

REALTEK

Realtek RG Rome Driver

Programming Guide

Version 1.1.0

(English Version)

Sep 25, 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

COPYRIGHT

©2012 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

DISCLAIMER

Realtek provides this document “as is”, without warranty of any kind, neither expressed nor implied, including, but not limited to, the particular purpose. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

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 example. 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 architecture figures & description English Translation | Chuck |
| 1.0.8 | 2013/05/27 | Update Gateway basic configuration example Add new RG APIs & example Add Gateway Advance Configuration examples Update Build Code commands Update diagshell introduction | Chuck |
| 1.0.9 | 2013/08/28 | Update RG Interface 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 performance test report. | Tony |

Table of Contents

| | | |
|----------|--|----|
| 1. | Architecture | 12 |
| 1.1. | RomeDriver Architecture | 12 |
| 1.2. | Lite RomeDriver Architecture | 14 |
| 2. | RTK RG APIs | 15 |
| 3. | Checkout Code | 20 |
| 4. | Build Code..... | 20 |
| 5. | NAPT External Port Collision Prevention | 20 |
| 5.1.1. | Enable/Disable NAPT External Port Collision Prevention | 21 |
| 5.1.2. | API: rtk_rg_naptExtPortGet..... | 23 |
| 5.1.3. | API: rtk_rg_naptExtPortFree | 23 |
| 5.1.4. | Sample Codes for Hooking NAPT External Port Collision Prevention Mechanism in Linux 2.6.30 | 24 |
| 6. | (Lite)RomeDriver Initiation..... | 26 |
| 6.1. | Gateway Basic Configuration | 27 |
| 6.2. | Gateway Advanced Configuration | 34 |
| 6.2.1. | Port Binding..... | 34 |
| 6.2.2. | VLAN Binding | 34 |
| 6.2.3. | ACL Filter / QoS Remarking | 35 |
| 6.2.4. | Rate Limit..... | 37 |
| 6.2.4.1. | ACL-Based Share Meter | 37 |
| 6.2.4.2. | Egress Port -Based Rate Limit | 38 |
| 6.2.4.3. | Ingress Port-Based Rate Limit | 38 |
| 6.2.5. | QoS Strict Priority / WFQ..... | 39 |
| 6.2.5.1. | DSCP to Internal Priority | 40 |
| 6.2.5.2. | Port to Internal Priority..... | 40 |
| 6.2.6. | L2 per-Port learning count limit | 41 |
| 6.2.7. | MacFilter | 42 |
| 6.2.8. | Application-Level-Gateway(ALG) | 42 |
| 6.2.9. | Virtual Server | 42 |
| 6.2.10. | DmzHost | 43 |
| 6.2.11. | UpnpConnection | 43 |
| 6.2.12. | MulticastFlow | 44 |
| 6.2.13. | StormControl | 44 |
| 6.2.14. | PortMirror | 45 |
| 6.2.15. | Physical Port Ability | 45 |

| | | |
|---------|---|----|
| 6.2.16. | Denial-of-service (D.O.S) | 46 |
| 7. | Wireless LAN Hardware Forward..... | 47 |
| 8. | Test by Web UI configuration | 50 |
| 8.1. | Configure LAN by Web UI | 50 |
| 8.2. | Configure WAN by Web UI..... | 51 |
| 9. | Test by Diagnostics Shell Command | 51 |
| 9.1. | Diag Shell Introduction | 51 |
| 9.1.1. | Startup DiagShell | 52 |
| 9.1.2. | Command-Line-Completion for Diag Cmd | 52 |
| 9.1.3. | Mapping for RG API & Diag Cmd | 54 |
| 9.2. | NIC Driver network device name & HW Port Index mapping | 56 |
| 9.3. | Diag Shell for Realtek RG APIs..... | 56 |
| 9.3.1. | Initial RG API..... | 56 |
| 9.3.2. | Initial RG API and register callback function..... | 56 |
| 9.3.3. | Add LAN Interface..... | 58 |
| 9.3.4. | Add WAN Interface..... | 58 |
| 9.3.5. | Set Static WAN Configuration..... | 59 |
| 9.4. | Diag Shell for Realtek RG Runtime API | 59 |
| 9.4.1. | Get RG API Version Info..... | 59 |
| 9.4.2. | Add MAC Entry | 59 |
| 9.4.3. | Add ARP Entry | 60 |
| 9.4.4. | Add NAPT Entry | 60 |
| 10. | Realtek RG Debug Tools | 60 |
| 11. | Test Report | 65 |
| 11.1. | Chariot-LAN to LAN Bridge Mode-TCP Throughput..... | 65 |
| 11.2. | Chariot-LAN to WAN NAPT Mode-TCP Throughput | 66 |
| 11.3. | IXIA-LAN to LAN Bridge Mode-TCP(Packet Size:1518Bytes)..... | 67 |
| 11.4. | IXIA-LAN to LAN Bridge Mode-TCP(Packet Size:64Bytes)..... | 68 |
| 11.5. | IXIA-LAN to UTP WAN NAPT Mode-TCP(Packet Size:1518Bytes)..... | 69 |
| 11.6. | IXIA-LAN to UTP WAN NAPT Mode-TCP(Packet Size:64Bytes)..... | 69 |
| 11.7. | IXIA-LAN to GPON WAN NAPT Mode-UDP(Packet Size:1518Bytes) .. | 70 |
| 11.8. | IXIA-GPON WAN to LAN NAPT Mode-UDP(Packet Size:1518Bytes) .. | 71 |
| 11.9. | IXIA-LAN to GPON WAN NAPT Mode-UDP(Packet Size:64Bytes) .. | 71 |
| 11.10. | IXIA-GPON WAN to LAN NAPT Mode-UDP(Packet Size:64Bytes) .. | 72 |
| 12. | RTK RG APIs Prototype..... | 73 |
| 12.1. | rtk_rg_driverVersion_get..... | 73 |
| 12.2. | rtk_rg_initParam_get..... | 73 |
| 12.3. | rtk_rg_initParam_set | 74 |

| | | |
|--------|---|----|
| 12.4. | rtk_rg_lanInterface_add | 74 |
| 12.5. | rtk_rg_wanInterface_add..... | 74 |
| 12.6. | rtk_rg_staticInfo_set..... | 75 |
| 12.7. | rtk_rg_dhcpRequest_set | 76 |
| 12.8. | rtk_rg_dhcpClientInfo_set | 76 |
| 12.9. | rtk_rg_pppoeClientInfoBeforeDial_set..... | 77 |
| 12.10. | rtk_rg_pppoeClientInfoAfterDial_set | 77 |
| 12.11. | rtk_rg_interface_del..... | 78 |
| 12.12. | rtk_rg_intfInfo_find | 79 |
| 12.13. | rtk_rg_cvlan_add..... | 79 |
| 12.14. | rtk_rg_cvlan_del..... | 80 |
| 12.15. | rtk_rg_vlanBinding_add..... | 80 |
| 12.16. | rtk_rg_vlanBinding_del..... | 81 |
| 12.17. | rtk_rg_vlanBinding_find..... | 81 |
| 12.18. | rtk_rg_algServerInLanAppslpAddr_add | 81 |
| 12.19. | rtk_rg_algServerInLanAppslpAddr_del | 82 |
| 12.20. | rtk_rg_algApps_set | 82 |
| 12.21. | rtk_rg_algApps_get | 82 |
| 12.22. | rtk_rg_dmzHost_set | 82 |
| 12.23. | rtk_rg_dmzHost_get | 83 |
| 12.24. | rtk_rg_virtualServer_add | 83 |
| 12.25. | rtk_rg_virtualServer_del | 84 |
| 12.26. | rtk_rg_virtualServer_find | 84 |
| 12.27. | rtk_rg_aclFilterAndQos_add..... | 84 |
| 12.28. | rtk_rg_aclFilterAndQos_del..... | 87 |
| 12.29. | rtk_rg_aclFilterAndQos_find | 87 |
| 12.30. | rtk_rg_macFilter_add | 88 |
| 12.31. | rtk_rg_macFilter_del..... | 88 |
| 12.32. | rtk_rg_macFilter_find | 88 |
| 12.33. | rtk_rg_urlFilterString_add..... | 89 |
| 12.34. | rtk_rg_urlFilterString_del | 89 |
| 12.35. | rtk_rg_urlFilterString_find | 89 |
| 12.36. | rtk_rg_upnpConnection_add | 90 |
| 12.37. | rtk_rg_upnpConnection_del | 90 |
| 12.38. | rtk_rg_upnpConnection_find | 91 |
| 12.39. | rtk_rg_naptConnection_add | 91 |
| 12.40. | rtk_rg_naptConnection_del | 92 |
| 12.41. | rtk_rg_naptConnection_find | 92 |

| | | |
|--------|---|-----|
| 12.42. | rtk_rg_multicastFlow_add | 93 |
| 12.43. | rtk_rg_multicastFlow_del..... | 93 |
| 12.44. | rtk_rg_multicastFlow_find..... | 93 |
| 12.45. | rtk_rg_multicastDataIngressAlllowedPortMask_set | 94 |
| 12.46. | rtk_rg_multicastDataIngressAlllowedPortMask_get | 94 |
| 12.47. | rtk_rg_macEntry_add | 94 |
| 12.48. | rtk_rg_macEntry_del | 95 |
| 12.49. | rtk_rg_macEntry_find | 95 |
| 12.50. | rtk_rg_arpEntry_add..... | 96 |
| 12.51. | rtk_rg_arpEntry_del..... | 96 |
| 12.52. | rtk_rg_arpEntry_find..... | 96 |
| 12.53. | rtk_rg_neighborEntry_add..... | 97 |
| 12.54. | rtk_rg_neighborEntry_del..... | 97 |
| 12.55. | rtk_rg_neighborEntry_find..... | 98 |
| 12.56. | rtk_rg_softwareSourceAddrLearningLimit_set | 98 |
| 12.57. | rtk_rg_softwareSourceAddrLearningLimit_get | 98 |
| 12.58. | rtk_rg_dosPortMaskEnable_set | 99 |
| 12.59. | rtk_rg_dosPortMaskEnable_get | 99 |
| 12.60. | rtk_rg_dosType_set | 99 |
| 12.61. | rtk_rg_dosType_get | 100 |
| 12.62. | rtk_rg_dosFloodType_set..... | 101 |
| 12.63. | rtk_rg_dosFloodType_get | 101 |
| 12.64. | rtk_rg_portMirror_set..... | 102 |
| 12.65. | rtk_rg_portMirror_get..... | 102 |
| 12.66. | rtk_rg_portMirror_clear..... | 102 |
| 12.67. | rtk_rg_portEgrBandwidthCtrlRate_set..... | 102 |
| 12.68. | rtk_rg_portIgrBandwidthCtrlRate_set | 103 |
| 12.69. | rtk_rg_portEgrBandwidthCtrlRate_get..... | 103 |
| 12.70. | rtk_rg_portIgrBandwidthCtrlRate_get | 103 |
| 12.71. | rtk_rg_phyPortForceAbility_set | 104 |
| 12.72. | rtk_rg_phyPortForceAbility_get | 104 |
| 12.73. | rtk_rg_portMibInfo_get | 104 |
| 12.74. | rtk_rg_portMibInfo_clear | 107 |
| 12.75. | rtk_rg_stormControl_add..... | 107 |
| 12.76. | rtk_rg_stormControl_del..... | 108 |
| 12.77. | rtk_rg_stormControl_find..... | 108 |
| 12.78. | rtk_rg_shareMeter_set | 108 |
| 12.79. | rtk_rg_shareMeter_get | 109 |

| | | |
|---------|---|-----|
| 12.80. | rtk_rg_qosStrictPriorityOrWeightFairQueue_set | 109 |
| 12.81. | rtk_rg_qosStrictPriorityOrWeightFairQueue_get | 109 |
| 12.82. | rtk_rg_qosInternalPriMapToQueueId_set | 110 |
| 12.83. | rtk_rg_qosInternalPriMapToQueueId_get | 110 |
| 12.84. | rtk_rg_qosInternalPriDecisionByWeight_set | 111 |
| 12.85. | rtk_rg_qosInternalPriDecisionByWeight_get | 111 |
| 12.86. | rtk_rg_qosDscpRemapToInternalPri_set | 111 |
| 12.87. | rtk_rg_qosDscpRemapToInternalPri_get | 112 |
| 12.88. | rtk_rg_qosPortBasedPriority_set | 112 |
| 12.89. | rtk_rg_qosPortBasedPriority_get | 112 |
| 12.90. | rtk_rg_qosDot1pPriRemapToInternalPri_set | 113 |
| 12.91. | rtk_rg_qosDot1pPriRemapToInternalPri_get | 113 |
| 12.92. | rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_set | 113 |
| 12.93. | rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get | 114 |
| 12.94. | rtk_rg_qosDscpRemarkByInternalPri_set | 114 |
| 12.95. | rtk_rg_qosDscpRemarkByInternalPri_get | 115 |
| 12.96. | rtk_rg_qosDscpRemarkByDscp_set | 115 |
| 12.97. | rtk_rg_qosDscpRemarkByDscp_get | 115 |
| 12.98. | rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_set | 116 |
| 12.99. | rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get | 116 |
| 12.100. | rtk_rg_qosDot1pPriRemarkByInternalPri_set | 117 |
| 12.101. | rtk_rg_qosDot1pPriRemarkByInternalPri_get | 117 |
| 12.102. | rtk_rg_portBasedCVlanId_get | 118 |
| 12.103. | rtk_rg_portStatus_get | 118 |
| 12.104. | rtk_rg_naptExtPortGet | 118 |
| 12.105. | rtk_rg_naptExtPortFree | 119 |

List of Figures

| | | |
|-----------|--|----|
| Figure1. | RomeDriver Architecture | 13 |
| Figure2. | Lite RomeDriver Architecture | 15 |
| Figure3. | Enable NAPT External Port Collision Prevention, step1..... | 21 |
| Figure4. | Enable NAPT External Port Collision Prevention, step2..... | 22 |
| Figure5. | Enable NAPT External Port Collision Prevention, step3..... | 22 |
| Figure6. | Enable NAPT External Port Collision Prevention, step4..... | 23 |
| Figure7. | NAPT External Port choosing flow..... | 23 |
| Figure8. | RomeDriver Initiation Flow Chart | 26 |
| Figure9. | NIC TX Descriptor Format..... | 47 |
| Figure10. | NIC RX Descriptor Format | 48 |
| Figure11. | Wireless LAN Hardware Forward..... | 49 |
| Figure12. | Web UI:LAN Interface Setting | 50 |
| Figure13. | Web UI:Ethernet WAN Interface Setting..... | 51 |
| Figure14. | UART: ADSL Main Menu | 52 |
| Figure15. | DiagShell tips for next keyword | 53 |
| Figure16. | DiagShell command-line-completion | 53 |
| Figure17. | Chariot LAN to LAN TCP connection | 66 |
| Figure18. | Chariot LAN to WAN TCP connection | 67 |
| Figure19. | IXIA LAN to LAN TCP connection(Large Packet)..... | 68 |
| Figure20. | IXIA LAN to LAN TCP connection(Small Packet) | 68 |
| Figure21. | IXIA LAN to WAN TCP connection(Large Packet)..... | 69 |
| Figure22. | IXIA LAN to WAN TCP connection(Small Packet) | 70 |
| Figure23. | IXIA LAN to GPON WAN UDP connection (Large Packet)..... | 70 |
| Figure24. | IXIA GPON WAN to LAN UDP connection (Large Packet)..... | 71 |
| Figure25. | IXIA LAN to GPON WAN UDP connection(Small Packet) | 72 |
| Figure26. | IXIA GPON WAN to LAN UDP connection(Small Packet) | 72 |

List of Tables

| | | |
|----------|--|----|
| Table1. | List of RTK RG APIs | 18 |
| Table2. | List of RTK RG APIs access Hardware Table..... | 19 |
| Table3. | Example: hook rtk_rg_naptExtPortGet..... | 25 |
| Table4. | Example: hook rtk_rg_naptExtPortFree | 25 |
| Table5. | Example: rtk_rg_initParam_set (Default)..... | 27 |
| Table6. | Example: rtk_rg_initParam_set (Customise)..... | 28 |
| Table7. | Example: rtk_rg_lanInterface_add | 29 |
| Table8. | Example: rtk_rg_wanInterface_add | 30 |
| Table9. | Example: rtk_rg_staticInfo_set(for LiteRomeDriver ARP set by USER) | 31 |
| Table10. | Example: rtk_rg_staticInfo_set(for RomeDriver ARP auto learning).... | 32 |
| Table11. | Example: rtk_rg_macEntry_add | 33 |
| Table12. | Example: rtk_rg_arpEntry_add..... | 33 |
| Table13. | Example: rtk_rg_naptConnection_add | 33 |
| Table14. | Example: rtk_rg_wanInterface_add (with Port Binding)..... | 34 |
| Table15. | Example: rtk_rg_vlanBinding_add..... | 35 |
| Table16. | List of ACL Action Type..... | 36 |
| Table17. | List of ACL QoS Actions | 36 |
| Table18. | List of ACL Patterns..... | 37 |
| Table19. | Example: rtk_rg_aclFilterAndQos_add..... | 37 |
| Table20. | Example: rtk_rate_shareMeter_set | 38 |
| Table21. | Example: rtk_rate_portEgrBandwidthCtrlRate_set | 38 |
| Table22. | Example: rtk_rate_portIgrBandwidthCtrlRate_set | 39 |
| Table23. | Example: rtk_qos_schedulingQueue_set..... | 40 |
| Table24. | Example: rtk_rg_qosDscpRemapToInternalPri_set..... | 40 |
| Table25. | Example: rtk_rg_qosPortBasedPriority_set..... | 41 |
| Table26. | Example: rtk_l2_portLimitLearningCnt_set..... | 41 |
| Table27. | Example: rtk_rg_macFilter_add | 42 |
| Table28. | Example: rtk_rg_algApps_set | 42 |
| Table29. | Example: rtk_rg_virtualServer_add | 43 |
| Table30. | Example: rtk_rg_dmzHost_set | 43 |
| Table31. | Example: rtk_rg_upnpConnection_add | 44 |
| Table32. | Example: rtk_rg_multicastFlow_add | 44 |
| Table33. | Example: rtk_rg_stormControl_add..... | 45 |
| Table34. | Example: rtk_rg_portMirror_set..... | 45 |
| Table35. | Example: rtk_rg_phyPortForceAbility_set | 46 |

| | | |
|----------|---|----|
| Table36. | Example: rtk_rg_dosPortMaskEnable_set | 46 |
| Table37. | Example: rtk_rg_dosType_set..... | 46 |
| Table38. | Example: rtk_rg_dosFloodType_set..... | 47 |
| Table39. | NIC RX Descriptor for CPU Tag | 49 |
| Table40. | List of Diag Command second keywords | 54 |
| Table41. | List of Diag Command Third keywords..... | 55 |

Realtek
CONFIDENTIAL
for Loso Technology, Inc

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 Stackand, 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 Protocol 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 port 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 Protocol Stack when it can not be processed
4. UI or User Application Interface: RomeDriver and LiteRomeDriver provides 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 Protocol Stack, such as Rouing Table, and call User Application, such as DHCP Client and PPPoE Client.

