry.vlan_id - The VLAN id.
macEntry.port_idx - The port index,0~5 for phy. port, 6 for CPU port, 7~11 for ext. port.
macEntry.arp_used - The entry is used for NAT/NAPT.
macEntry.static_entry - The MAC is static or not.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_NO_MORE_ENTRY_FOUND	no more exist entry is found.
RT_ERR_RG_NULL_POINTER	Input parameter is NULL pointer.

12.48. rtk_rg_arpEntry_add

int32 rtk_rg_arpEntry_add(rtk_rg_arpEntry_t *_arpEntry_, int *_arp_entry_idx_)
Add an ARP Entry into ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*arpEntry
[in] Fill rtk_rg_arpEntry_t each fields for adding.
*arp_entry_idx
[out] Return ARP entry index.

Comments
arpEntry.macEntryIdx - this MAC entry index of this ARP entry.
arpEntry.ipv4Addr - the IPv4 IP address of this ARP entry.
arpEntry.static_entry - this entry is static ARP, and it will never age out.
If the arpEntry.static_entry is set. The MAC entry must set static, too.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.
RT_ERR_RG_L2_ENTRY_NOT_FOUND	L2 entry not found.

12.49. rtk_rg_arpEntry_del

int32 rtk_rg_arpEntry_del(int *arp_entry_idx***)**
Delete an ASIC ARP Entry.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
arp_entry_idx
[out] The ARP entry index for deleting.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_ENTRY_NOT_EXIST	the index is not exist.

12.50. rtk_rg_arpEntry_find

int32 rtk_rg_arpEntry_find(rtk_rg_arpInfo_t **arpInfo***, int** **arp_valid_idx***)**
Find the entire ASIC ARP table from index arp_valid_idx till valid one.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*arpInfo
[in] An empty buffer for storing the ARP entry data structure.
[out] The data structure of found ARP entry.
*arp_valid_idx
[in] The index which find from.
[out] The existing entry index.

Comments
arpInfo.arpEntry - The ARP entry data structure.
arpInfo.valid - The ARP entry is valid or not.
arpEntry.idleSecs - The ARP entry idle time in seconds.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_NULL_POINTER	input buffer pointer is NULL.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range
RT_ERR_RG_NO_MORE_ENTRY_FOUND	no more exist entry is found.

12.51. rtk_rg_neighborEntry_add

int32 rtk_rg_neighborEntry_add(rtk_rg_neighborEntry_t **neighborEntry***, int** **neighbor_idx***)**
Add an Neighbor Entry into ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*neighborEntry
[in] Fill rtk_rg_neighborEntry_t each fields for adding.
*neighbor_idx
[out] Return Neighbor entry index.

Comments
neighborEntry.l2Idx - this MAC entry index of this Neighbor entry.
neighborEntry.matchRouteIdx - the routing entry idx that match the ip address.

neighborEntry.interfaceId - the 64 bits interface identifier of this IPv6 address.
neighborEntry.staticEntry - this entry is static ARP, and it will never age out.
neighborEntry.valid - this entry is valid.
If the neighborEntry.static_entry is set. The MAC entry must set static, too.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.
RT_ERR_RG_L2_ENTRY_NOT_FOUND	L2 entry not found.
RT_ERR_RG_NEIGHBOR_FULL	neighbor table is full

12.52. rtk_rg_neighborEntry_del

int32 rtk_rg_neighborEntry_del(int *neighbor_idx***)**
Delete an ASIC Neighbor Entry.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
neighbor_idx
[out] The Neighbor entry index for deleting.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range.
RT_ERR_RG_ENTRY_NOT_EXIST	the index is not exist.

12.53. rtk_rg_neighborEntry_find

int32 rtk_rg_neighborEntry_find(rtk_rg_neighborInfo_t **neighborInfo***, int** **neighbor_valid_idx***)**
Find the entire ASIC Neighbor table from index neighbor_valid_idx till valid one.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*neighborInfo
[in] An empty buffer for storing the Neighbor entry data structure.
[out] The data structure of found Neighbor entry.int
*neighbor_valid_idx
[in] The index which find from.
[out] The existing entry index.

Comments
neighborInfo.arpEntry - The Neighbor entry data structure.
neighborInfo.idleSecs - The Neighbor entry idle time in seconds.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NOT_INIT	system is not initiated.
RT_ERR_RG_NULL_POINTER	input buffer pointer is NULL.
RT_ERR_RG_INDEX_OUT_OF_RANGE	the index out of range
RT_ERR_RG_NO_MORE_ENTRY_FOUND	no more exist entry is found.
RT_ERR_RG_NEIGHBOR_NOT_FOUND	the indicated neighbor ifid is not found.

12.54. rtk_rg_softwareSourceAddrLearningLimit_set

int32 rtk_rg_softwareSourceAddrLearningLimit_set(rtk_rg_saLearningLimitInfo_t *sa_learnLimit_info*, **rtk_rg_port_idx_t** *port_idx*)
Set source address learning limit and action when using software learning.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
sa_learnLimit_info
[in] The information entered for the dedicated port.
port_idx
[in] The port number to set source address learning information.

Comments
sa_learnLimit_info.learningLimitNumber - the maximum number can be learned in the port.
sa_learnLimit_info.action - what to do if a packet is exceed the learning limit.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	system is not initiated.