Below figure shows the architecture. 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, such 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.

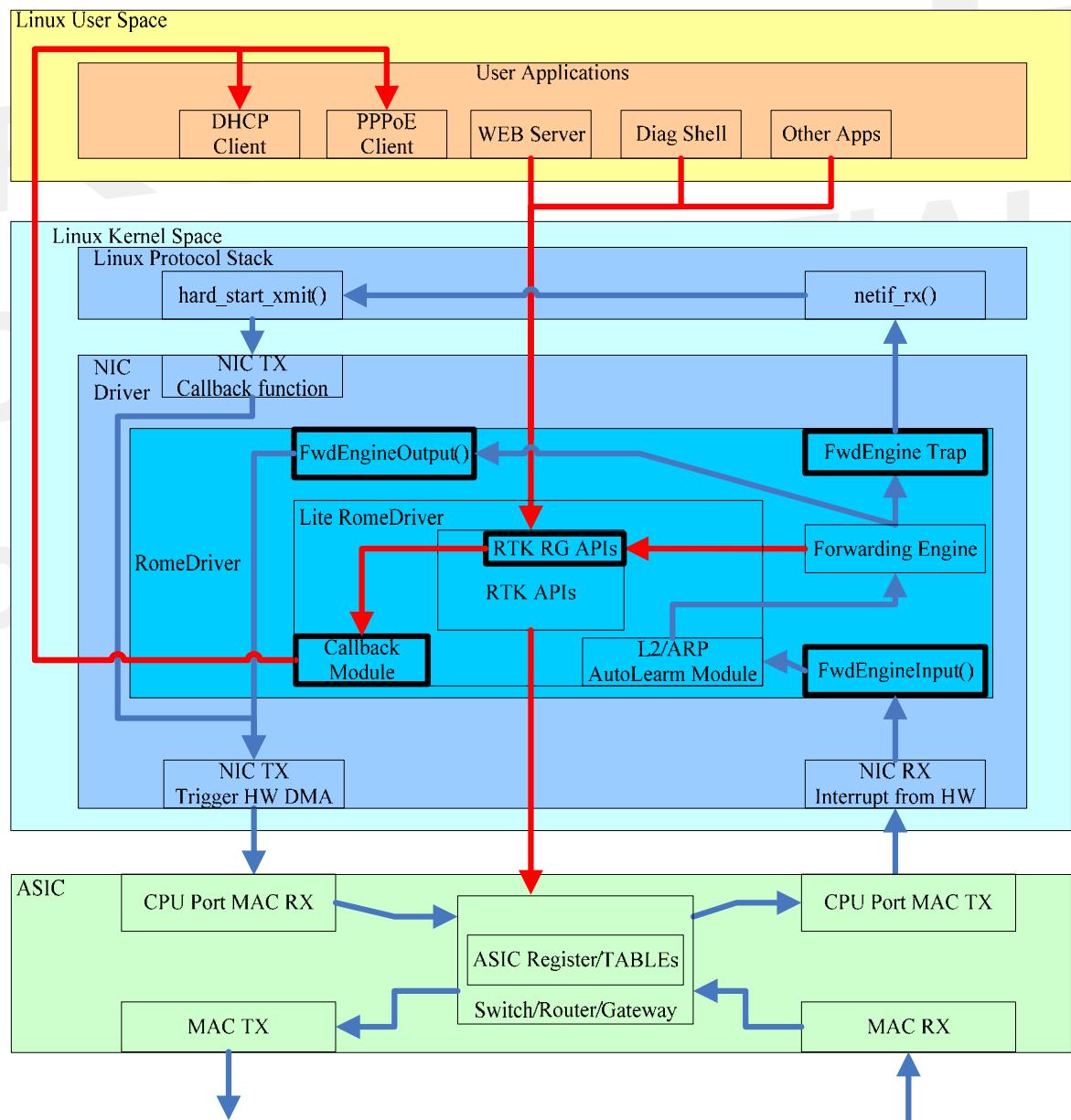


Figure1. RomeDriver Architecture

1.2. Lite RomeDriver Architecture

Lite-RomeDriver is a lighter version of RomeDriver without Forwarding Engine for which users have to call Protocol 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 Protocol Stack by RTK RG APIs. Besides, Layer2 MAC & ARP Entry shoule be maintainind by L2/ARP AutoLearning Module automatically or maintainind by user calling RTK RG APIs to keep Layer2 MAC & ARP synchronous between Hardware and Protocol Stack. If L2/ARP AutoLearning Module is enabled, the Lite RomeDriver Input function should be registered in NIC RX Interface.

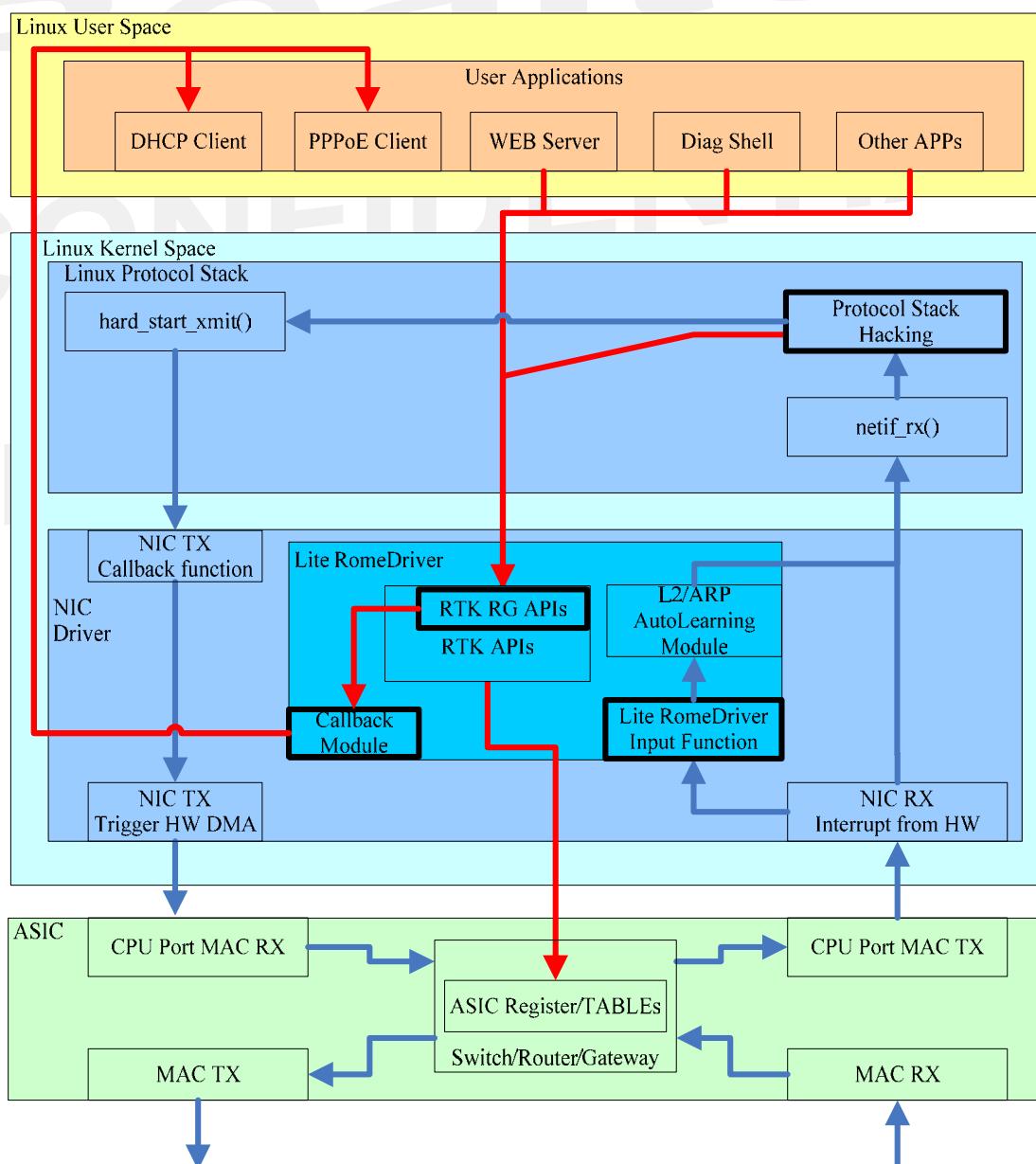


Figure2. Lite RomeDriver Architecture

2. RTK RG APIs

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

| Category | R | L | 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. |
| 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_pppoeClientInfoBeforeDial_set | Set WAN Type:PPPoE dial 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 | Find a VLAN Binding Entry which is added. |
| ALG | ● | ○ | rtk_rg_algApps_set | Set the starting ALG Application type. |
| | | | rtk_rg_algApps_get | Set the current ALG Application type. |
| Virtual Server | ● | ○ | rtk_rg_virtualServer_add | Add a Virtual Server Rule. |

| | | | | |
|--------------------------|----------------------------------|----------------------------------|--|---|
| | | | <code>rtk_rg_virtualServer_del</code> | Delete a Virtual Server Rule. |
| | | | <code>rtk_rg_virtualServer_find</code> | Find a Virtual Server Rule. |
| DMZ | <input checked="" type="radio"/> | <input type="radio"/> | <code>rtk_rg_dmzHost_set</code> | Set DMZ Host to assigned WAN Interface. |
| | | | <code>rtk_rg_dmzHost_get</code> | Get DMZ Host information from assigned WAN Interface. |
| ACL and Qos Remarking | <input checked="" type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_aclFilterAndQos_add</code> | Add an ACL Rule(include QoS Remarking). |
| | | | <code>rtk_rg_aclFilterAndQos_del</code> | Delete an ACL Rule(include QoS Remarking). |
| | | | <code>rtk_rg_aclFilterAndQos_find</code> | Find an ACL Rule which is added. |
| MAC Filter | <input checked="" type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_macFilter_add</code> | Add a MAC Filter Entry. |
| | | | <code>rtk_rg_macFilter_del</code> | Delete a MAC Filter Entry. |
| | | | <code>rtk_rg_macFilter_find</code> | Find a MAC Filter Entry which is added. |
| URL Filter | <input checked="" type="radio"/> | <input type="radio"/> | <code>rtk_rg_urlFilterString_add</code> | Add a URLFilter Entry. |
| | | | <code>rtk_rg_urlFilterString_del</code> | Delete a URLFilter Entry. |
| | | | <code>rtk_rg_urlFilterString_find</code> | Find a URLFilter Entry which is added. |
| UPnP | <input checked="" type="radio"/> | <input type="radio"/> | <code>rtk_rg_upnpConnection_add</code> | Add an uPnP connection. |
| | | | <code>rtk_rg_upnpConnection_del</code> | Delete an uPnP connection. |
| | | | <code>rtk_rg_upnpConnection_find</code> | Find an uPnP connection which is added. |
| Multicast | <input type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_multicastFlow_add</code> | Add a Multicast connection. |
| | | | <code>rtk_rg_multicastFlow_del</code> | Delete a Multicast connection. |
| | | | <code>rtk_rg_multicastFlow_find</code> | Find a Multicast connection which is added. |
| NAPT | <input type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_naptConnection_add</code> | Add a NAPT connection. |
| | | | <code>rtk_rg_naptConnection_del</code> | Delete a NAPT connection. |
| | | | <code>rtk_rg_naptConnection_find</code> | Find a NAPT connection which is added. |
| MAC | <input type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_macEntry_add</code> | Add a MAC Entry. |
| | | | <code>rtk_rg_macEntry_del</code> | Delete a MAC Entry. |
| | | | <code>rtk_rg_macEntry_find</code> | Find a MAC Entry which is added. |
| ARP | <input type="radio"/> | <input checked="" type="radio"/> | <code>rtk_rg_arpEntry_add</code> | Add an ARP Entry. |
| | | | <code>rtk_rg_arpEntry_del</code> | Delete an ARP Entry. |

| | | | |
|---------------------------|----------------------------------|--|---|
| | | rtk_rg_arpEntry_find | Find an ARP Entry which is added. |
| Neighbor | <input type="radio"/> | 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. |
| StormControl | <input checked="" type="radio"/> | 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. |
| SourceAddrLearningLimit | <input checked="" type="radio"/> | rtk_rg_softwareSourceAddrLearningLimit_set | Set SourceAddrLearningLimit |
| | | rtk_rg_softwareSourceAddrLearningLimit_get | Get SourceAddrLearningLimit which has been set. |
| PortMirror | <input checked="" type="radio"/> | rtk_rg_portMirror_set | Set portMirror |
| | | rtk_rg_portMirror_get | Get portMirror which has been set |
| Egress Bandwidth Control | <input checked="" type="radio"/> | rtk_rg_portEgrBandwidthCtrlRate_set | Set Egress Bandwidth Control rate |
| | | rtk_rg_portEgrBandwidthCtrlRate_get | Get Egress Bandwidth Control rate which has been set |
| Ingress Bandwidth Control | <input checked="" type="radio"/> | rtk_rg_portIgrBandwidthCtrlRate_set | Set Ingress Bandwidth Control rate |
| | | rtk_rg_portIgrBandwidthCtrlRate_get | Get Ingress Bandwidth Control rate which has been set |
| Physical Port | <input checked="" type="radio"/> | rtk_rg_phyPortForceAbility_set | Set Physical Port Ability |
| | | rtk_rg_phyPortForceAbility_get | Get Physical Port Ability which has been set |
| ShareMeter | <input checked="" type="radio"/> | rtk_rg_shareMeter_set | Set an ShareMeter Entry. |
| | | rtk_rg_shareMeter_get | Get an ShareMeter Entry which has been set |
| DOS | <input checked="" type="radio"/> | 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 |

Table1. List of RTK RG APIs

PS: In the Above field, R stands for RomeDriver API, L stands for Lite RomeDriver API. The gray field is not supported by API in this version.