12.55. rtk_rg_softwareSourceAddrLearningLimit_get

int32 rtk_rg_softwareSourceAddrLearningLimit_get(rtk_rg_saLearningLimitInfo_t **sa_learnLimit_info*, **rtk_rg_port_idx_t** *port_idx*)
Get source address learning limit and action when using software learning by port number.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*sa_learnLimit_info
[out] The information contained of the dedicated port.
port_idx
[in] The port number to get source address learning information.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	system is not initiated.

12.56. rtk_rg_dosPortMaskEnable_set

int32 rtk_rg_dosPortMaskEnable_set(rtk_rg_mac_portmask_t *dos_port_mask*)
Enable/Disable denial of service security port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dos_port_mask
[in] Security MAC port mask.

Comments
None

Return Codes

RT_ERR_RG_OK	Success

12.57. rtk_rg_dosPortMaskEnable_get

int32 rtk_rg_dosPortMaskEnable_get(rtk_rg_mac_portmask_t **dos_port_mask*)
Get denial of service port security port state.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*dos_port_mask
[out] Security MAC port mask.

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input port mask pointer is NULL.

12.58. rtk_rg_dosType_set

int32 rtk_rg_dosType_set(rtk_rg_dos_type_t *dos_type*, **int** *dos_enabled*,
rtk_rg_dos_action_t *dos_action*)
Set denial of service type function.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dos_type
[in] Port security type.
dos_enabled
[in] Port security function enabled/disabled.
dos_action
[in] Port security action.

Comments dos_type.RTK_RG_DAEQSA_DENY - packets while SMAC is the same as DMAC. dos_type.RTK_RG_LAND_DENY - packets while SIP is the same as DIP(support IPv4 only). dos_type.RTK_RG_BLAT_DENY - packets while the TCP/UDP SPORT is the same as DPORT destination TCP/UDP port. dos_type.RTK_RG_SYNFIN_DENY - TCP packets while SYN and FIN bits are set. dos_type.RTK_RG_XMA_DENY - TCP packets while sequence number is zero and FIN,URG,PSH bits are set. dos_type.RTK_RG_NULLSCAN_DENY - TCP packets while sequence number is zero and all contorl bits are zeros. dos_type.RTK_RG_SYN_SPORTL1024_DENY - TCP SYN packets with source port less than 1024. dos_type.RTK_RG_TCPHDR_MIN_CHECK - the length of a TCP header carried in an unfragmented IP(IPv4 and IPv6) datagram or the first fragment of a fragmented IP(IPv4) datagram is less than MIN_TCP_Header_Size(20 bytes). dos_type.RTK_RG_TCP_FRAG_OFF_MIN_CHECK - the Frangment_Offset=1 in anyfragment of a fragmented IP datagram carrying part of TCP data. dos_type.RTK_RG_ICMP_FRAG_PKTS_DENY - ICMPv4/ICMPv6 data unit carried in a fragmented IP datagram. dos_type.RTK_RG_POD_DENY - IP packet size > 65535 bytes, ((IP offset *8) + (IP length) ¡V (IPIHL*4))>65535. dos_type.RTK_RG_UDPDOMB_DENY - UDP length > IP payload length. dos_type.RTK_RG_SYNWITHDATA_DENY - 1. IP length > IP header + TCP header length while SYN flag is set 1. 2. IP More Fragment and Offset > 0 while SYN is set to 1. dos_action.RTK_RG_DOS_ACTION_DROP - Drop packet while hit DoS criteria. dos_action.RTK_RG_DOS_ACTION_TRAP - Trap packet to CPU while hit DoS criteria.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.

12.59. rtk_rg_dosType_get

int32 rtk_rg_dosType_get(rtk_rg_dos_type_t *dos_type*, **int** **dos_enabled*, **rtk_rg_dos_action_t** **dos_action*)
Get denial of service type function.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dos_type
[in] Port security type.
*dos_enabled
[out] Port security function enabled/disabled.
*dos_action
[out] Port security action.

Comments
dos_type.RTK_RG_DAEQSA_DENY - packets while SMAC is the same as DMAC. dos_type.RTK_RG_LAND_DENY - packets while SIP is the same as DIP(support IPv4 only). dos_type.RTK_RG_BLAT_DENY - packets while the TCP/UDP SPORT is the same as DPORT destination TCP/UDP port. dos_type.RTK_RG_SYNFIN_DENY - TCP packets while SYN and FIN bits are set. dos_type.RTK_RG_XMA_DENY - TCP packets while sequence number is zero and FIN,URG,PSH bits are set. dos_type.RTK_RG_NULLSCAN_DENY - TCP packets while sequence number is zero and all contorl bits are zeros. dos_type.RTK_RG_SYN_SPORTL1024_DENY - TCP SYN packets with source port less than 1024. dos_type.RTK_RG_TCPHDR_MIN_CHECK - the length of a TCP header carried in an unfragmented IP(IPv4 and IPv6) datagram or the first fragment of a fragmented IP(IPv4) datagram is less than MIN_TCP_Header_Size(20 bytes). dos_type.RTK_RG_TCP_FRAG_OFF_MIN_CHECK - the Frangment_Offset=1 in anyfragment of a fragmented IP datagram carrying part of TCP data. dos_type.RTK_RG_ICMP_FRAG_PKTS_DENY - ICMPv4/ICMPv6 data unit carried in a fragmented IP datagram. dos_type.RTK_RG_POD_DENY - IP packet size > 65535 bytes, ((IP offset *8) + (IP length) ¡V (IPIHL*4))>65535. dos_type.RTK_RG_UDPDOMB_DENY - UDP length > IP payload length. dos_type.RTK_RG_SYNWITHDATA_DENY - 1. IP length > IP header + TCP header length while SYN flag is set 1. 2. IP More Fragment and Offset > 0 while SYN is set to 1. dos_action.RTK_RG_DOS_ACTION_DROP - Drop packet while hit DoS criteria. dos_action.RTK_RG_DOS_ACTION_TRAP - Trap packet to CPU while hit DoS criteria.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input port mask pointer is NULL.