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

| Category | Function Name | L 2 | V L | A R | N I | L 3 | N H | N B | B I | W A | I P | P P | N A | A C | C F |
|---------------------------|--------------------------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| System | rtk_rg_driverVersion_get | | | | | | | | | | | | | | |
| | rtk_rg_initParam_get | | | | | | | | | | | | | | |
| | rtk_rg_initParam_set | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| LAN Intf | rtk_rg_lanInterface_add | | ● | | ● | ● | | | | | | | | | |
| | rtk_rg_interface_del | | ● | | ● | ● | | | | | | | | | |
| | rtk_rg_intfInfo_find | | | | | | | | | | | | | | |
| WAN Intf/ Port Binding | rtk_rg_wanInterface_add | | ● | | ● | ● | ● | | ● | ● | | ● | | | |
| | rtk_rg_interface_del | | ● | | ● | ● | ● | | ● | ● | | ● | | | |
| | rtk_rg_intfInfo_find | | | | | | | | | | | | | | |
| WAN Type | rtk_rg_staticInfo_set | ● | | ● | | ● | | ● | | | | | | | |
| | rtk_rg_dhcpClientInfo_set | ● | | ● | | ● | | ● | | | | | | | |
| | rtk_rg_pppoeClientInfoBeforeDial_set | | | | | | | | | | | | | | |
| | rtk_rg_pppoeClientInfoAfterDial_set | ● | | ● | | ● | | ● | | | | ● | | | |
| VLAN Binding | rtk_rg_vlanBinding_add | | | | | | | | ● | | | | | | |
| | rtk_rg_vlanBinding_del | | | | | | | | ● | | | | | | |
| | rtk_rg_vlanBinding_find | | | | | | | | | | | | | | |
| ALG | rtk_rg_algApps_set | | | | | | | | | | | | | | |
| | rtk_rg_algApps_get | | | | | | | | | | | | | | |
| Virtual Server | rtk_rg_virtualServer_add | | | | | | | | | | | | | | |
| | rtk_rg_virtualServer_del | | | | | | | | | | | | | | |
| | rtk_rg_virtualServer_find | | | | | | | | | | | | | | |
| DMZ | rtk_rg_dmzHost_set | | | | | | | | | | | | | | |
| | rtk_rg_dmzHost_get | | | | | | | | | | | | | | |
| ACL and | rtk_rg_aclFilterAndQos_add | | | | | | | | | | | | ● | | |
| | rtk_rg_aclFilterAndQos_del | | | | | | | | | | | | ● | | |

| | | | | | | | | | | | | | | | | |
|----------------------|-----------------------------|---|---|--|--|--|--|---|--|--|--|--|--|---|---|--|
| Qos Remark ing | rtk_rg_aclFilterAndQos_find | | | | | | | | | | | | | | | |
| MAC Filter | rtk_rg_macFilter_add | ● | | | | | | | | | | | | | | |
| | rtk_rg_macFilter_del | ● | | | | | | | | | | | | | | |
| | rtk_rg_macFilter_find | | | | | | | | | | | | | | | |
| URL Filter | rtk_rg_urlFilterString_add | | | | | | | | | | | | | | ● | |
| | rtk_rg_urlFilterString_del | | | | | | | | | | | | | | ● | |
| | rtk_rg_urlFilterString_find | | | | | | | | | | | | | | | |
| UPnP | rtk_rg_upnpConnection_add | | | | | | | | | | | | | | | |
| | rtk_rg_upnpConnection_del | | | | | | | | | | | | | | | |
| | rtk_rg_upnpConnection_find | | | | | | | | | | | | | | | |
| Multicast | rtk_rg_multicastFlow_add | ● | | | | | | | | | | | | | | |
| | rtk_rg_multicastFlow_del | ● | | | | | | | | | | | | | | |
| | rtk_rg_multicastFlow_find | ● | | | | | | | | | | | | | | |
| NAPT | rtk_rg_naptConnection_add | | | | | | | | | | | | | ● | | |
| | rtk_rg_naptConnection_del | | | | | | | | | | | | | ● | | |
| | rtk_rg_naptConnection_find | | | | | | | | | | | | | ● | | |
| MAC | rtk_rg_macEntry_add | ● | | | | | | | | | | | | | | |
| | rtk_rg_macEntry_del | ● | | | | | | | | | | | | | | |
| | rtk_rg_macEntry_find | | | | | | | | | | | | | | | |
| ARP | rtk_rg_arpEntry_add | ● | ● | | | | | | | | | | | | | |
| | rtk_rg_arpEntry_del | ● | ● | | | | | | | | | | | | | |
| | rtk_rg_arpEntry_find | | | | | | | | | | | | | | | |
| Neighbor | rtk_rg_neighborEntry_add | ● | | | | | | ● | | | | | | | | |
| | rtk_rg_neighborEntry_del | ● | | | | | | ● | | | | | | | | |
| | rtk_rg_neighborEntry_find | | | | | | | | | | | | | | | |

Table2. List of RTK RG 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/uClinix-dist
```

User should acquire source code package from Realtek FAE.

4. Build Code

Following command shows how to build source code to Image:

Read the default configuration setting:

```
$ cd uClinix-dist
$ make -f LDLMakefile preconfig2630_luna_dual_linux
```

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
```

5. NAPT External Port Collision Prevention

In Current SDK, Forwarding Engine can not support some advanced flow connection, such as USER-DEFINED-ALG. So, these kind of connection packet will be trapped to Protocol-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 connection, no matter forwardingEngine or OS take care of this, it must choose an external port. In current 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 transplanting in different OS, and the maintenance 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:

```
--- Choose a Vendor/Product combination.  
(Realtek/RTL8670) Vendor/Product  
[*] Dual Linux  
--- Kernel is linux-2.6.x  
(2.6.30) LINUX Kernel 2.6 Version  
[*] Use RSDK-Wrapper Toolchain  
    RSDK Toolchain Path for Master: "/share/r...  
    RSDK Toolchain Path for Slave: "/share/r...  
[*] Customize Kernel Settings (NEW)  
[ ] Customize Vendor/User Settings  
[ ] Config busybox-1.12.4  
[*] Customize Slave Kernel Settings  
[*] Customize Slave Vendor/User Settings  
[ ] Customize Slave Busybox-1.12.4 Setting  
[ ] Tiny Image  
[*] VoIP IPC DSP Architecture  
[ ] VoIP Turnkey Call Manager  
[ ] Use Preloader Parameters  
[*] Luna trunk only  
--- RTL8686 IPC Shared Memory Setting  
(00000000) RTL8686 CPU MEM Base  
(02000000) RTL8686 CPU MEM Size  
(02000000) RTL8686 IPC MEM Base  
(00100000) RTL8686 IPC MEM Size  
(02100000) RTL8686 DSP MEM Base  
(01F00000) RTL8686 DSP MEM Size
```

Figure3. Enable NAPT External Port Collision Prevention, step1

Step2: Enter to “Device Drivers”

```

System Configuration --->
Kernel type --->
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
PCI Bus options --->
Power management options --->
[*] Networking support --->
[*] Device Drivers --->
File systems --->
Kernel hacking --->
Security options --->
<*> Cryptographic API --->
Library routines --->
RTK VoIP Suite --->
---
Load an Alternate Configuration File
Save an Alternate Configuration File

```

Figure4. Enable NAPT External Port Collision Prevention, step2

Step3: Enter to “Realtek RG support”

```

Generic Driver Options --->
< > Connector - unified userspace <-> kernelspace linker --->
<*> Memory Technology Device (MTD) support --->
< > Parallel port support --->
[ ] Block devices --->
[ ] Misc devices --->
SCSI device support --->
< > Serial ATA (prod) and Parallel ATA (experimental) drivers --->
[ ] Multiple devices driver support (RAID and LVM) --->
[*] Network device support --->
[*] Realtek RG support --->
[ ] ISDN support --->
< > Telephony support --->
Input device support --->
Character devices --->

```

Figure5. Enable NAPT External Port Collision Prevention, step3

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

```

--- Realtek RG support
    Realtek RG API Type Selection (RomeDriver APIs) --->
[*]  Realtek RG debug tools
[*]  Realtek RG NAPT TCP auto learning by Forwarding Engine
[+]  Realtek RG NAPT external port collision prevention
[*]  Realtek RG NAPT UPNP support by Forwarding Engine
[*]  Realtek RG NAPT Virtual Server support by Forwarding Engine
[*]  Realtek RG NAPT DMZ support by Forwarding Engine
[*]  Realtek RG ARP auto learning by Forwarding Engine
[*]  Realtek RG Neighbor auto learning by Forwarding Engine
[*]  Realtek RG Master CPU WLAN HWNAT acceleration
[*]  Realtek RG NAPT auto age out by Forwarding Engine
[*]  Realtek RG ARP auto age out by Forwarding Engine
[*]  Realtek RG callback function Enable

```

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 connection 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.

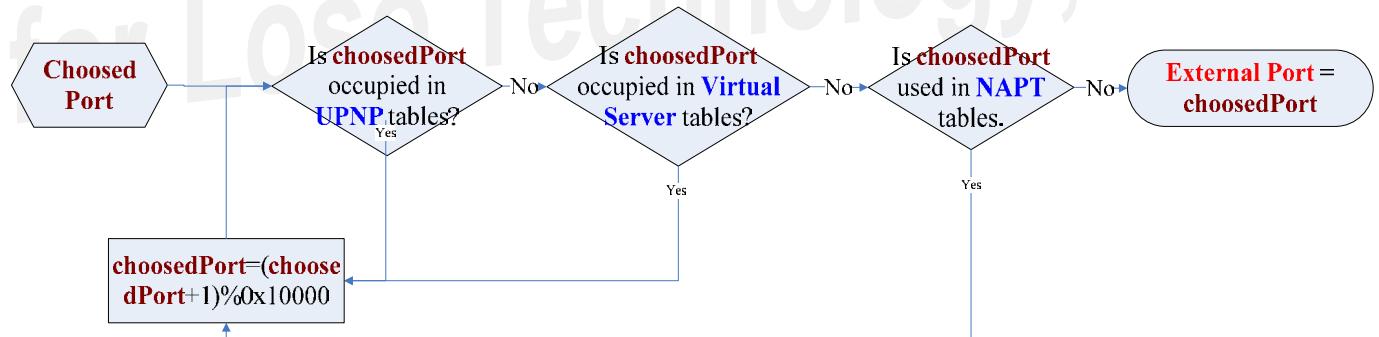


Figure7. NAPT External Port choosing flow.

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 connection.

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

Prevention Mechanism in Linux 2.6.30

Here is an example for hooking the NAPT External Port Collision Prevention Mechanism into Linux Kernel 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 Protocol-Stack sample code as following:

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

```
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:
#ifndef CONFIG_RG_NAPT_PORT_COLLISION_PREVENTION
    if(maniptype == IP_NAT_MANIP_SRC) //SNAT
    {
        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;
        isTCP = (tuple->dst.protonum==IPPROTO_TCP)?1:0;
        pPort = &tuple->src.u.all;
        while(pPort)
        {
            if(i++>65535) break;
            rtk_rg_naptExtPortGet(isTCP,pPort);
            if(nf_nat_used_tuple(tuple, ct))
            {
                rtk_rg_naptExtPortFree(isTCP,*pPort);
                *pPort=(*pPort+1)%0x10000;
                continue;
            }
            break;
        }
    }
    else //DNAT
    {
        //Do nothing!
    }
#endif
    rcu_read_unlock();
}
```

Table3. 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();

#ifndef CONFIG_NF_NAPT
    return;
#endif

#ifdef CONFIG_NF_NAPT
    #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);
}
```

Table4. Example: hook rtk_rg_naptExtPortFree

6. (Lite)RomeDriver Initiation

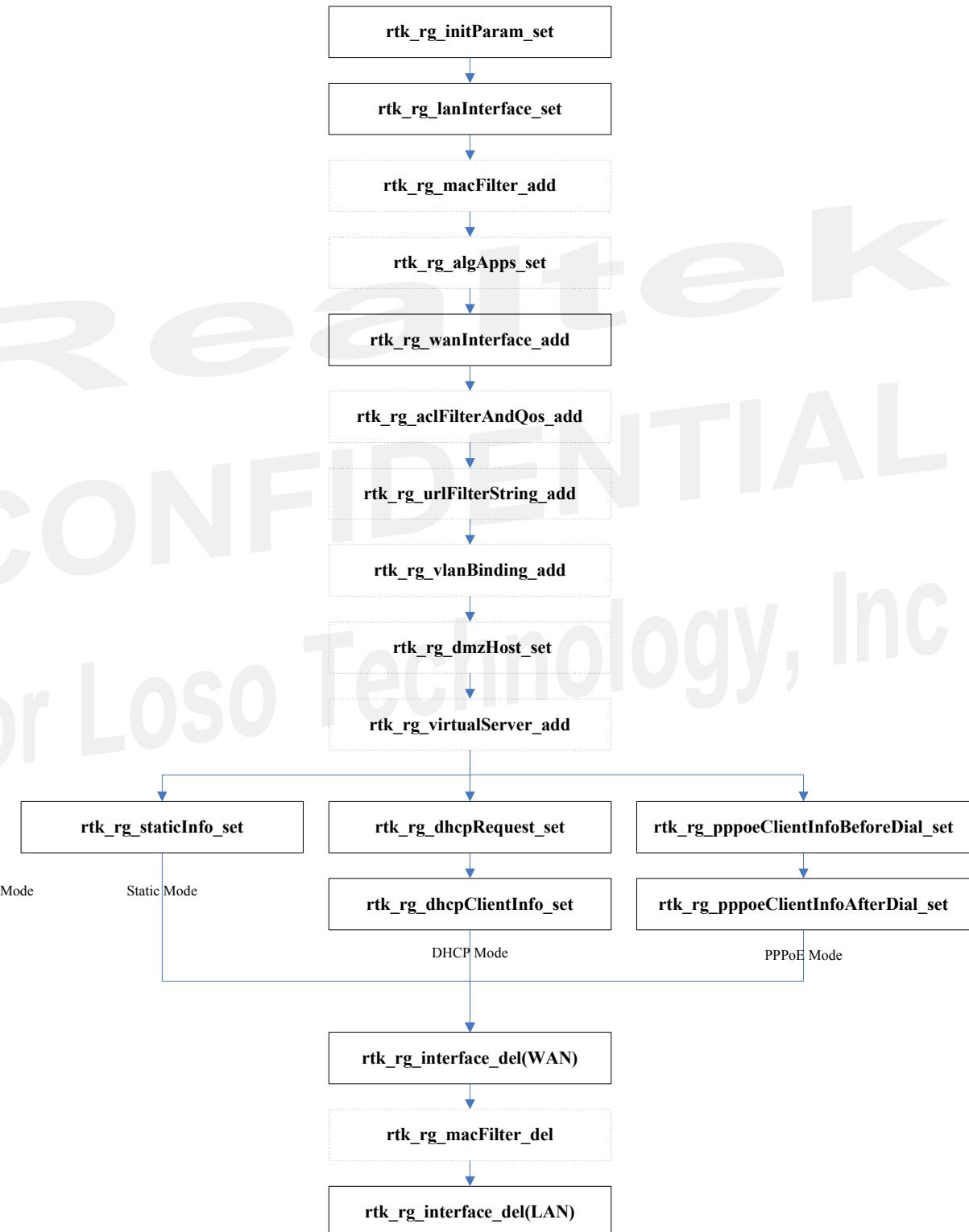


Figure8. RomeDriver Initiation Flow Chart

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

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 Configuration

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;
```

Table5. Example: rtk_rg_initParam_set (Default)

If user wants to control the initialization, the parameter init_param.igmpSnoopingEnable can use to enable/disable the Igmpsnooping, and the parameter init_param.macBasedTagDecision can use to enable/disable the MacBasedVlan feature. 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 & MacBasedVlan and register a callback function in init parameters for notifying 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;
```

Table6. Example: rtk_rg_initParam_set (Customise)

Following description explain the usage of parameter init_param.macBasedTagDecision:

This parameter use to control the Hardware DMAC_TO_CVID(Mac Based Vlan) enable/disable.

However, according to the Hardware vlan decision behavior, some API parameter configuration rules should be followed either enable or disable this feature:

While init_param.macBasedTagDecision=DISABLED:

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 Interfce, which bring up by *rtk_rg_wanInterface_add*, is BridgeWan, the VLAN must same as Lan Interface.

While init_param.macBasedTagDecision=ENABLED:

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 not be set as IVL.

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_LENGTH=2001::1/64 , MTU=1500.

```
rtk_rg_lanIntfConf_t lan_info;
int lanIdx;
memset(&lan_info,0,sizeof(lan_info));
lan_info.ip_version=2;
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.ip_addr=0xc0a80101; //192.168.1.1
```

```

lan_info.ip_network_mask=0xffffffff00;
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));
if(rtk_rg_lanInterface_add(&lanInfo,&lanIdx)) return -1;

```

Table7. 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.

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

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 set *init_param.macBasedTagDecision=ENABLED* while calling *rtk_rg_initParam_set*, then only *SVL* can be set. 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, 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.

```
rtk_rg_wanIntfConf_t wan_info;
int wanIdx;
memset(&wan_info, 0, sizeof(wan_info));
wan_info.egress_vlan_id=8;
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(rtк_rg_wanInterface_add(&wan_info,&wanIdx)) return -1;
```

Table8. Example: rtk_rg_wanInterface_add

`wan_info.port_binding_mask.portmask` is limited to 0x0 if user set

`init_param.macBasedTagDecision=DISABLED` while calling `rtk_rg_initParam_set`. And

`wan_info.isIVL` is limited to SVL if user set `init_param.macBasedTagDecision=ENABLED` while calling `rtk_rg_initParam_set`.

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=0xffffffff;
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;
```

Table9. 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=0xfffffffffe;
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;

```

Table10. 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;

```

Table11. 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;

```

Table12. 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;

```

Table13. Example: rtk_rg_naptConnection_add

6.2. Gateway Advanced Configuration

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 Interface 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_Inf_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(rtк_rg_wanInterface_add(&wanIntf,&wanIdx)) return -1;
```

Table14. 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;
```

Table15. 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 determined to take VLAN Tag=8 or not by wlanIntf.egress_vlan_tag_on. When packet is sent from 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 / QoS Remarking

rtk_rg_aclFilterAndQos_add can be used to filter under particular condition, it also can trap packet to CPU conducting 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* 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 handled 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. <i>qos_actions</i> is relevant only when <i>action_type</i> 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 Protocol Stack in Kernel directly. |
|----------------------------|---|

Table16. List of ACL Action Type

qos_actions field support following activities:

| | |
|--|--|
| ACL_ACTION_1P_REMARKING_BIT | Transformed CVLAN Priority, 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. |

Table17. List of ACL QoS Actions

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

The filtered conditions and filtered condition glossary are defined 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_etheretype |
| 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 |

Table18. List of ACL Patterns

Below is a typical scenario 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.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;
```

Table19. Example: rtk_rg_aclFilterAndQos_add

6.2.4. Rate Limit

6.2.4.1. 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 limited 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 code 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.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;
```

Table20. Example: rtk_rate_shareMeter_set

6.2.4.2. 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 bandwidth and per-egress port & queue share meter bandwidth are all set for the same purpose in the same time, the smallest one will be referred 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;
```

Table21. Example: rtk_rate_portEgrBandwidthCtrlRate_set

6.2.4.3. 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;
```

Table22. Example: rtk_rate_portIgrBandwidthCtrlRate_set

6.2.5. QoS Strict Priority / WFQ

The RTK 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 of 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 scenario 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 sequential 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;

```

Table23. Example: rtk_qos_schedulingQueue_set

6.2.5.1. 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;

```

Table24. Example: rtk_rg_qosDscpRemapToInternalPri_set

6.2.5.2. 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(rt_k_rg_qosInternalPriDecisionByWeight_set (weight)) return -1;
if(rt_k_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT0,0)) return -1;
if(rt_k_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT1,1)) return -1;
if(rt_k_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT2,2)) return -1;
if(rt_k_rg_qosPortBasedPriority_set (RTK_RG_MAC_PORT3,3)) return -1;
```

Table25. Example: rtk_rg_qosPortBasedPriority_set

6.2.6. 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(rt_k_l2_portLimitLearningCnt_set(RTK_RG_MAC_PORT0,10)) return -1;
```

Table26. Example: rtk_l2_portLimitLearningCnt_set

6.2.7. 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;
```

Table27. Example: rtk_rg_macFilter_add

6.2.8. 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;
```

Table28. Example: rtk_rg_algApps_set

6.2.9. 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 Interfceae 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;

```

Table29. Example: rtk_rg_virtualServer_add

6.2.10. 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;

```

Table30. Example: rtk_rg_dmzHost_set

6.2.11. 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;

```

Table31. Example: rtk_rg_upnpConnection_add

6.2.12. 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.src_intf_idx = 1;
mcFlow.port_mask.portmask = (1<<RTK_RG_MAC_PORT0)|(1<<RTK_RG_MAC_PORT1);
if(rtk_rg_multicastFlow_add(&mcFlow,&flowIdx))
    return -1;

```

Table32. Example: rtk_rg_multicastFlow_add

6.2.13. 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;

```

Table33. Example: rtk_rg_stormControl_add

6.2.14. 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;

```

Table34. Example: rtk_rg_portMirror_set

6.2.15. 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;
```

Table35. Example: rtk_rg_phyPortForceAbility_set

6.2.16. Denial-of-service (D.O.S)

rtk_rg_dosPortMaskEnable_set is used to enable the DoS feature for each port. After that, we can use rtk_rg_dosType_set to determine 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 be 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;
```

Table36. 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;
```

Table37. 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_DROPOUT))
    return -1;
```

Table38. 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 | E | FS | LS | I | L | KEEP | B | C | v | D | 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 |
| Source Extension Port(3 bits) (extspa) | Tx destination port number (6 bits) (Tx portmask) | | Tx destination stream ID (7 bits) (PON stream ID) | | | Reserved(13bits) | | | L34 | KEE | T | P | P | | Offset 12 | |
| LGSEN | Large-Send MTU value (11 bits) | | | | RSVD (20 bits) | | | | | | | | | | Offset 16 | |

Figure9. 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 | C R C E r r | I V 4 C S F | I P C C S F | L 4 C DF | IP FRAG | PPP oE tag | RWT | PktType (4bits) | L3 routing | Orig Format | P C T R L | RSVD (2bits) (max: 0x800) | Data_Length (12bits) (max: 0x800) | Offset 0 |
| RxBuff (32 bits) | | | | | | | | | | | | | | | | | |
| Cputag PTP in CPU Tag exist SVLAN Tag exist RSVD (2bits) PON stream id(7bits) RSVD (3bits) CTAVA CVLAN TAG (16 bits) | | | | | | | | | | | | | | | | Offset 4 | |
| Source Port Number (5 bits) Destination port mask (6 bits) (extension port mask) Reason (8 bits) Internal priority (3 bits) Extension port TTL-1 (5 bits) RSVD (5 bits) | | | | | | | | | | | | | | | | Offset 8 | |
| Source Port Number (5 bits) Destination port mask (6 bits) (extension port mask) Reason (8 bits) Internal priority (3 bits) Extension port TTL-1 (5 bits) RSVD (5 bits) | | | | | | | | | | | | | | | | Offset 12 | |

Figure10. NIC RX Descriptor Format

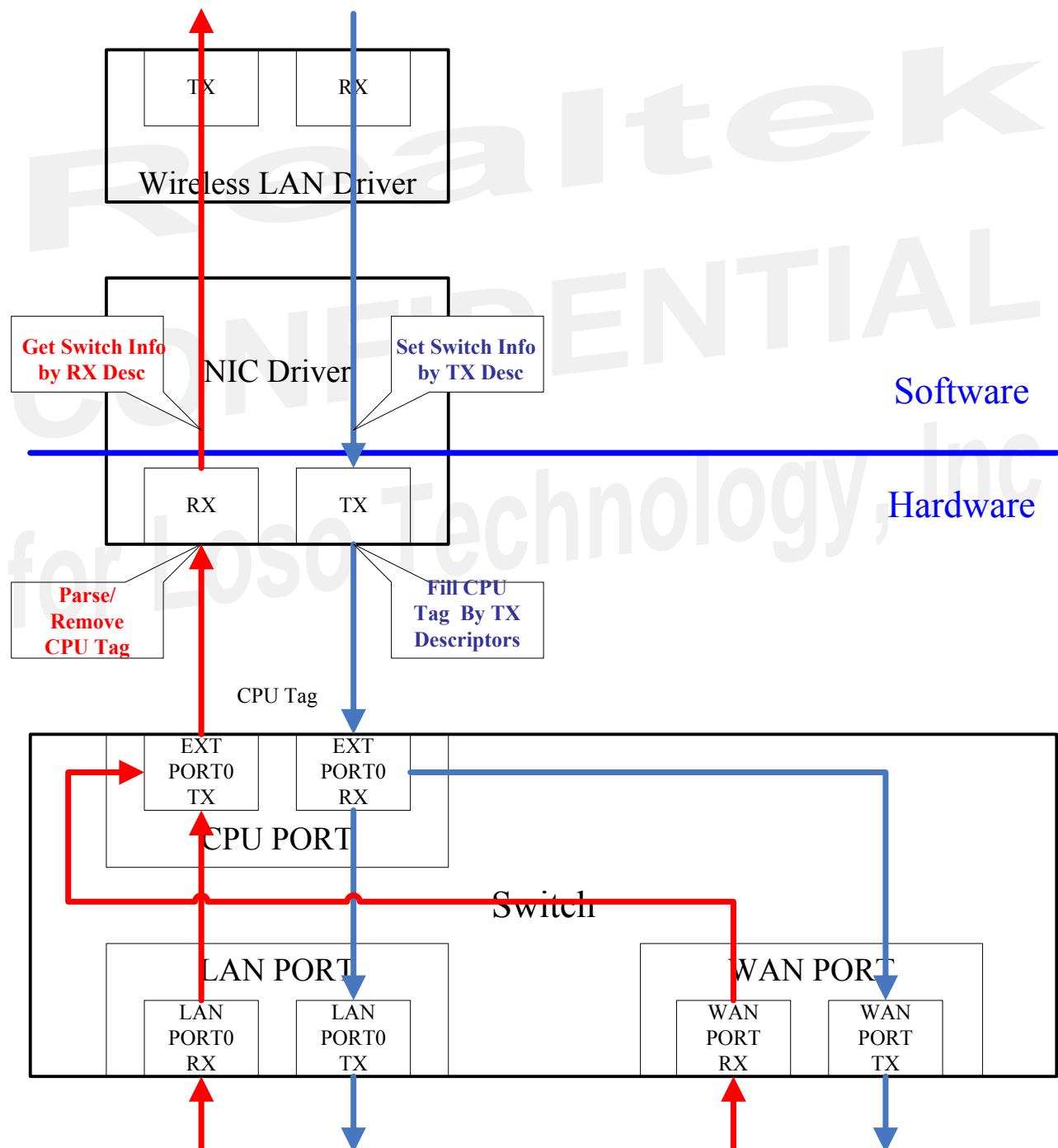
Upon receipt of a packet from Switch in NIC, the CPU Tag information will be parsed and port to NIC RX Descriptor fields as above figure shown, After that CPU Tag will be removed. The Software can acquire informations from Switch by RX Descriptor fields: L3 Routing, Orig Format, Cputag, Source Port Number, 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, Hardware will determine |

| | |
|--|--|
| | whether 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. |
|--|--|

Table39. 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 architecture is as following figure:

**Figure11. Wireless LAN Hardware Forward**

8. Test by Web UI configuration

the machine is turned on , the machine will take the last UI configuration from Flash, and Linux Kernel 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 Gateway 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.

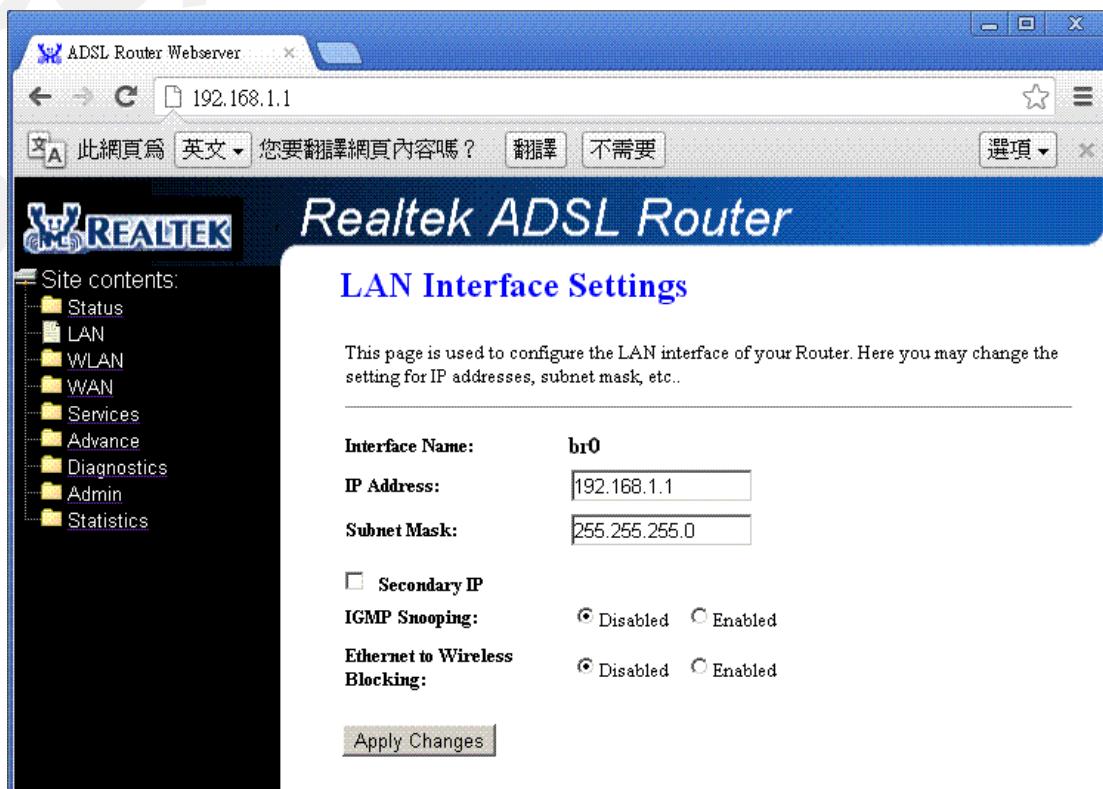


Figure12. Web UI:LAN Interface Setting

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

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.

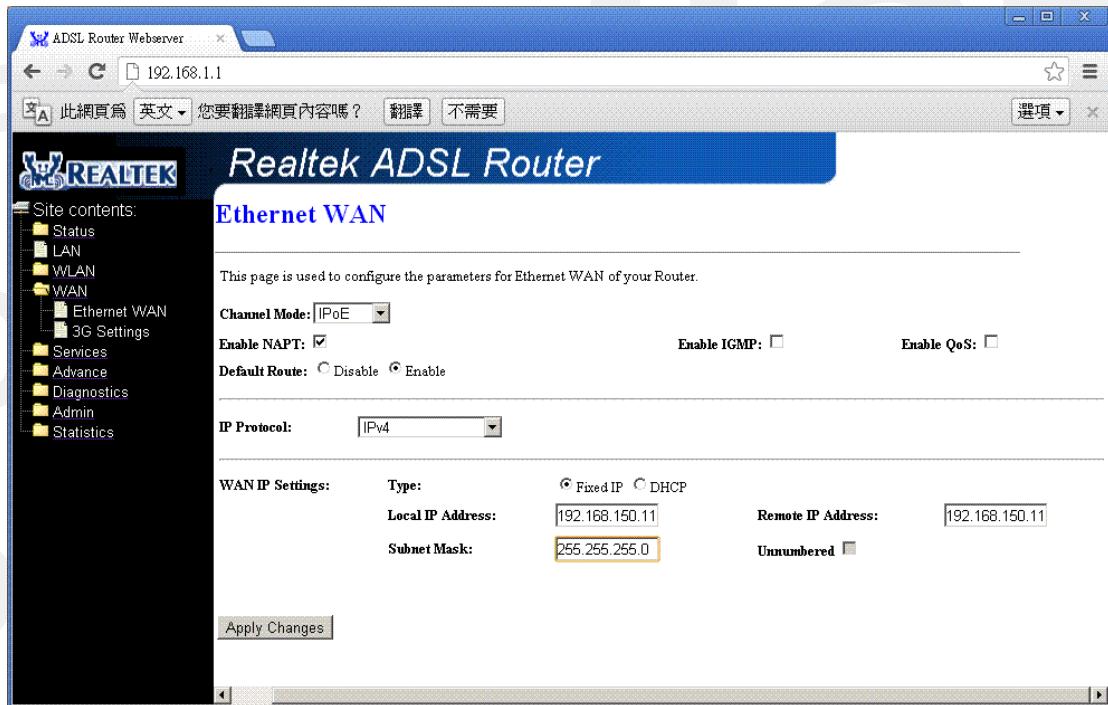


Figure13. Web UI:Ethernet WAN Interface Setting

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 system, and log in with account: **admin** and password: **system**.

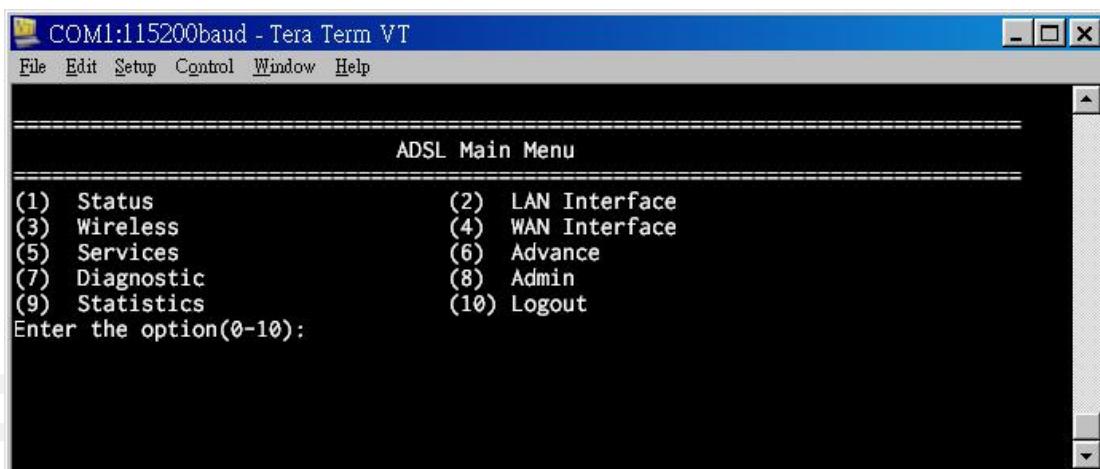


Figure14. UART: ADSL Main Menu

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.

```
COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

RTK.0> rg show <input [tab]>
acl-filter
arp-entry
binding
callback
cylan
dhcpClientInfo
lan-intf
mac-entry
macfilter
napt-connection
neighbor-entry
pppoeClientInfoAfterDial
pppoeClientInfoBeforeDial
stormControl
upnpConnection
virtualServer
wan-intf
wan-intf-static-info

RTK.0> rg show
```

Figure15. DiagShell tips for next keyword

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

```
COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

RTK.0> rg show pppo
^Incomplete command

RTK.0> rg show pppoeClientInfo
<input [tab]>
```

Figure16. 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 such 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 |

Table40. 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 |

Table41. 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:PO, 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 configuring LAN/WAN Interface of protocol stack can be referenced by [Sec. Set Lan/Wan Configuration of Protocol 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 Protocol 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
```

User can register his own callback functions without using default callback functions. but need to remember where he stores it in Kernal memory (ex, 0xffffffff). 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 0xffffffff00
```

```
RTK.0>rg set callback arpDelByHwCallBack 0xffffffff01
RTK.0>rg set callback bindingAddByHwCallBack 0xffffffff02
RTK.0>rg set callback bindingDelByHwCallBack 0xffffffff03
RTK.0>rg set callback initByHwCallBack 0xffffffff04
RTK.0>rg set callback interfaceAddByHwCallBack 0xffffffff05
RTK.0>rg set callback interfaceDelByHwCallBack 0xffffffff06
RTK.0>rg set callback macAddByHwCallBack 0xffffffff07
RTK.0>rg set callback macDelByHwCallBack 0xffffffff08
RTK.0>rg set callback naptAddByHwCallBack 0xffffffff09
RTK.0>rg set callback naptDelByHwCallBack 0xffffffff0a
RTK.0>rg set callback neighborAddByHwCallBack 0xffffffff0b
RTK.0>rg set callback neighborDelByHwCallBack 0xffffffff0c
RTK.0>rg set callback pppoeBeforeDiagByHwCallBack 0xffffffff0d
RTK.0>rg set callback routingAddByHwCallBack 0xffffffff0e
RTK.0>rg set callback routingDelByHwCallBack 0xffffffff0f
RTK.0>rg set callback v6RoutingAddByHwCallBack 0xffffffff10
RTK.0>rg set callback v6RoutingDelByHwCallBack 0xffffffff11
RTK.0>rg init callback
```