12.60. rtk_rg_dosFloodType_set

int32 rtk_rg_dosFloodType_set(rtk_rg_dos_type_t *dos_type*, **int** *dos_enabled*, **rtk_rg_dos_action_t** *dos_action*, **int** *dos_threshold*)
Set denial of service flooding attack protection function.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dos_type
[in] Port security type.
dos_enabled
[in] Port security function enabled/disabled.
dos_action
[in] Port security action.
dos_threshold
[in] System

Comments
dos_type.RTK_RG_SYNFLOOD_DENY - Receiving TCP SYN packet number is over

threshold. dos_type.RTK_RG_FINFLOOD_DENY - Receiving TCP FIN packet number is over threshold. dos_type.RTK_RG_ICMPFLOOD_DENY - Receiving ICMP packet number is over threshold. dos_action.RTK_RG_DOS_ACTION_DROP - Drop packet while hit DoS criteria. dos_action.RTK_RG_DOS_ACTION_TRAP - Trap packet to CPU while hit DoS criteria. dos_threshold - Allowable SYN/FIN/ICMP packet frame rate. The forwarding rate is dos_threshold*1k packets per second.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.

12.61. rtk_rg_dosFloodType_get

int32 rtk_rg_dosFloodType_get(rtk_rg_dos_type_t *dos_type***, int** **dos_enabled***, rtk_rg_dos_action_t** **dos_action***, int** **dos_threshold***)**

Get denial of service port security setting.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

dos_type
[in] Port security type.
*dos_enabled
[out] Port security function enabled/disabled.
*dos_action
[out] Port security action.
*dos_threshold
[out] System

Comments

dos_type.RTK_RG_SYNFLOOD_DENY - Receiving TCP SYN packet number is over threshold. dos_type.RTK_RG_FINFLOOD_DENY - Receiving TCP FIN packet number is over threshold. dos_type.RTK_RG_ICMPFLOOD_DENY - Receiving ICMP packet number is over threshold. dos_action.RTK_RG_DOS_ACTION_DROP - Drop packet while hit DoS criteria. dos_action.RTK_RG_DOS_ACTION_TRAP - Trap packet to CPU while hit DoS criteria. dos_threshold - Allowable SYN/FIN/ICMP packet frame rate. The forwarding rate is dos_threshold*1k packets per second.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.
RT_ERR_RG_NULL_POINTER	input port mask pointer is NULL.

12.62. rtk_rg_portMirror_set

int32 rtk_rg_portMirror_set(rtk_rg_portMirrorInfo_t *portMirrorInfo***)**

Enable portMirror for monitor Tx/Rx packet.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

portMirrorInfo
[in] assign monitor port

Comments

None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.
RT_ERR_RG_NULL_POINTER	input port mask pointer is NULL.

12.63. rtk_rg_portMirror_get

int32 rtk_rg_portMirror_get(rtk_rg_portMirrorInfo_t **portMirrorInfo*)
Get portMirror for monitor Tx/Rx packet information.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*portMirrorInfo
[out] assigned monitor port

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input port mask pointer is NULL.

12.64. rtk_rg_portMirror_clear

int32 rtk_rg_portMirror_clear(*void*)
clear port mirror setting
Defined in: rtk_rg_liteRomeDriver.h

Parameters
void

Comments
None

Return Codes

RT_ERR_RG_OK	Success

12.65. rtk_rg_portEgrBandwidthCtrlRate_set

int32 rtk_rg_portEgrBandwidthCtrlRate_set(rtk_rg_mac_port_idx_t *port*, **uint32** *rate*)
Set egress rate limitation for physical port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port
[in] assigned rate limit port
rate
[in] assigned rate, unit Kbps

Comments
(1) The rate unit is 1 kbps and the range is from 8k to 1048568k.
(2) The granularity of rate is 8 kbps
(3) rate set 0, means unlimit

Return Codes

RT_ERR_RG_OK	Success

12.66. rtk_rg_portIgrBandwidthCtrlRate_set

int32 rtk_rg_portIgrBandwidthCtrlRate_set(rtk_rg_mac_port_idx_t *port*, **uint32** *rate*)
Set ingress rate limitation for physical port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port
[in] assigned rate limit port

	rate	
	[in]	assigned rate, unit Kbps
Comments	(1) The rate unit is 1 kbps and the range is from 8k to 1048568k. (2) The granularity of rate is 8 kbps (3) rate set 0, means unlimit	
Return Codes	RT_ERR_RG_OK	Success

12.67. rtk_rg_portEgrBandwidthCtrlRate_get

int32 rtk_rg_portEgrBandwidthCtrlRate_get(rtk_rg_mac_port_idx_t *port*, **uint32** **rate*)

Get egress rate limitation of physical port.

Defined in: rtk_rg_liteRomeDriver.h

Parameters	port	
	[in]	assigned rate limit port
	*rate	
	[out]	assigned rate, unit Kbps
Comments	None	
Return Codes	RT_ERR_RG_OK	Success
	RT_ERR_RG_NULL_POINTER	input rate pointer is NULL.

12.68. rtk_rg_portIgrBandwidthCtrlRate_get

int32 rtk_rg_portIgrBandwidthCtrlRate_get(rtk_rg_mac_port_idx_t *port*, **uint32** **rate*)

Get ingress rate limitation of physical port.

Defined in: rtk_rg_liteRomeDriver.h

Parameters	port	
	[in]	assigned rate limit port
	*rate	
	[out]	assigned rate, unit Kbps
Comments	None	
Return Codes	RT_ERR_RG_OK	Success
	RT_ERR_RG_NULL_POINTER	input rate pointer is NULL.

12.69. rtk_rg_phyPortForceAbility_set

int32 rtk_rg_phyPortForceAbility_set(rtk_rg_mac_port_idx_t *port*, **rtk_rg_phyPortAbilityInfo_t** *ability*)

Force Set physical port status & ability.

Defined in: rtk_rg_liteRomeDriver.h

Parameters	port	
	[in]	assigned physical port
	ability	
	[in]	ENABLED:force link
Comments	None	
Return Codes	RT_ERR_RG_OK	Success

12.70. rtk_rg_phyPortForceAbility_get

int32 rtk_rg_phyPortForceAbility_get(rtk_rg_mac_port_idx_t *port*, rtk_rg_phyPortAbilityInfo_t **ability*)
Get physical port status & ability.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port
[in] assigned physical port
*ability
[out] ENABLED:force link

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input ability pointer is NULL.

12.71. rtk_rg_portMibInfo_get

int32 rtk_rg_portMibInfo_get(rtk_rg_mac_port_idx_t *port*, rtk_rg_port_mib_info_t **mibInfo*)
Get physical port mib data.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port
[in] assigned physical port
*mibInfo
[out] Enabled/Disabled Force assigned phyPort ability

Comments mibInfo->ifInOctets - The total number of octets received on the interface¡A including framing characters. mibInfo->ifInUcastPkts - The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. mibInfo->ifInMulticastPkts - The number of packets¡A delivered by this sub-layer to a higher (sub-)layer¡A which were addressed to a multicast address at this sub-layer. For a MAC layer protocol¡A this includes both Group and Functional addresses. mibInfo->ifInBroadcastPkts - The number of packets¡A delivered by this sub-layer to a higher (sub-)layer¡A which were addressed to a broadcast address at this sub-layer. mibInfo->ifInDiscards - The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. mibInfo->ifOutOctets - The total number of octets transmitted out of the interface¡A including framing characters. mibInfo->ifOutDiscards - The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. mibInfo->ifOutUcastPkts - The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. mibInfo->ifOutMulticastPkts - The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. mibInfo->ifOutBrocastPkts - The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. mibInfo->dot1dBasePortDelayExceededDiscards - The number of transmitted frames that were discarded due to the maximum bridge transit delay being exceeded. mibInfo->dot1dTpPortInDiscards - Count of valid frames received which were

discarded (i.e., filtered) by the Forwarding Process. mibInfo->dot1dTpHcPortInDiscards - The total number of Forwarding Database entries¡A which have been or would have been learnt¡A but have been discarded due to lack of space to store them in the Forwarding Database. mibInfo->dot3InPauseFrames - A count of MAC Control frames received on this interface with an opcode indicating the PAUSE operation. mibInfo->dot3OutPauseFrames - A count of MAC control frames transmitted on this interface with an opcode indicating the PAUSE operation. mibInfo->dot3StatsAligmentErrors - The total number of bits of a received frame is not divisible by eight. mibInfo->dot3StatsFCSErrors - The number of received frame that frame check sequence error. mibInfo->dot3StatsSingleCollisionFrames - A count of frames that are involved in a single collision¡A and are subsequently transmitted successfully. mibInfo->dot3StatsMultipleCollisionFrames - A count of frames that are involved in more than one collision and are subsequently transmitted successfully. mibInfo->dot3StatsDeferredTransmissions - A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. mibInfo->dot3StatsLateCollisions - The number of times that a collision is detected on a particular interface later than one slotTime into the transmission of a packet. mibInfo->dot3StatsExcessiveCollisions - A count of frames for which transmission on a particular interface fails due to excessive collisions. mibInfo->dot3StatsFrameTooLongs - RX_etherStatsOversizePkt - RX_etherStatsPkts1519toMaxOctets. mibInfo->dot3StatsSymbolErrors - Number of data symbol errors. mibInfo->dot3ControlInUnknownOpcodes - A count of MAC Control frames received on this interface that contain an opcode that is not supported by this device. mibInfo->etherStatsDropEvents - Number of received packets dropped due to lack of resource. mibInfo->etherStatsOctets - ifInOctets(bad+good) + ifOutOctets(bad+good). mibInfo->etherStatsBcastPkts - The total number of packets that were directed to the broadcast address. Note that this does not include multicast packets. mibInfo->etherStatsMcastPkts - The total number of packets that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address. mibInfo->etherStatsUndersizePkts - The total number of packets that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. mibInfo->etherStatsOversizePkts - Number of valid packets whose size are more than 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsFragments - Number of invalid (FCS error / alignment error) packets whose size are less than 64 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsJabbers - Number of invalid (FCS error / alignment error) packets whose size are more than 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsCollisions - The best estimate of the total number of collisions on this Ethernet segment. mibInfo->etherStatsCRCAlignErrors - The total number of packets that had a length (excluding framing bits, but including FCS octets) of between 64 and 1518 octets, inclusive, but had either a bad Frame Check Sequence (FCS) with an integral number of mibInfo->etherStatsPkts64Octets - Number of all packets whose size are exactly 64 bytes¡Aincluding MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsPkts65to127Octets - Number of all packets whose size are between 65 ~ 127 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsPkts128to255Octets - Number of all packets whose size are between 128 ~ 255 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsPkts256to511Octets - Number of all packets whose size are between 256 ~ 511 bytes¡A including FCS¡A excluding MAC header¡A preamble and SFD. mibInfo->etherStatsPkts512to1023Octets - Number of all packets whose size are between 512 ~ 1023 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsPkts1024to1518Octets - Number of all packets whose size are between 1024 ~ 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxOctets - ifOutOctets. mibInfo->etherStatsTxUndersizePkts - The total number of packets transmitted that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. mibInfo->etherStatsTxOversizePkts - Number of transmitted valid packets whose size are more than 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts64Octets - Number