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

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

The screenshot shows a terminal window titled "COM1:115200baud - Tera Term VT". The menu bar includes File, Edit, Setup, Control, Window, and Help. The main window displays the output of the command "# cat proc/rg/callback". The output lists various callback functions and their corresponding memory addresses:

| [Address] | [Function] |
|------------|--------------------------------------|
| 0x8026a97c | _rtk_rg_initParameterSetByHwCallBack |
| 0x80266bd8 | _rtk_rg_interfaceAddByHwCallBack |
| 0x80266580 | _rtk_rg_interfaceDelByHwCallBack |
| 0x802639b4 | _rtk_rg_pppoeBeforeDiagByHwCallBack |
| 0x80265084 | _rtk_rg_dhcpRequestByHwCallBack |
| 0x802638a0 | _rtk_rg_arpAddByHwCallBack |
| 0x802638a8 | _rtk_rg_arpDelByHwCallBack |
| 0x802638b0 | _rtk_rg_macAddByHwCallBack |
| 0x802638b8 | _rtk_rg_macDelByHwCallBack |
| 0x802638c0 | _rtk_rg_naptAddByHwCallBack |
| 0x802638c8 | _rtk_rg_naptDelByHwCallBack |
| 0x802638d0 | _rtk_rg_routingAddByHwCallBack |
| 0x802638d8 | _rtk_rg_routingDelByHwCallBack |
| 0x802638e0 | _rtk_rg_bindingAddByHwCallBack |
| 0x802638e8 | _rtk_rg_bindingDelByHwCallBack |

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 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 egress-vlan-pri 0 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 255.255.255.254 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 255.255.255.254 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[45] 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 45 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>
```

```
COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/netif
>>ASIC Netif Table:

[0]-vid[2] 00:e0:4c:86:70:01 L3/4 HW acc enabled
    7 MAC Addresses, MTU 1500 Bytes
        Untag member ports:0 1 2 3 4 5 6
        Active member ports:0 1 4 6

[1]-vid[8] 00:e0:4c:86:70:02 L3/4 HW acc enabled
    7 MAC Addresses, MTU 1500 Bytes
        Untag member ports:0 1 4 5 6
        Active member ports:5 6

#
```

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

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

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/l3
>>L3 Routing Table:
[0] 192.168.1.1/24 NETIF(0) LAN
    [ARP PROCESS]: ARPSTA(0) ARPEND(63)

[1] 192.168.150.116/31 NETIF(1) WAN
    [ARP PROCESS]: ARPSTA(64) ARPEND(64)

[2] 0.0.0.0/0 NETIF(1) WAN
    [CPU PROCESS]

[3] 0.0.0.0/0 NETIF(1) WAN
    [CPU PROCESS]

[4] 0.0.0.0/0 NETIF(1) WAN
    [CPU PROCESS]

[5] 0.0.0.0/0 NETIF(1) WAN
    [CPU PROCESS]

[6] 0.0.0.0/0 NETIF(1) WAN
    [CPU PROCESS]

[7] 0.0.0.0/0 NETIF(1) WAN
    [NxtHop PROCESS]: NHSTA(0) NHHNUM(1) NHNXT(0) NHALGO(PER-SIP) IPDOMAIN(6)
)
#

```

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 >

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/ip
>>IP Table:
[0] (Invalid)
[1] intip(0.0.0.0) extip(192.168.150.116) type(NAPT) nhIdx(0) PriValid(0)
Priority(0)
[2] (Invalid)
[3] (Invalid)
[4] (Invalid)
[5] (Invalid)
[6] (Invalid)
[7] (Invalid)
#

```

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 >

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/vlan
>>ASIC VLAN Table:

-- VID[1] --
Member Ports:0x7f
Extension Member Ports:0x3f
Untag Member Ports:0x7f
FID: 2, IVL_SVL: SVL
Based Priority: disable, 0
Extension Ports: 0 1 2 3 4 5

-- VID[2] --
Member Ports:0x53
Extension Member Ports:0x1
Untag Member Ports:0x7f
FID: 2, IVL_SVL: SVL
Based Priority: disable, 0
Extension Ports: 0

-- VID[8] --
Member Ports:0x60
Extension Member Ports:0x1
Untag Member Ports:0x73
FID: 1, IVL_SVL: IVL
Based Priority: disable, 0
Extension Ports: 0

-- VID[9] --
Member Ports:0x73
Extension Member Ports:0x1
Untag Member Ports:0x73
FID: 2, IVL_SVL: IVL
Based Priority: disable, 0
Extension Ports: 0
#

```

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 >

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/arp
>>Arp Table:
[ 34 ] : 192.168.1.34      -> L2:140
[257] : 192.168.150.117   -> L2:672
#

```

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

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/nh
>>ASIC Next Hop Table:
[0] type(ethernet, Non-Keep) IFIdx(1) pppoeIdx(1) nextHop(168,0)
#
#

```

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

COM1:115200baud - Tera Term VT

```

# cat /proc/dump/l2
----- LUT TABLE (48)-----
LUT idx=48
[P1] mac=00-e0-4c-86-70-02 cvid=1 l3lookup=0 ivl=0
efid=0 fid=2 sapri_en=0 spa=6 age=5 auth1x=0 sablock=0
dablock=0 ext_spa=0 arp_used=0 lutpri_en=0 lutpri=0 fwdpri_en=0 notsalearn=0
----- LUT TABLE (60)-----
LUT idx=60
[P1] mac=00-e0-4c-86-70-01 cvid=1 l3lookup=0 ivl=0
efid=0 fid=2 sapri_en=0 spa=6 age=5 auth1x=0 sablock=0
dablock=0 ext_spa=0 arp_used=0 lutpri_en=0 lutpri=0 fwdpri_en=0 notsalearn=0
----- LUT TABLE (140)-----
LUT idx=140
[P1] mac=00-1a-4b-5f-dc-1a cvid=9 l3lookup=0 ivl=1
efid=0 fid=0 sapri_en=0 spa=0 age=1 auth1x=0 sablock=0
dablock=0 ext_spa=0 arp_used=1 lutpri_en=0 lutpri=0 fwdpri_en=0 notsalearn=1
----- LUT TABLE (172)-----
LUT idx=172
[P1] mac=00-1a-4b-5f-dc-1a cvid=8 l3lookup=0 ivl=1
efid=0 fid=1 sapri_en=0 spa=5 age=7 auth1x=0 sablock=0

```

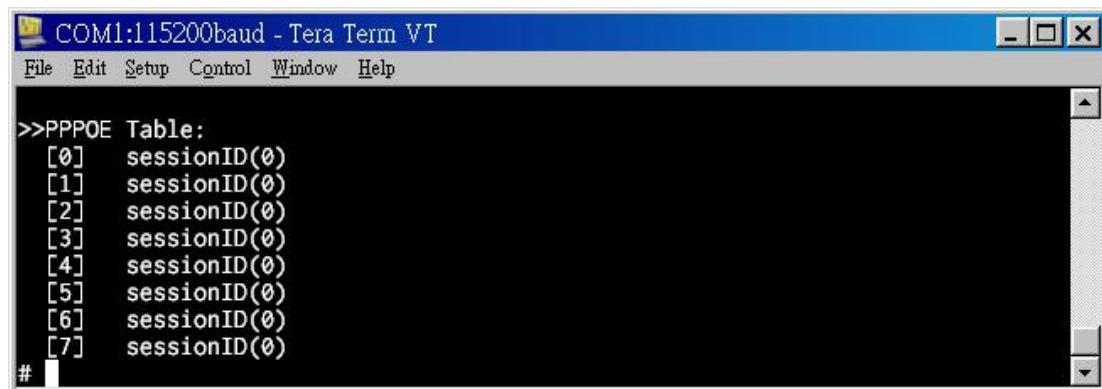
cat /proc/dump/napt <Show NAPT Table >

COM1:115200baud - Tera Term VT

```

# cat /proc/dump/napt
>>ASIC NAPT TCP/UDP Table:
----- Outbound -----
[ 704] INIDX(112) priValid(0) priority(0) - extPort(0x1018) state(CONNECTED) idle(850)
[1104] INIDX(104) priValid(0) priority(0) - extPort(0x1016) state(CONNECTED) idle(850)
[1288] INIDX(48) priValid(0) priority(0) - extPort(0x1028) state(CONNECTED) idle(830)
[1396] INIDX(80) priValid(0) priority(0) - extPort(0x1010) state(CONNECTED) idle(80)
[1600] INIDX(100) priValid(0) priority(0) - extPort(0x1015) state(CONNECTED) idle(850)
[1988] INIDX(356) priValid(0) priority(0) - extPort(0x1055) state(CONNECTED) idle(220)
----- Inbound -----
[ 48] 192.168.1.34:4136 V(2), IPIDX(1) REMHASH(64316) EPLSB(0x28) TCP(1) PRI_EN(0) PRI(0) - remote(174.36.85.72:0x50) idle(830)
[ 80] 192.168.1.34:4112 V(2), IPIDX(1) REMHASH(22368) EPLSB(0x10) TCP(1) PRI_EN(0) PRI(0) - remote(192.168.150.117:0x1bd) idle(80)
[ 100] 192.168.1.34:4117 V(2), IPIDX(1) REMHASH(21875) EPLSB(0x15) TCP(1) PRI_EN(0) PRI(0) - remote(74.125.31.94:0x50) idle(850)
[ 104] 192.168.1.34:4118 V(2), IPIDX(1) REMHASH(21828) EPLSB(0x16) TCP(1) PRI_EN(0) PRI(0) - remote(74.125.31.105:0x50) idle(850)
[ 112] 192.168.1.34:4120 V(2), IPIDX(1) REMHASH(58828) EPLSB(0x18) TCP(1) PRI_EN(0) PRI(0) - remote(173.194.72.94:0x50) idle(850)
[ 356] 192.168.1.34:4181 V(2), IPIDX(1) REMHASH(45140) EPLSB(0x55) TCP(1) PRI_EN(0) PRI(0) - remote(172.21.3.15:0x1f4e) idle(220)
Total entry: 6
# 
```

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

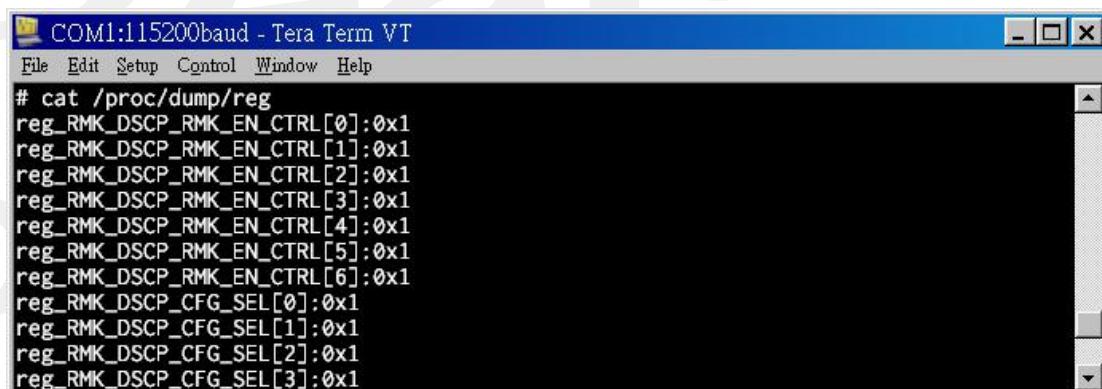


COM1:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
>>PPPOE Table:  
[0] sessionID(0)  
[1] sessionID(0)  
[2] sessionID(0)  
[3] sessionID(0)  
[4] sessionID(0)  
[5] sessionID(0)  
[6] sessionID(0)  
[7] sessionID(0)
```

```
# cat /proc/dump/reg <Show Register >
```



COM1:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
# cat /proc/dump/reg  
reg_RMK_DSCP_RMK_EN_CTRL[0]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[1]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[2]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[3]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[4]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[5]:0x1  
reg_RMK_DSCP_RMK_EN_CTRL[6]:0x1  
reg_RMK_DSCP_CFG_SEL[0]:0x1  
reg_RMK_DSCP_CFG_SEL[1]:0x1  
reg_RMK_DSCP_CFG_SEL[2]:0x1  
reg_RMK_DSCP_CFG_SEL[3]:0x1
```

```
# cat /proc/dump/hs <Show HSB/HSA/HSD>
```

```

COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help
# cat /proc/dump/hs
---- [HSB:]-----
spa: 0 pktLen: 96 ponIdx: 0
da: FF:FF:FF:FF:FF:FF sa: 00:1A:4B:5F:DC:1A etherType: 0x0800
ctag: 0 pri: 0 cfi: 0 vid: 0
stag: 0 pri: 0 cfi: 0 vid: 0
dip: 192.168.1.255 sip: 192.168.1.34 iptype: 1 tos_dscp: 0x00
14ok 13ok gt1 gt5 gre icmp udp tcp
1 1 1 0 0 0 1 0
ptp oam rlpp rldp llc snap pppoe session
0 0 0 0 0 0 0x0000
userfield valid: 0xfffff
00-07: 0x0089 0x0089 0x0000 0xf09c 0x4500 0x544e 0x0000 0x0000
08-15: 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0011 0x0002
-----
---- [HSA:]-----
Port      CPU  5   4   PON  2   1   0
user_pri: 0   0   0   0   0   0   0
qid:     0   0   0   0   0   0   0
dmp:     3   3   0   0   0   0   0
untagset: 1   1   1   0   0   1   1
spa ctag_act tag_if vid cfi pri vidzero
0   1   0   9   0   0   0
stag_type stag_if sp2s svid svidx dei spri pkt_spri vidsel frctag frctag_if
0   0   0   0   0   0   0   1   0   0
1p_rem 1p_rem_en dscp_rem dscp_rem_en keep ptp ipv4 ipv6 1042 pppoe
0   0   0   0   0   0   0   1   0   0   0
endsc bgdsc cpupri fwdrsn pon_sid pktlen regen_crc
0x03a4 0x0342 0   0   0   96   0
vc_spa: 0 vc_mask: 0x0000 ext_mask: 0x0001
13: 0 org: 1 l2trans: 0 l34trans: 0 src_mode: 1 l3chsum: 0x0000 l4schsum 0x0000
pppoe_idx: 0 pppoe_act: 0 ttl_extmask: 0x00 ttl_pmask: 0x00
newmac: 00:00:00:4B:5F:FF smac_idx: 0 newip: 192.168.1.34 newport: 0x0089
-----
---- [HSD:]-----
newmac: 00:00:00:4B:5F:FF l34mac: 00:E0:4C:86:70:01 newip: 192.168.1.34 newprt: 13
7
ep dsl_vc 34pppoe ttlpmask ttllexmsk l4cksum l3cksum pppoeact
6 0   0   0x00000 0x000000 0   0   0
src_mod l34trans l2trans org l3r sv_dei styp pktlen_ori qid
1   0   0   1   0   0   0   96   0
stdsc cpupri spri cori cmdy crms cins cvid cfi regencrc pppoe
834 0   0   1   0   0   0   9   0   0
rfc1042 ipv6 ipv4 ptp remdscp_pri rem1q_pri remdscp_en rem1q_en
0   0   1   0   0   0   1   0   0
svid instag incstag pktlen spa dpc extmsk vcmsk ponsid trprsn
0   0   0   104   0   2   0x0001 0x0000 0
-----
# 

```

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.

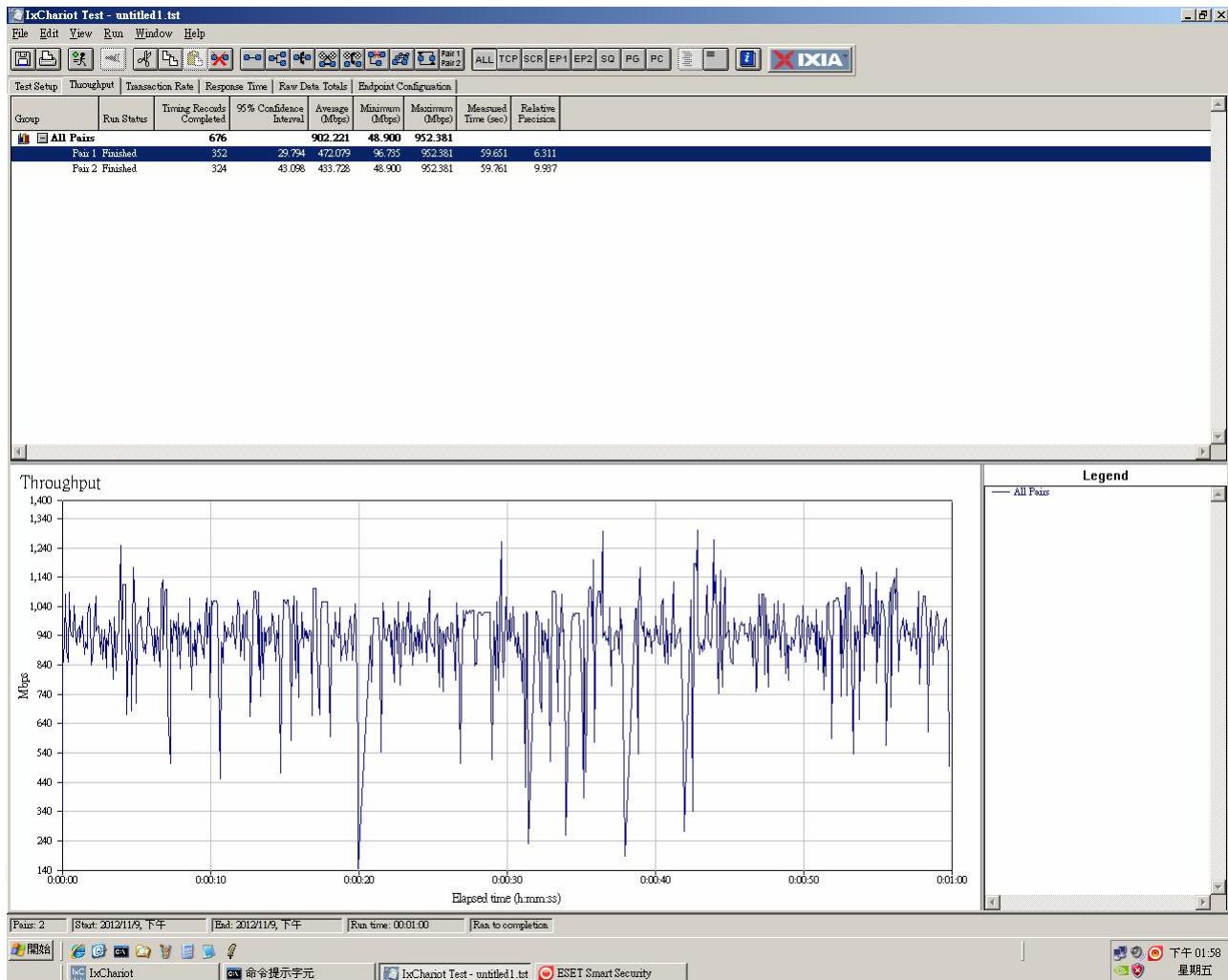


Figure17. Chariot LAN to LAN TCP connection

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.