of all transmitted packets whose size are exactly 64 bytes¡Aincluding MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts65to127Octets - Number of all transmitted packets whose size are between 65 ~ 127 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts128to255Octets - Number of all transmitted packets whose size are between 128 ~ 255 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts256to511Octets - Number of all transmitted packets whose size are between 256 ~ 511 bytes¡A including FCS¡A excluding MAC header¡A preamble and SFD. mibInfo->etherStatsTxPkts512to1023Octets - Number of all transmitted packets whose size are between 512 ~ 1023 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts1024to1518Octets - - Number of all transmitted packets whose size are between 1024 ~ 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxPkts1519toMaxOctets - The total number of packets (including bad packets) transmitted that were between 1519 and Max octets in length inclusive (excluding framing bits but including FCS octets). mibInfo->etherStatsTxBcastPkts - The total number of good packets transmitted that were directed to the broadcast address. Note that this does not include multicast packets. mibInfo->etherStatsTxMcastPkts - The total number of good packets transmitted that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address. mibInfo->etherStatsTxFragments - Number of transmitted invalid (FCS error / alignment error) packets whose size are less than 64 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxJabbers - Number of transmitted invalid (FCS error / alignment error) packets whose size are more than 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsTxCRCAlignErrors - The total number of packets transmitted that had a length (excluding framing bits, but including FCS octets) of between 64 and 1518 octets, inclusive, but had either a bad Frame Check Sequence (FCS) with an integ mibInfo->etherStatsRxUndersizePkts - The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. mibInfo->etherStatsRxUndersizeDropPkts - The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. mibInfo->etherStatsRxOversizePkts - Number of received valid packets whose size are more than 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts64Octets - Number of all received packets whose size are exactly 64 bytes¡Aincluding MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts65to127Octets - Number of all received packets whose size are between 65 ~ 127 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts128to255Octets - Number of all received packets whose size are between 128 ~ 255 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts256to511Octets - Number of all received packets whose size are between 256 ~ 511 bytes¡A including FCS¡A excluding MAC header¡A preamble and SFD. mibInfo->etherStatsRxPkts512to1023Octets - Number of all received packets whose size are between 512 ~ 1023 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts1024to1518Octets - Number of all received packets whose size are between 1024 ~ 1518 bytes¡A including MAC header and FCS¡A excluding preamble and SFD. mibInfo->etherStatsRxPkts1519toMaxOctets - The total number of packets (including bad packets) received that were between 1519 and Max octets in length inclusive (excluding framing bits but including FCS octets). mibInfo->inOampduPkts - Number of received OAMPDUs. mibInfo->outOampduPkts - A count of OAMPDUs transmitted on this interface.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input ability pointer is NULL.
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.

12.72. rtk_rg_portMibInfo_clear

int32 rtk_rg_portMibInfo_clear(rtk_rg_mac_port_idx_t *port*)
Clear physical port mib data.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port
[in] assigned physical port

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	the input parameter contains illegal information.

12.73. rtk_rg_stormControl_add

int32 rtk_rg_stormControl_add(rtk_rg_stormControlInfo_t **stormInfo*, int **stormInfo_idx*)
Add stormControl.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*stormInfo
[in] enable stormCtrol
*stormInfo_idx
[in] assigned port

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	parameter is null.
RT_ERR_RG_INVALID_PARAM	parameters value is out of range.
RT_ERR_RG_STORMCONTROL_ENTRY_HAS_BEEN_SET	same stormInfo has been set before .
RT_ERR_RG_STORMCONTROL_TYPE_FULL	add stormType full(ASIC support at most 4 different types)

12.74. rtk_rg_stormControl_del

int32 rtk_rg_stormControl_del(int *stormInfo_idx*)
Delete stormControl.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
stormInfo_idx
[out] The index of stormInfo which should be delete.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameters value is out of range.

12.75. rtk_rg_stormControl_find

int32 rtk_rg_stormControl_find(rtk_rg_stormControlInfo_t **stormInfo*, **int** **stormInfo_idx*)

Find stormInfo index.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

*stormInfo

[in] assigned compare port

*stormInfo_idx

[in] assigned compare stormType

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	parameter is null.
RT_ERR_RG_INVALID_PARAM	parameters value is out of range.
RT_ERR_RG_STORMCONTROL_ENTRY_NOT_FOUND	assigned stormInfo not found

12.76. rtk_rg_shareMeter_set

int32 rtk_rg_shareMeter_set(uint32 *index*, **uint32** *rate*, **rtk_rg_enable_t** *ifgInclude*)

Set shareMeter rate.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

index

[in] shared meter index

rate

[in] rate of share meter

ifgInclude

[in] include IFG or not, ENABLE:include DISABLE:exclude

Comments The API can set shared meter rate and ifg include for each meter. The rate unit is 1 kbps and the range is from 8k to 1048568k. The granularity of rate is 8 kbps. The ifg_include parameter is used for rate calculation with/without inter-frame-gap and preamble.