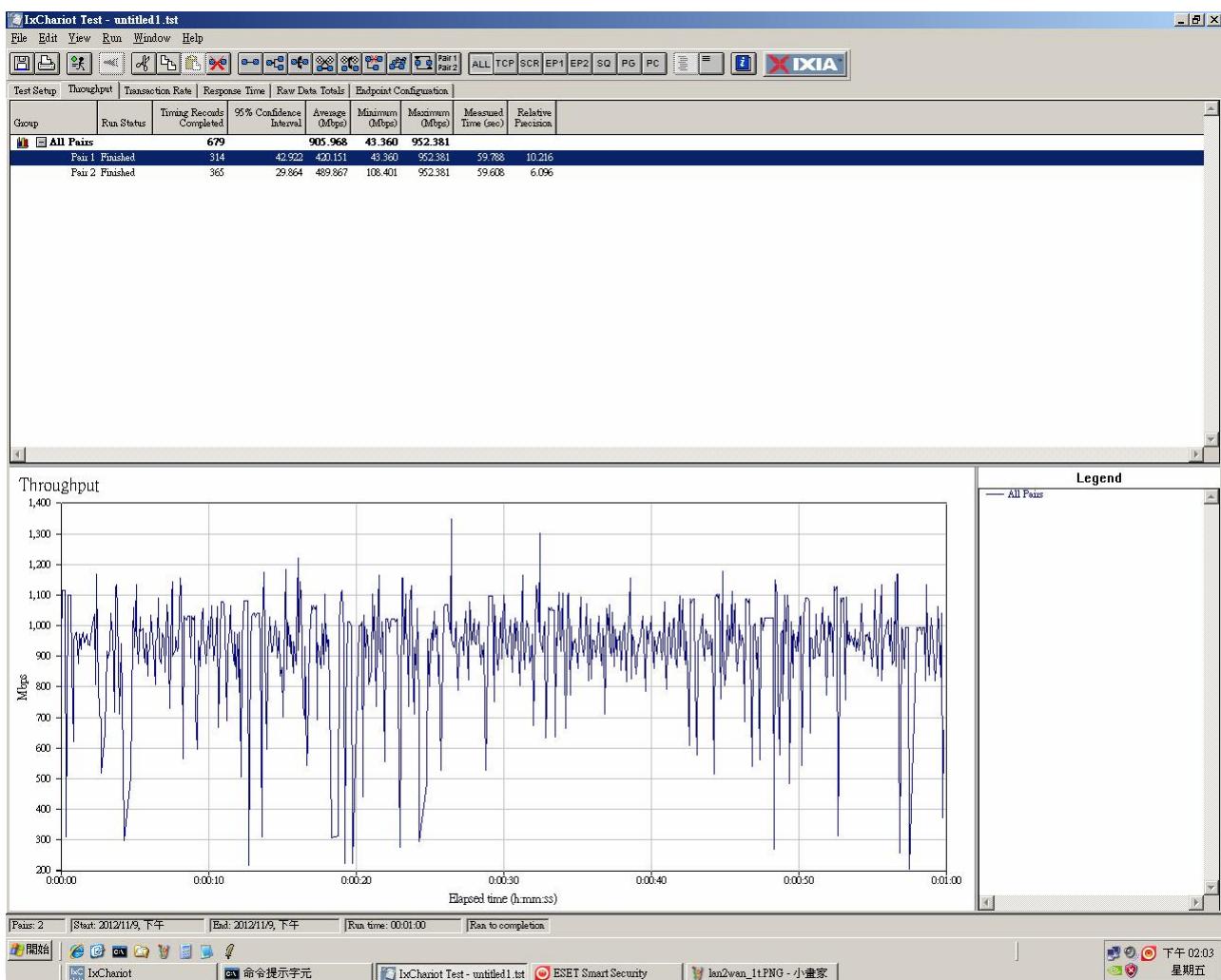


Figure18. Chariot LAN to WAN TCP connection

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

Data Bit Rate: **986.962Mbps**

Total Packets/Sec:**81486.31fps**

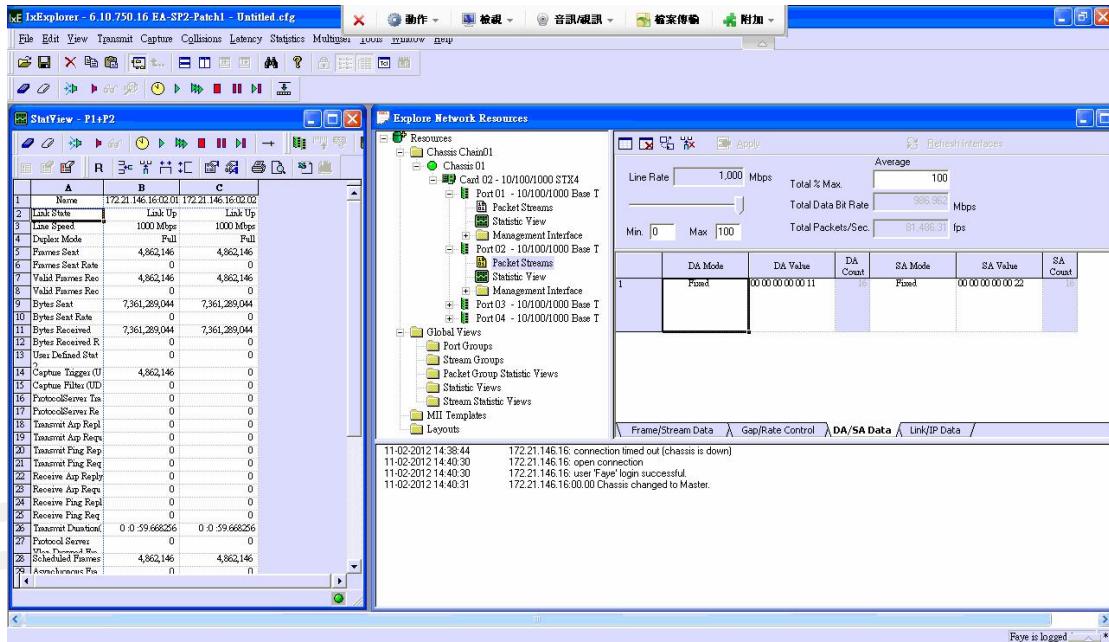


Figure19. IXIA LAN to LAN TCP connection(Large Packet)

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

Data Bit Rate: 761.905Mbps

Total Packets/Sec:1488095.2fps

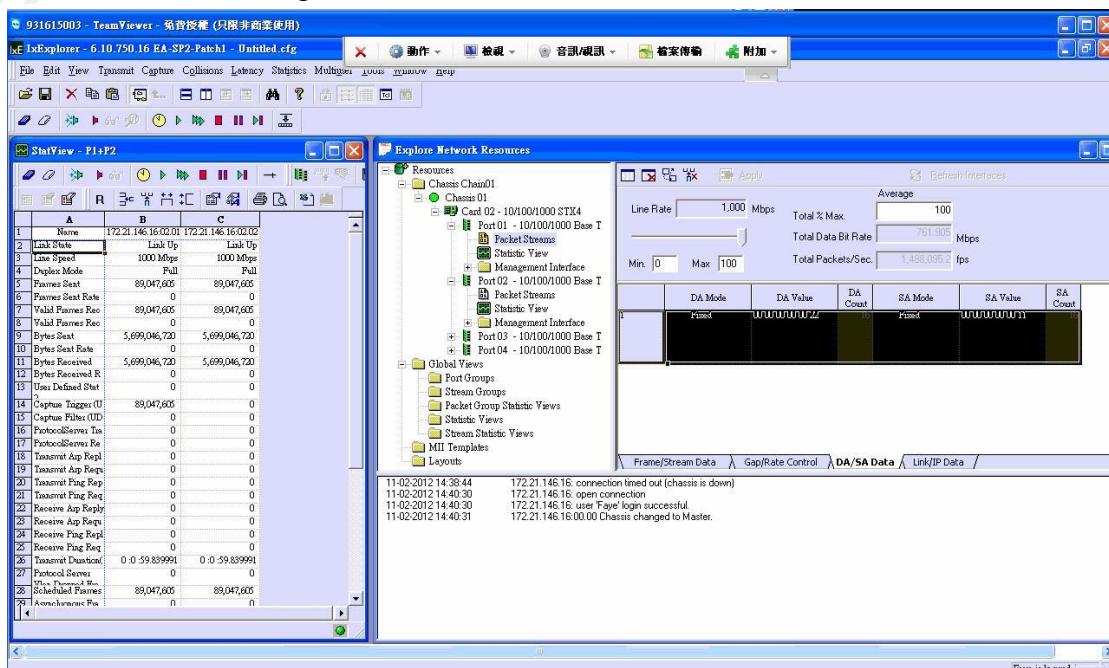


Figure20. IXIA LAN to LAN TCP connection(Small Packet)

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

Data Bit Rate: **986.962Mbps**

Total Packets/Sec:**81486.31fps**

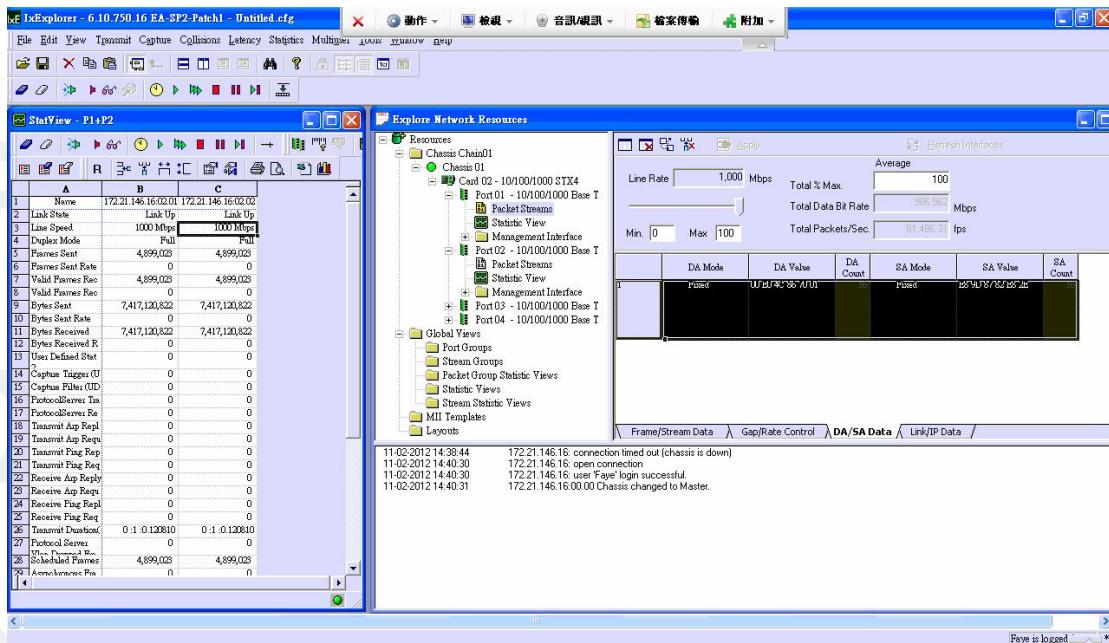


Figure21. IXIA LAN to WAN TCP connection(Large Packet)

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

Data Bit Rate: **761.905Mbps**

Total Packets/Sec:**1488095.2fps**

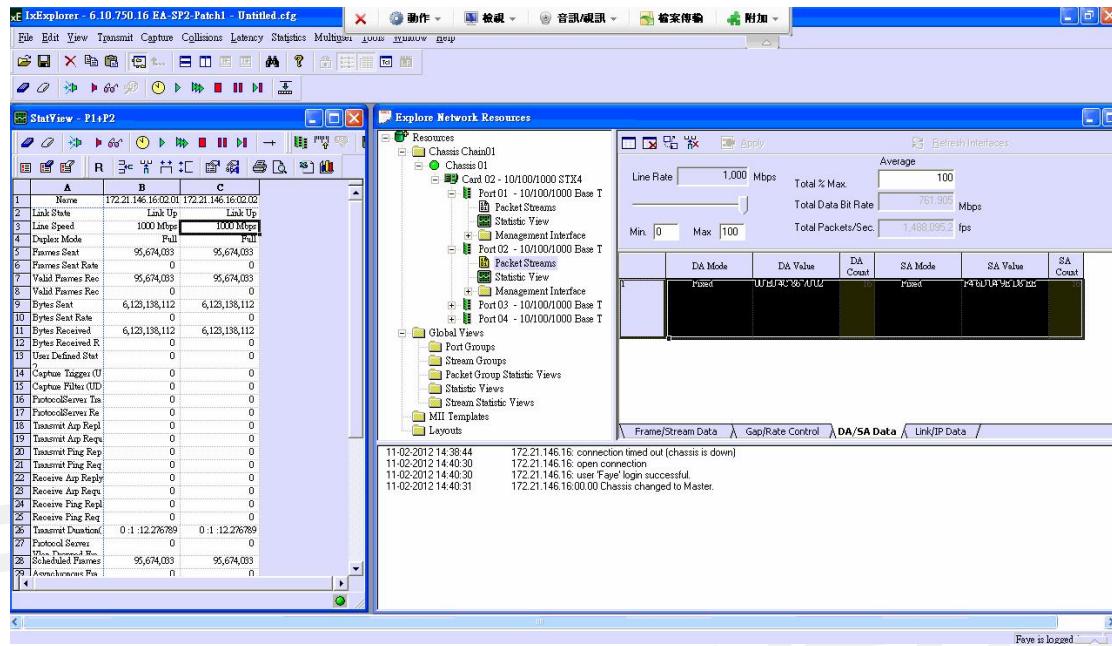


Figure22. IXIA LAN to WAN TCP connection(Small Packet)

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

Up-stream Data Bit Rate: 986.996Mbps

Total Packets/Sec: 81,274,382ps

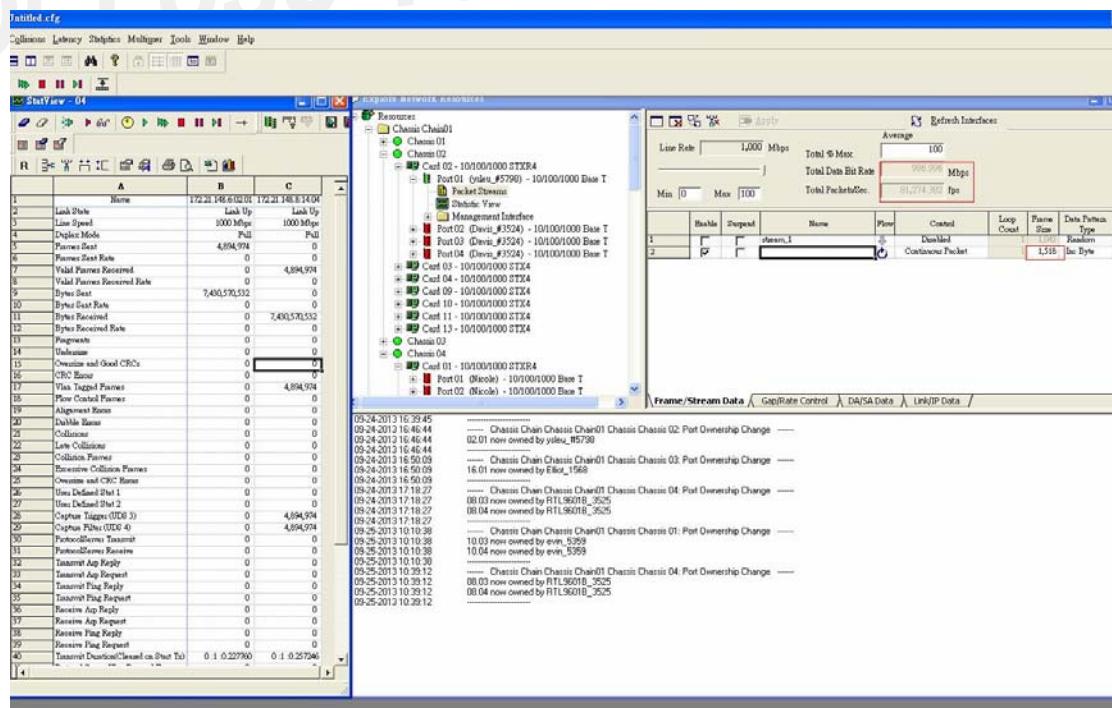


Figure23. IXIA LAN to GPON WAN UDP connection (Large Packet)

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

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

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

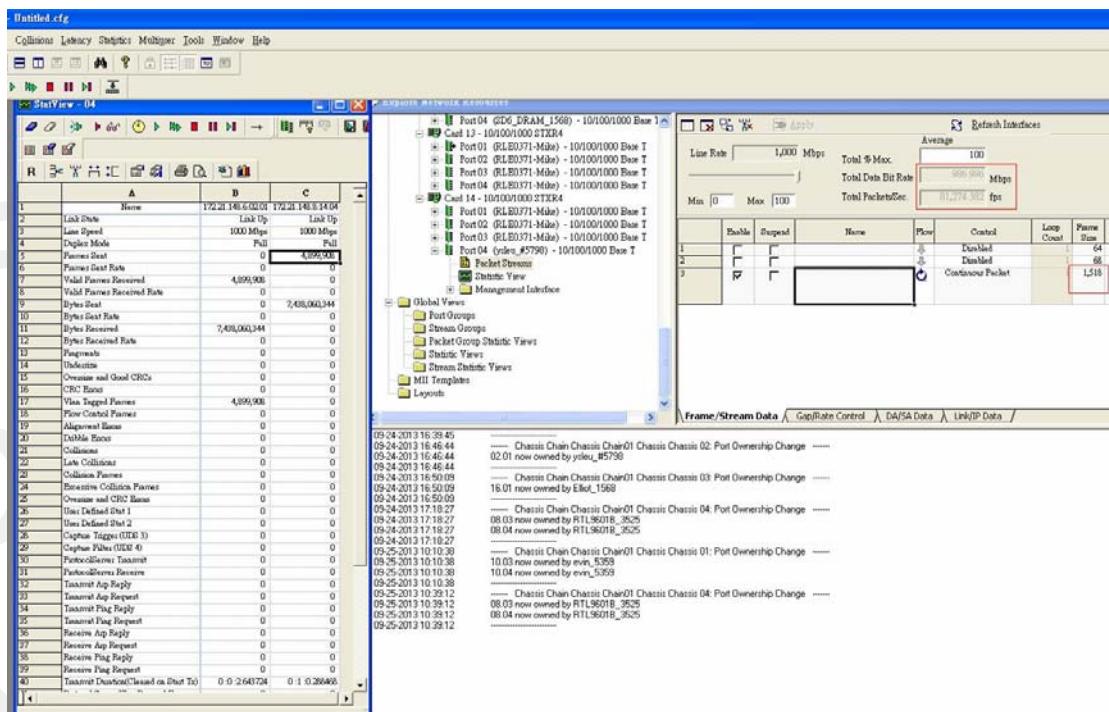


Figure 24. IXIA GPON WAN to LAN UDP connection (Large Packet)

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

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

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

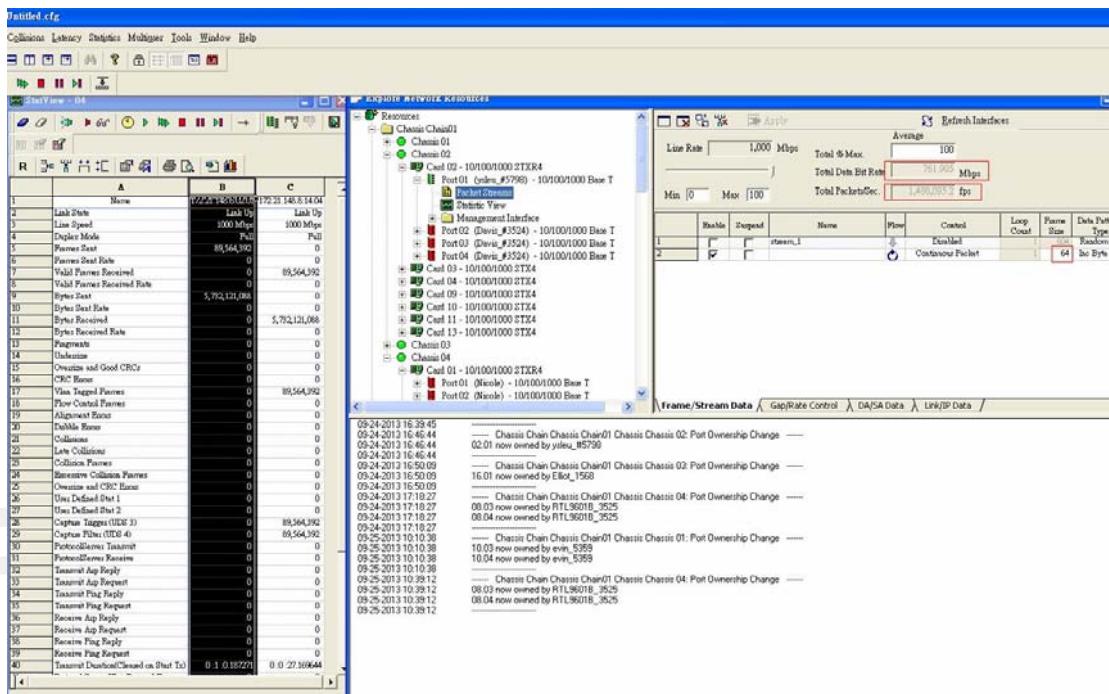


Figure25. IXIA LAN to GPON WAN UDP connection(Small Packet)

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

Down-stream Data Bit Rate: 761.905Mbps

Total Packets/Sec: 1,488,095.2ps

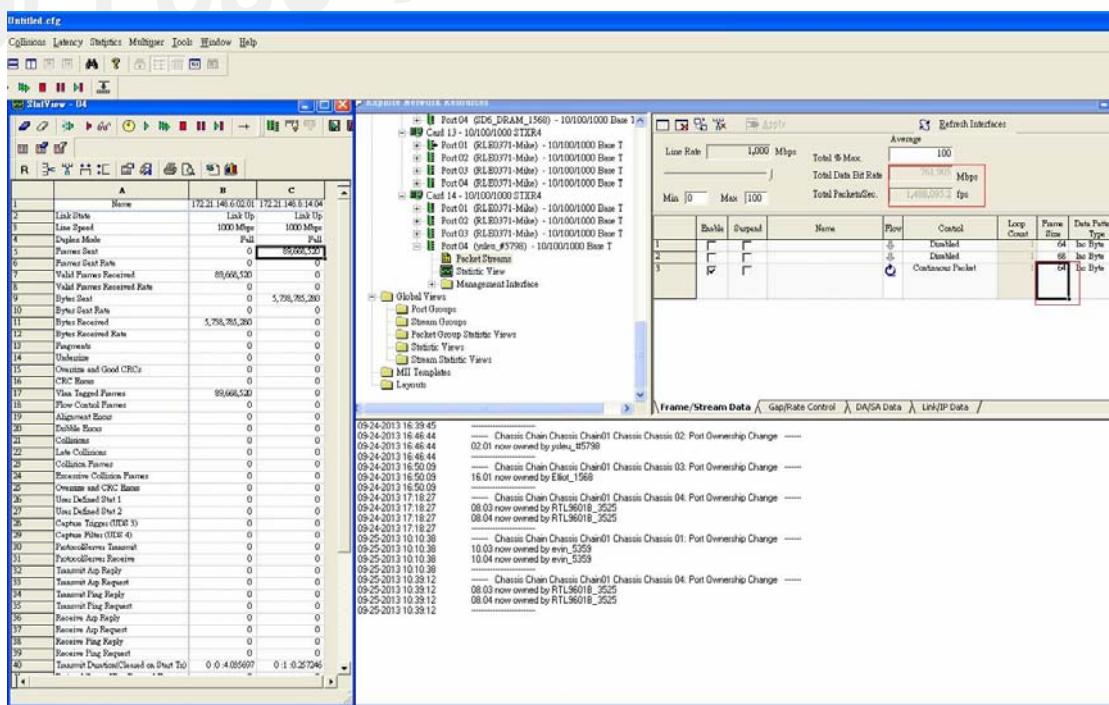


Figure26. IXIA GPON WAN to LAN UDP connection(Small Packet)

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 igmpSnoopingEnable - control IGMP snooping should on or off.

macBasedTagDecision - control tag decision based on MAC should on or off.
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

RT_ERR_RG_VLAN_SET_FAIL

The RG module is failed to initialize

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.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

the input parameter may be NULL

RT_ERR_RG_NULL_POINTER

the input parameter contains illegal information or range

RT_ERR_RG_INVALID_PARAM

the RG module didn't init

RT_ERR_RG_NOT_INIT

the interface number is beyond predefined limitation

RT_ERR_RG_ENTRY_FULL

the LAN interface can not add after WAN interface exist

RT_ERR_RG_MODIFY_LAN_AT_WA

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

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 | the input parameters may be NULL |
| RT_ERR_RG_NULL_POINTER | the input parameter contains illegal information or range |
| RT_ERR_RG_INVALID_PARAM | |
| 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 for chip A1, the PON port can not be set as WAN port |
| RT_ERR_RG_ENTRY_FULL | |
| 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

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

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 structure 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

RT_ERR_RG_INVALID_PARAM

the input parameters may be NULL

the input parameter contains illegal information or

| | |
|---------------------------|---|
| RT_ERR_RG_ENTRY_NOT_EXIST | range 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_demand - indicate that we connect only on demand.

app_info.idle_timeout_secs - indicate how long we should turn off connection after idle.

app_info.stats - indicate the status of connection.

app_info.dialOnDemandCallBack - 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

RT_ERR_RG_INVALID_PARAM

the input parameters may be NULL

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 | <i>wan_intf_idx</i> [in] The interface index of previous setup for PPPoE mode. <i>*clientPppoe_info</i> [in] The configuration related to PPPoE mode. |
| Comments | <i>clientPppoe_info.hw_info</i> - the static_info structure of this PPPoE mode WAN interface. <i>clientPppoe_info.sessionId</i> - 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 RT_ERR_RG_INVALID_PARAM RT_ERR_RG_ENTRY_NOT_EXIST RT_ERR_RG_ARP_FULL RT_ERR_RG_ARP_NOT_FOUND RT_ERR_RG_GW_MAC_NOT_SET RT_ERR_RG_ROUTE_GET_FAIL RT_ERR_RG_ENTRY_FULL RT_ERR_RG_ROUTE_SET_FAIL RT_ERR_RG_EXTIP_GET_FAIL RT_ERR_RG_EXTIP_SET_FAIL |
| | the input parameters may be NULL the input parameter contains illegal information or range the wan interface was not added before ARP table is not available for this new interface the remote gateway is not found or time gateway mac should be set for Lite RomeDriver for chip A1, the PON port can not be set as WAN port the function is not support for the chip unbind bridge WAN can't assign VLAN id different than LAN interface the global switch macBasedTagDecision didn't turn on |

12.11. rtk_rg_interface_del

| | |
|---------------------|---|
| Parameters | <code>int32 rtk_rg_interface_del(int lan_or_wan_intf_idx)</code> Delete the indicated interface, may be LAN or WAN interface. Defined in: <i>rtk_rg_liteRomeDriver.h</i> |
| Comments | [in] The index of interface. |
| Return Codes | RT_ERR_RG_OK RT_ERR_RG_INVALID_PARAM RT_ERR_RG_MODIFY_LAN_AT_WA_N_EXIST RT_ERR_RG_INTF_GET_FAIL RT_ERR_RG_ROUTE_GET_FAIL RT_ERR_RG_ROUTE_SET_FAIL RT_ERR_RG_NXP_GET_FAIL RT_ERR_RG_PPPOE_SET_FAIL RT_ERR_RG_NXP_SET_FAIL RT_ERR_RG_ROUTE_SET_FAIL RT_ERR_RG_EXTIP_GET_FAIL 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 |
| | the input parameter contains illegal information or range LAN interface should not be deleted when WAN interface existed the wan interface was not added before ARP table is not available for this new interface the remote gateway is not found or time gateway mac should be set for Lite RomeDriver for chip A1, the PON port can not be set as WAN port the function is not support for the chip unbind bridge WAN can't assign VLAN id different than LAN interface the global switch macBasedTagDecision didn't turn on |

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, either 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, either 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 increas 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

the input parameters may be NULL

RT_ERR_RG_NULL_POINTER

the input parameter contains illegal information or range

RT_ERR_RG_INVALID_PARAM

the wan interface was not added before

RT_ERR_RG_ENTRY_NOT_EXIST

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.

Return Codes

RT_ERR_RG_OK

the input parameters may be NULL

RT_ERR_RG_INVALID_PARAM

the input parameter contains illegal information or

RT_ERR_RG_NOT_INIT

| | |
|-------------------------------------|---|
| | range |
| RT_ERR_RG_VLAN_USED_BY_INTE_RFACE | the prefered VLAN ID had been used as interface VLAN ID |
| RT_ERR_RG_VLAN_USED_BY_VLA_NBINDING | 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

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 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

**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_INTE_RFACE

the VLAN id can't used by interface

RT_ERR_RG_VLAN_USED_BY_1QV_LAN

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

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.

Comments

**valid_idx*
[in] The index of the VLAN binding entry.
[out] The index of first valid VLAN binding entry.
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 | the input parameters may be NULL |
| RT_ERR_RG_NULL_POINTER | the input parameter contains illegal information or range |
| RT_ERR_RG_INVALID_PARAM | |
| 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 | the input parameters may be NULL |
| RT_ERR_RG_INVALID_PARAM | the server ip address had been assigned |
| RT_ERR_RG_ALG_SRV_IN_LAN_EX | IST |

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

Although 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 before turn on Server In Lan services, the server ip _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

Although 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.

| | |
|---------------------|---|
| Comments | <p><i>*dmz_info</i> [in] the DMZ setting. 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.</p> |
| 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.

| | |
|---------------------|--|
| Parameters | Defined in: rtk_rg_liteRomeDriver.h |
| | <i>wan_intf_idx</i> [in] the DMZ enabled wan interface index. |
| Comments | <p><i>*dmz_info</i> [in] the DMZ rule setting. 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.</p> |
| 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.

| | |
|---------------------|--|
| Parameters | Defined in: rtk_rg_liteRomeDriver.h |
| | <i>*virtual_server</i> [in] the svirtual server data structure. |
| Comments | <p><i>*virtual_server_idx</i> [in] the index of virtual server table. 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.</p> |
| 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 can't find entry in virtual server table.

_NOT_FOUND

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_etherype - assign the packet ingress ethertype pattern(valid when INGRESS_EHTERTYPE_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 assigned 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 asic PORT_RANGE_TABLE 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 asic DSCP_REMARKING_TABLE 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_FULL | the software ACL entry is full |
| RT_ERR_RG_ACL_SW_ENTRY_ACCESS_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_ACSES FAILED | access the hardware asic ACL IP_RANGE_TABLE entry failed |
| RT_ERR_RG_ACL_PORTTABLE_ACCESS FAILED | access the hardware ACL asic PORT_RANGE_TABLE entry failed |
| RT_ERR_RG_CF_ENTRY_ACCESS_FAILED | 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_ACCESS FAILED | access the hardware Classification asic PORT_RANGE_TABLE entry failed |
| RT_ERR_RG_CF_DSCPTABLE_ACCESS FAILED | access the hardware Classification asic DSCP_REMARKING_TABLE entry failed |
| RT_ERR_RG_ACL_SW_ENTRY_ACCESS 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 | the rg has not been init. (rtk_rg_initParam_set() should be called first) |
| RT_ERR_RG_NOT_INIT | the input parameters may be NULL |
| RT_ERR_RG_NULL_POINTER | the input parameter contains illegal information or range |
| RT_ERR_RG_INVALID_PARAM | access the software ACL entry failed |
| RT_ERR_RG_ACL_SW_ENTRY_ACCESS FAILED | can not find the assigned ACL entry |
| RT_ERR_RG_ACL_SW_ENTRY_NOT FOUND | |

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

[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.

None

Comments

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.

None

Comments

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.

None

Comments

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 UPNP 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_NO can't find entry in UPNP mapping table.

T_FOUND

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_DUPPLIC

duplicate NAPT flow.

| | |
|--------------------------|------------------------------|
| ATE | |
| 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_NO the NAPT entry can not be found.

T_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_NO the NAPT entry can not be found.

T_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, must be class D address.

mcFlow.src_intf_idx - This packet is from which interface index.

mcFlow.port_mask - This MC packet will forward to those 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 accepted.

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.

None

Success

Comments

RT_ERR_RG_OK

system is not initiated.

RT_ERR_RG_INDEX_OUT_OF_RANGE

the index out of range.

GE

the index is not exist.

RT_ERR_RG_ENTRY_NOT_EXIST

failed

RT_ERR_FAILED

Empty LUT entry.

RT_ERR_L2_EMPTY_ENTRY

Invalid input parameters.

RT_ERR_INPUT

invalid L2 Hash key

RT_ERR_L2_HASH_KEY

the entry is empty(invalid)

RT_ERR_L2_EMPTY_ENTRY

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

| | | |
|---------------------|-------------------------------|--|
| Comments | [in] | An empty buffer for storing the structure. |
| Return Codes | [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. |
| | GE | |
| | RT_ERR_RG_NO_MORE_ENTRY_FOUND | no more exist entry is found. |
| | UND | |
| | RT_ERR_FAILED | failed |
| | RT_ERR_L2_EMPTY_ENTRY | Empty LUT entry. |
| | RT_ERR_INPUT | Invalid input parameters. |

12.45. rtk_rg_multicastDataIngressAllowedPortMask_set

`int32 rtk_rg_multicastDataIngressAllowedPortMask_set(rtk_rg_mac_portmask_t *macPort_mask)`

Set allowed port mask for multicast data packets.

Defined in: rtk_rg_liteRomeDriver.h

Parameters `*macPort_mask`

[in] set allowed port mask for multicast data packets.

Comments None

Return Codes RT_ERR_RG_OK Success
RT_ERR_RG_NOT_INIT system is not initiated.