Return Codes

RT_ERR_RG_OK	Success

12.77. rtk_rg_shareMeter_get

int32 rtk_rg_shareMeter_get(uint32 *index*, **uint32** **pRate*, **rtk_rg_enable_t** **pIfgInclude*)

Get shareMeter rate.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

index

[in] shared meter index

*pRate

[out] pointer of rate of share meter

*pIfgInclude

[out] include IFG or not, ENABLE:include DISABLE:exclude

Comments The API can get shared meter rate and ifg include for each meter. The rate unit is 1 kbps and

the granularity of rate is 8 kbps. The ifg_include parameter is used for rate calculation with/without inter-frame-gap and preamble

Return Codes

RT_ERR_RG_OK	Success

12.78. rtk_rg_qosStrictPriorityOrWeightFairQueue_set

int32 rtk_rg_qosStrictPriorityOrWeightFairQueue_set(rtk_rg_mac_port_idx_t *port_idx*, **rtk_rg_qos_queue_weights_t** *q_weight*)

Set the scheduling types and weights of queues on specific port in egress scheduling.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

port_idx

[in] The port index

q_weight

[in] The array of weights for WRR/WFQ queue (valid:1~128, 0 for STRICT_PRIORITY queue)

Comments

The types of queue are: WFQ_WRR_PRIORITY or STRICT_PRIORITY. If the weight is 0 then the type is STRICT_PRIORITY, else the type is WFQ_WRR_PRIORITY.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_FAILED	parameters value is out of range.

12.79. rtk_rg_qosStrictPriorityOrWeightFairQueue_get

int32 rtk_rg_qosStrictPriorityOrWeightFairQueue_get(rtk_rg_mac_port_idx_t *port_idx*, **rtk_rg_qos_queue_weights_t** **pQ_weight*)

Get the scheduling types and weights of queues on specific port in egress scheduling.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

port_idx

[in] The port index.

*pQ_weight

[in] Pointer to the array of weights for WRR/WFQ queue (valid:1~128, 0 for STRICT_PRIORITY queue)

Comments

The types of queue are: WFQ_WRR_PRIORITY or STRICT_PRIORITY. If the weight is 0 then the type is STRICT_PRIORITY, else the type is WFQ_WRR_PRIORITY.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.
RT_ERR_RG_FAILED	assigned stormInfo not found

12.80. rtk_rg_qosInternalPriMapToQueueId_set

int32 rtk_rg_qosInternalPriMapToQueueId_set(int *int_pri*, **int** *queue_id*)
Set the entry of internal priority to QID mapping table.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal priority
queue_id
[in] Mapping queue Id.

Comments
Below is an example of internal priority to QID mapping table.

```
-
- Priority     0     1     2     3     4     5     6     7
-                   ====================
- queue         0     0     1     1     2     2     3     3
-for table index 0
-       pPri2qid[0] = 0     internal priority 0 map to queue 0
-       pPri2qid[1] = 0     internal priority 1 map to queue 0
-       pPri2qid[2] = 1     internal priority 2 map to queue 1
-       pPri2qid[3] = 1     internal priority 3 map to queue 1
-       pPri2qid[4] = 2     internal priority 4 map to queue 2
-       pPri2qid[5] = 2     internal priority 5 map to queue 2
-       pPri2qid[6] = 3     internal priority 6 map to queue 3
-       pPri2qid[7] = 3     internal priority 7 map to queue 3
```

Return Codes

RT_ERR_RG_INVALID_PARAM	Success
RT_ERR_OK	ok

12.81. rtk_rg_qosInternalPriMapToQueueId_get

int32 rtk_rg_qosInternalPriMapToQueueId_get(int *int_pri*, **int** **pQueue_id*)
Get the entry of internal priority to QID mapping table.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal priority
*pQueue_id
[out] Pointer to mapping queue Id.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.82. rtk_rg_qosInternalPriDecisionByWeight_set

int32 rtk_rg_qosInternalPriDecisionByWeight_set(rtk_rg_qos_priSelWeight_t *weightOfPriSel*)
Set weight of each priority assignment.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
weightOfPriSel
[in] weight of each priority assignment

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.83. rtk_rg_qosInternalPriDecisionByWeight_get

int32 rtk_rg_qosInternalPriDecisionByWeight_get(rtk_rg_qos_priSelWeight_t **pWeightOfPriSel*)
Get weight of each priority assignment.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*pWeightOfPriSel
[out] weight of each priority assignment

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	parameter is null.