12.46. rtk_rg_multicastDataIngressAllowedPortMask_get

`int32 rtk_rg_multicastDataIngressAllowedPortMask_get(rtk_rg_mac_portmask_t *macPort_mask)`

Get allowed port mask for multicast data packets.

Defined in: rtk_rg_liteRomeDriver.h

Parameters `*macPort_mask`

[out] Return allowed port mask for multicast data packets.

Comments None

Return Codes RT_ERR_RG_OK Success
RT_ERR_RG_NOT_INIT system is not initiated.

12.47. rtk_rg_macEntry_add

`int32 rtk_rg_macEntry_add(rtk_rg_macEntry_t *macEntry, int *entry_idx)`
Add a MAC Entry into ASIC

| | | |
|---------------------|---|-------------------------------------|
| Parameters | Defined in: rtk_rg_liteRomeDriver.h *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 RT_ERR_RG_NOT_INIT | Success system is not initiated. |

12.48. rtk_rg_macEntry_del

int32 rtk_rg_macEntry_del(int entry_idx)
Delete an ASIC MAC Entry.

| | |
|-------------------|--|
| Parameters | Defined in: rtk_rg_liteRomeDriver.h |
| entry_idx | [in] The MAC entry index for deleting. |

Comments None

| | | |
|---------------------|---|---|
| Return Codes | RT_ERR_RG_OK RT_ERR_RG_NOT_INIT RT_ERR_RG_INDEX_OUT_OF_RANGE GE RT_ERR_RG_ENTRY_NOT_EXIST | Success system is not initiated. the index out of range. the index is not exist. |
|---------------------|---|---|

12.49. rtk_rg_macEntry_find

int32 rtk_rg_macEntry_find(rtk_rg_macEntry_t *macEntry, int *valid_idx)
Find an exist ASIC MAC Entry.

| | |
|-------------------|--|
| Parameters | Defined in: rtk_rg_liteRomeDriver.h |
| *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 RT_ERR_RG_NOT_INIT |

| | |
|-----------------------------------|----------------------------------|
| RT_ERR_RG_INDEX_OUT_OF_RAN GE | the index out of range. |
| RT_ERR_RG_NO_MORE_ENTRY_FO UND | no more exist entry is found. |
| RT_ERR_RG_NULL_POINTER | Input parameter is NULL pointer. |

12.50. 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

*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_FOU L2 entry not found.

ND

12.51. rtk_rg_arpEntry_del

int32 rtk_rg_arpEntry_del(int arp_entry_idx)

Delete an ASIC ARP Entry.

Defined in: rtk_rg_liteRomeDriver.h

arp_entry_idx

[out] The ARP entry index for deleting.

None

Comments

RT_ERR_RG_OK Success

RT_ERR_RG_NOT_INIT system is not initiated.

RT_ERR_RG_INDEX_OUT_OF_RAN the index out of range.

GE

RT_ERR_RG_ENTRY_NOT_EXIST the index is not exist.

12.52. 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

*arpInfo

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

[out] The data structure of found ARP entry.

| | | |
|---------------------|---|-------------------------------|
| | <i>*arp_valid_idx</i> | |
| | [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 |
| | GE | |
| | RT_ERR_RG_NO_MORE_ENTRY_FOUND | no more exist entry is found. |
| | UND | |

12.53. rtk_rg_neighborEntry_add

```
int32 rtk_rg_neighborEntry_add(rtk_rg_neighborEntry_t *neighborEntry, int
*ineighbor_idx)
```

Add an Neighbor Entry into ASIC

Defined in: rtk_rg_liteRomeDriver.h

**neighborEntry*

[in] Fill rtk_rg_neighborEntry_t each fields for adding.

**neighbor_idx*

[out] Return Neighbor entry index.

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.

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.

ND

RT_ERR_RG_NEIGHBOR_FULL

neighbor table is full

12.54. rtk_rg_neighborEntry_del

```
int32 rtk_rg_neighborEntry_del(int neighbor_idx)
```

Delete an ASIC Neighbor Entry.

Defined in: rtk_rg_liteRomeDriver.h

neighbor_idx

[out] The Neighbor entry index for deleting.

None

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.

GE

RT_ERR_RG_ENTRY_NOT_EXIST

the index is not exist.

12.55. 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.

**neighbor_valid_idx* [out] The data structure of found Neighbor entry.int

[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. |
|------------------------------|---|

| | |
|----|--|
| ND | |
|----|--|

12.56. 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. |
|-------------------------|--------------------------|

12.57. 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.

None

Comments

RT_ERR_RG_OK

Success

Return Codes

RT_ERR_RG_INVALID_PARAM

system is not initiated.

12.58. 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

12.59. 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.

12.60. 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 control bits are zeros.
 dos_type.RTK_RG_SYN_SPORL1024_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_OFFSET_MIN_CHECK - the Fragment_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. |

12.61. 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

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 control bits are zeros.

dos_type.RTK_RG_SYN_SPORL1024_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_OFFSET_MIN_CHECK - the Fragment_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 RT_ERR_RG_NULL_POINTER | Success input port mask pointer is NULL. |
| | | |

12.62. 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 | <i>dos_type</i> [in] Port security type. <i>dos_enabled</i> [in] Port security function enabled/disabled. <i>dos_action</i> [in] Port security action. <i>dos_threshold</i> [in] System |
|-------------------|--|

| | |
|-----------------|---|
| Comments | <i>dos_type.RTK_RG_SYNFFLOOD_DENY</i> - Receiving TCP SYN packet number is over threshold. <i>dos_type.RTK_RG_FINFFLOOD_DENY</i> - Receiving TCP FIN packet number is over threshold. <i>dos_type.RTK_RG_ICMPFFLOOD_DENY</i> - Receiving ICMP packet number is over threshold. <i>dos_action.RTK_RG_DOS_ACTION_DROP</i> - Drop packet while hit DoS criteria. <i>dos_action.RTK_RG_DOS_ACTION_TRAP</i> - Trap packet to CPU while hit DoS criteria. <i>dos_threshold</i> - Allowable SYN/FIN/ICMP packet frame rate. The forwarding rate is <i>dos_threshold*1k</i> packets per second. |
|-----------------|---|

| | | |
|---------------------|---|--|
| Return Codes | RT_ERR_RG_OK RT_ERR_RG_INVALID_PARAM | Success the input parameter contains illegal information. |
|---------------------|---|--|

12.63. 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 | <i>dos_type</i> [in] Port security type. <i>*dos_enabled</i> [out] Port security function enabled/disabled. <i>*dos_action</i> [out] Port security action. <i>*dos_threshold</i> [out] System |
|-------------------|--|

| | |
|-----------------|---|
| Comments | <i>dos_type.RTK_RG_SYNFFLOOD_DENY</i> - Receiving TCP SYN packet number is over threshold. <i>dos_type.RTK_RG_FINFFLOOD_DENY</i> - Receiving TCP FIN packet number is over threshold. <i>dos_type.RTK_RG_ICMPFFLOOD_DENY</i> - Receiving ICMP packet number is over threshold. <i>dos_action.RTK_RG_DOS_ACTION_DROP</i> - Drop packet while hit DoS criteria. <i>dos_action.RTK_RG_DOS_ACTION_TRAP</i> - Trap packet to CPU while hit DoS criteria. <i>dos_threshold</i> - Allowable SYN/FIN/ICMP packet frame rate. The forwarding rate is <i>dos_threshold*1k</i> 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. |

12.64. 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.

12.65. 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.

12.66. 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

12.67. 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

| | | |
|---------------------|--------------|---|
| | <i>rate</i> | |
| Comments | [in] | assigned rate, unit Kbps |
| | (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 |

12.68. 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 | <i>port</i> | |
| | [in] | assigned rate limit port |
| | <i>rate</i> | |
| | [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 |

12.69. 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 | <i>port</i> | |
| | [in] | assigned rate limit port |
| | <i>*rate</i> | |
| | [out] | assigned rate, unit Kbps |
| Comments | None | |
| Return Codes | RT_ERR_RG_OK | Success |
| | RT_ERR_RG_NULL_POINTER | input rate pointer is NULL. |

12.70. 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 | <i>port</i> | |
| | [in] | assigned rate limit port |
| | <i>*rate</i> | |
| | [out] | assigned rate, unit Kbps |
| Comments | None | |
| Return Codes | RT_ERR_RG_OK | Success |

| | |
|------------------------|-----------------------------|
| RT_ERR_RG_NULL_POINTER | input rate pointer is NULL. |
|------------------------|-----------------------------|

12.71. 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

12.72. 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.

12.73. 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->ifOutBroadcastPkts - 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->dot3StatsAlignmentErrors - 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;A including 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;A including 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;A including 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 RT_ERR_RG_NULL_POINTER RT_ERR_RG_INVALID_PARAM | Success input ability pointer is NULL. the input parameter contains illegal information. |
|---------------------|---|--|

12.74. 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 | <i>port</i> [in] assigned physical port |
|-------------------|--|

Comments None

| | | |
|---------------------|---|--|
| Return Codes | RT_ERR_RG_OK RT_ERR_RG_INVALID_PARAM | Success the input parameter contains illegal information. |
|---------------------|---|--|

12.75. 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
**stormInfo*

[in] enable stormCtrol
**stormInfo_idx*

[in] assigned port

Comments None

| | | |
|---------------------|---|--|
| Return Codes | RT_ERR_RG_OK RT_ERR_RG_NULL_POINTER RT_ERR_RG_INVALID_PARAM RT_ERR_RG_STORMCONTROL_EN TRY_HAS_BEEN_SET RT_ERR_RG_STORMCONTROL_TYP_E_FULL | Success parameter is null. parameters value is out of range. same stormInfo has been set before . TRY_HAS_BEEN_SET add stormType full(ASIC support at most 4 different types) |
|---------------------|---|--|

12.76. rtk_rg_stormControl_del

Parameters

Comments

Return Codes

```
int32 rtk_rg_stormControl_del(int stormInfo_idx)
Delete stormControl.
Defined in: rtk_rg_liteRomeDriver.h
stormInfo_idx [out] The index of stormInfo which should be delete.
None
RT_ERR_RG_OK Success
RT_ERR_RG_INVALID_PARAM parameters value is out of range.
```

12.77. rtk_rg_stormControl_find

Parameters

Comments

Return Codes

```
int32 rtk_rg_stormControl_find(rtk_rg_stormControlInfo_t *stormInfo, int
*stormInfo_idx)
Find stormInfo index.
Defined in: rtk_rg_liteRomeDriver.h
*stormInfo [in] assigned compare port
*stormInfo_idx [in] assigned compare stormType
None
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_EN assigned stormInfo not found
TRY_NOT_FOUND
```

12.78. rtk_rg_shareMeter_set

Parameters

Comments

Return Codes

```
int32 rtk_rg_shareMeter_set(uint32 index, uint32 rate, rtk_rg_enable_t ifgInclude)
Set shareMeter rate.
Defined in: rtk_rg_liteRomeDriver.h
index [in] shared meter index
rate [in] rate of share meter
ifgInclude [in] include IFG or not, ENABLE:include DISABLE:exclude
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.
RT_ERR_RG_OK Success
```

12.79. 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

12.80. 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.

12.81. 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] * <i>pQ_weight</i> | The port index. |
| | [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. |

12.82. 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

12.83. 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. |

12.84. 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

weightOfPriSel

[in] weight of each priority assignment

Comments None

Return Codes RT_ERR_RG_OK

Success

RT_ERR_RG_INVALID_PARAM parameter is null.

12.85. 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

**pWeightOfPriSel*

[out] weight of each priority assignment

Comments None

Return Codes RT_ERR_RG_OK

Success

RT_ERR_RG_NULL_POINTER parameter is null.

12.86. 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

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. |

12.87. 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 | <i>dscp</i> | |
| | [in] | DSCP |
| | * <i>pInt_pri</i> | |
| | [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. |

12.88. 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 | <i>port_idx</i> | |
| | [in] | Port index, including extension port. |
| | * <i>int_pri</i> | |
| | [in] | Priorities assignment for specific port. |
| Comments | None | |
| Return Codes | RT_ERR_RG_OK | Success |
| | RT_ERR_RG_INVALID_PARAM | parameter is null. |

12.89. 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 | <i>port_idx</i> | |
| | [in] | Port index, including extension port. |
| | * <i>pInt_pri</i> | |
| | [out] | Priorities assignment 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. |

12.90. 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.

12.91. 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.

12.92. 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)

| | |
|---------------------|---|
| Parameters | <i>rmk_port</i> [in] Enable/Dsiable DSCP remarking port <i>rmk_enable</i> [in] Enable/Disable DSCP remark <i>rmk_src_select</i> [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. |

12.93. rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get

int32 rtk_rg_qosDscpRemarkEgressPortEnableAndSrcSelect_get(rtк_rg_mac_port_idx_t rmk_port, rtк_rg_enable_t *pRmk_enable, rtк_rg_qos_dscpRmkSrc_t *pRmk_src_select)
Get remark DSCP enable/disable port and remarking source.

Defined in: rtk_rg_liteRomeDriver.h

| | |
|---------------------|---|
| Parameters | <i>rmk_port</i> [in] DSCP remarking port <i>*pRmk_enable</i> [in] Pointer to DSCP remarking port enabled/disabled <i>*pRmk_src_select</i> [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 |

12.94. 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 | <i>int_pri</i> [in] Internal/User priority <i>rmk_dscp</i> [in] DSCP remarking value |
|-------------------|---|

| | | | | | |
|-------------------------|---|--------------|---------|-------------------------|--------------------|
| Comments | rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_DSCP - Using DSCP as DSCP remarking source. | | | | |
| Return Codes | <table border="0"> <tr> <td>RT_ERR_RG_OK</td> <td>Success</td> </tr> <tr> <td>RT_ERR_RG_INVALID_PARAM</td> <td>parameter is null.</td> </tr> </table> | RT_ERR_RG_OK | Success | RT_ERR_RG_INVALID_PARAM | parameter is null. |
| RT_ERR_RG_OK | Success | | | | |
| RT_ERR_RG_INVALID_PARAM | parameter is null. | | | | |

12.95. 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 | <i>int_pri</i> [in] Internal/User priority |
| | <i>*pRmk_dscp</i> [out] Pointer to remarking DSCP |

| | |
|-----------------|---|
| Comments | rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_DSCP - Using DSCP as DSCP remarking source. |
|-----------------|---|

| | | | | | | | |
|-------------------------|---|--------------|---------|-------------------------|--------------------|------------------------|-----------------------------------|
| Return Codes | <table border="0"> <tr> <td>RT_ERR_RG_OK</td><td>Success</td></tr> <tr> <td>RT_ERR_RG_INVALID_PARAM</td><td>parameter is null.</td></tr> <tr> <td>RT_ERR_RG_NULL_POINTER</td><td>parameters value is out of range.</td></tr> </table> | 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_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_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 | <i>dscp</i> [in] DSCP source |
| | <i>rmk_dscp</i> [in] DSCP remarking value |

| | |
|-----------------|---|
| Comments | rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_INT_PRI - Using internal priority as DSCP remarking source. rtk_rg_qos_dscpRmkSrc_t.RTK_RG_DSCP_RMKB_SRC_DSCP - Using DSCP as DSCP remarking source. |
|-----------------|---|

| | | | | | |
|-------------------------|---|--------------|---------|-------------------------|--------------------|
| Return Codes | <table border="0"> <tr> <td>RT_ERR_RG_OK</td><td>Success</td></tr> <tr> <td>RT_ERR_RG_INVALID_PARAM</td><td>parameter is null.</td></tr> </table> | RT_ERR_RG_OK | Success | RT_ERR_RG_INVALID_PARAM | parameter is null. |
| RT_ERR_RG_OK | Success | | | | |
| RT_ERR_RG_INVALID_PARAM | parameter is null. | | | | |

12.97. 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 | <i>dscp</i> [in] DSCP source * <i>pRmk_dscp</i> [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. |

12.98. 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 | <i>rmk_port</i> [in] Enable/Dsiable 802.1p remarking port. <i>rmk_enable</i> [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. |

12.99. rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get

| | |
|-------------------|--|
| | int32 rtk_rg_qosDot1pPriRemarkByInternalPriEgressPortEnable_get(rtk_rg_mac_port_idx_t rmk_port, rtk_rg_enable_t * <i>pRmk_enable</i>) |
| | Get remark 802.1p priority port. Defined in: rtk_rg_liteRomeDriver.h |
| Parameters | <i>rmk_port</i> [out] Pointer to 802.1p remarking port. <i>*pRmk_enable</i> [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 - |

| | | |
|---------------------|--|-----------------------------------|
| Return Codes | Using DSCP as DSCP remarking source. 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_qosDot1pPriRemarkByInternalPri_set

| | |
|---------------------|---|
| Parameters | <code>int32 rtk_rg_qosDot1pPriRemarkByInternalPri_set(int int_pri, int rmk_dot1p)</code> Set Internal priority to 802.1p remarking mapping. Defined in: rtk_rg_liteRomeDriver.h |
| | <i>int_pri</i> [in] Internal priority |
| | <i>rmk_dot1p</i> [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. |

12.101. rtk_rg_qosDot1pPriRemarkByInternalPri_get

| | |
|---------------------|---|
| Parameters | <code>int32 rtk_rg_qosDot1pPriRemarkByInternalPri_get(int int_pri, int *pRmk_dot1p)</code> Get Internal priority to 802.1p remarking mapping. Defined in: rtk_rg_liteRomeDriver.h |
| | <i>int_pri</i> [in] Internal priority |
| | <i>*pRmk_dot1p</i> [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. |

12.102. 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

parameters value is out of range.

12.103. 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

parameters value is out of range.

12.104. 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. |

12.105. 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