12.84. rtk_rg_qosDscpRemapToInternalPri_set

int32 rtk_rg_qosDscpRemapToInternalPri_set(uint32 *dscp*, uint32 *int_pri*)
Set remapped internal priority of DSCP.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dscp
[in] DSCP
int_pri
[in] internal priority

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.85. rtk_rg_qosDscpRemapToInternalPri_get

int32 rtk_rg_qosDscpRemapToInternalPri_get(uint32 *dscp*, **uint32** **pInt_pri*)
Get remapped internal priority of DSCP.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dscp
[in] DSCP
*pInt_pri
[out] internal priority

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.86. rtk_rg_qosPortBasedPriority_set

int32 rtk_rg_qosPortBasedPriority_set(rtk_rg_mac_port_idx_t *port_idx*, **uint32** *int_pri*)
Set internal priority of one port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port_idx
[in] Port index, including extension port.
int_pri
[in] Priorities assigment for specific port.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.87. rtk_rg_qosPortBasedPriority_get

int32 rtk_rg_qosPortBasedPriority_get(rtk_rg_mac_port_idx_t *port_idx*, **uint32** **pInt_pri*)
Get internal priority of one port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port_idx
[in] Port index, including extension port.
*pInt_pri
[out] Priorities assigment for specific port.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.88. rtk_rg_qosDot1pPriRemapToInternalPri_set

int32 rtk_rg_qosDot1pPriRemapToInternalPri_set(uint32 *dot1p*, **uint32** *int_pri*)
Set remapped internal priority of 802.1p priority.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dot1p
[in] 802.1p priority
int_pri
[in] internal priority

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.89. rtk_rg_qosDot1pPriRemapToInternalPri_get

int32 rtk_rg_qosDot1pPriRemapToInternalPri_get(uint32 *dot1p*, **uint32** **pInt_pri*)
Get remapped internal priority of 802.1p priority.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dot1p
[in] 802.1p priority
*pInt_pri
[out] internal priority

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.90. rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set

int32 rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set(rtk_rg_mac_port_idx_t *rmk_port*, **rtk_rg_enable_t** *rmk_enable*, **rtk_rg_qos_dscpRmkSrc_t** *rmk_src_select*)
Set remark DSCP enable/disable port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
rmk_port
[in] Enable/Dsiable DSCP remarking port
rmk_enable
[in] Enable/Disable DSCP remark

rmk_src_select
[in] DSCP remarking source

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.91. rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get

int32 rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get(rtk_rg_mac_port_idx_t *rmk_port***, rtk_rg_enable_t** **pRmk_enable***, rtk_rg_qos_dscpRmkSrc_t** **pRmk_src_select***)**
Get remark DSCP enable/disable port and remarking source.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
rmk_port
[in] DSCP remarking port
*pRmk_enable
[in] Pointer to DSCP remarking port enabled/disabled
*pRmk_src_select
[in] Pointer to DSCP remarking source

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.
RT_ERR_RG_FAILED	assigned stormInfo not found

12.92. rtk_rg_qosDscpRemarkByInternalPri_set

int32 rtk_rg_qosDscpRemarkByInternalPri_set(int *int_pri***, int** *rmk_dscp***)**
Set Internal priority/User priority to DSCP remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal/User priority
rmk_dscp
[in] DSCP remarking value

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.93. rtk_rg_qosDscpRemarkByInternalPri_get

int32 rtk_rg_qosDscpRemarkByInternalPri_get(int *int_pri*, **int** **pRmk_dscp*)
Get Internal priority/User priority to DSCP remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal/User priority
*pRmk_dscp
[out] Pointer to remarking DSCP

Comments
rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.94. rtk_rg_qosDscpRemarkByDscp_set

int32 rtk_rg_qosDscpRemarkByDscp_set(int *dscp*, **int** *rmk_dscp*)
Set DSCP to DSCP remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dscp
[in] DSCP source
rmk_dscp
[in] DSCP remarking value

Comments
rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.95. rtk_rg_qosDscpRemarkByDscp_get

int32 rtk_rg_qosDscpRemarkByDscp_get(int *dscp*, **int** **pRmk_dscp*)
Get DSCP to DSCP remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
dscp
[in] DSCP source
*pRmk_dscp
[out] Pointer to DSCP remarking value

Comments
rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP -

Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.96. rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set

int32
rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set(rtk_rg_mac_port_idx_t *rmk_port***, rtk_rg_enable_t** *rmk_enable***)**
Set remark 802.1p priority port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters

rmk_port
[in] Enable/Dsiable 802.1p remarking port.
rmk_enable
[in] Enable/Disable 1p remark.

Comments

rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.97. rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get

int32
rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get(rtk_rg_mac_port_idx_t *rmk_port***, rtk_rg_enable_t** **pRmk_enable***)**
Get remark 802.1p priority port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters

rmk_port
[out] Pointer to 802.1p remarking port.
*pRmk_enable
[in] Enable/Disable 1p remark.

Comments

rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.98. rtk_rg_qosDot1pPriRemarkByInternalPri_set

int32 rtk_rg_qosDot1pPriRemarkByInternalPri_set(int *int_pri*, **int** *rmk_dot1p*)
Set Internal priority to 802.1p remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal priority
rmk_dot1p
[in] 802.1p priority remarking value

Comments
rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.

12.99. rtk_rg_qosDot1pPriRemarkByInternalPri_get

int32 rtk_rg_qosDot1pPriRemarkByInternalPri_get(int *int_pri*, **int** **pRmk_dot1p*)
Get Internal priority to 802.1p remarking mapping.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
int_pri
[in] Internal priority
*pRmk_dot1p
[out] Pointer to 802.1p priority remarking value

Comments
rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	parameters value is out of range.

12.100. rtk_rg_portBasedCVlanId_set

int32 rtk_rg_portBasedCVlanId_set(rtk_rg_mac_port_idx_t *port_idx*, **int** *pvid*)
Set port vlan ID.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
port_idx
[in] Port index
pvid
[in] Wished port vlan id.

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_ENTRY_NOT_EXIST	VlanID not exist

12.101. rtk_rg_portBasedCVlanId_get

int32 rtk_rg_portBasedCVlanId_get(rtk_rg_mac_port_idx_t *port_idx*, **int** **pPvid*)

Get port vlan ID.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

port_idx

[in] Port index

*pPvid

[in] Pointer to port vlan id.

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	VlanID not exist

12.102. rtk_rg_portStatus_get

int32 rtk_rg_portStatus_get(rtk_rg_mac_port_idx_t *port*, **rtk_rg_portStatusInfo_t** **portInfo*)

Get port linkup status.

Defined in: rtk_rg_liteRomeDriver.h

Parameters

port

[in] Port index

*portInfo

[out] port linkup information.

Comments rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMK_SRC_DSCP - Using DSCP as DSCP remarking source.

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_INVALID_PARAM	parameter is null.
RT_ERR_RG_NULL_POINTER	VlanID not exist

12.103. rtk_rg_naptExtPortGet

int32 rtk_rg_naptExtPortGet(int *isTcp*, **uint16** **pPort*)
Find a usable external port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
isTcp
[in] TCP or UDP
*pPort
[out] Pointer to a specific wish port

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	parameter is null.

12.104. rtk_rg_naptExtPortFree

int32 rtk_rg_naptExtPortFree(int *isTcp*, **uint16** *port*)
Free external port.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
isTcp
[in] TCP or UDP
port
[out] L4 port number

Comments
Apollo only support group 0

Return Codes

RT_ERR_RG_OK	Success

12.105. rtk_rg_gpon_infoSettings_set

int32 rtk_rg_gpon_infoSettings_set(rtk_rg_gpon_infoSettings_t *gpon_info*)
Set GPON MAC informations.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
gpon_info
[in] MAC serialNumber, password, and LOID.

Comments
None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_GPON_SET_INFO_FAIL	parameter is null.

ED

12.106. rtk_rg_gpon_infoSettings_get

int32 rtk_rg_gpon_infoSettings_get(rtk_rg_gpon_infoSettings_t **gpon_info*)
Get GPON MAC informations.
Defined in: rtk_rg_liteRomeDriver.h

Parameters *gpon_info
[out] MAC serialNumber, password, and LOID.
Comments None
Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input gpon_info is NULL.
RT_ERR_RG_GPON_GET_INFO_FAILED	VlanID not exist

12.107. rtk_rg_gpon_status_get

int32 rtk_rg_gpon_status_get(rtk_rg_gpon_status_t **gpon_status*)
Get GPON ONU status.
Defined in: rtk_rg_liteRomeDriver.h
Parameters *gpon_status
[out] FSM status.
Comments None
Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input gpon_status is NULL.
RT_ERR_RG_GPON_GET_STATUS_FAILED	VlanID not exist

12.108. rtk_rg_epon_infoSettings_set

int32 rtk_rg_epon_infoSettings_set(int *llid_idx*, **rtk_rg_epon_infoSettings_t** *epon_info*)
Set EPON MAC informations.
Defined in: rtk_rg_liteRomeDriver.h
Parameters llid_idx
[in] LLID index want to set.
epon_info
[in] LLID MAC mappings, and LOID.
Comments None
Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_GPON_SET_INFO_FAILED	input gpon_status is NULL.

12.109. rtk_rg_epon_infoSettings_get

int32 rtk_rg_epon_infoSettings_get(int *llid_idx*, **rtk_rg_epon_infoSettings_t** **epon_info*)
Get EPON MAC informations.
Defined in: rtk_rg_liteRomeDriver.h
Parameters llid_idx
[in] LLID index want to get.
*epon_info
[out] LLID MAC mappings, and LOID.
Comments None
Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input epon_info is NULL.
RT_ERR_RG_EPON_GET_INFO_FAI	VlanID not exist

12.110. rtk_rg_pon_transceiverInfo_get

int32 rtk_rg_pon_transceiverInfo_get(rtk_rg_transceiverType_t *type*, rtk_rg_transceiverBuf_t **transc_info*)
Get PON transceiver informations.
Defined in: rtk_rg_liteRomeDriver.h

Parameters
type
[in] parameter type want to get.
*transc_info
[out] parameter data.

Comments None

Return Codes

RT_ERR_RG_OK	Success
RT_ERR_RG_NULL_POINTER	input transc_info is NULL.
RT_ERR_RG_PON_GET_INFO_FAILED	VlanID not exist

12.111. rtk_rg_classifyEntry_add

int32 rtk_rg_classifyEntry_add(rtk_rg_classifyEntry_t **classifyFilter*)
Add an classification entry to ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters
*classifyFilter
The classification configuration that this function will add comparison rule

Comments None.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Pointer classifyFilter point to NULL.
RT_ERR_INPUT	Invalid input parameters.

12.112. rtk_rg_classifyEntry_find

int32 rtk_rg_classifyEntry_find(int *index*, rtk_rg_classifyEntry_t **classifyFilter*)
Get an classification entry from ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters
index
The Index of classification entry
*classifyFilter
The classification configuration

Comments None.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Pointer pClassifyCfg point to NULL.
RT_ERR_INPUT	Invalid input parameters.

12.113. rtk_rg_classifyEntry_del

int32 rtk_rg_classifyEntry_del(int *index***)**
Delete an classification configuration from ASIC
Defined in: rtk_rg_liteRomeDriver.h

Parameters
index
index of classification entry.

Comments
None.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_ENTRY_INDEX	Invalid classification index .