



# REALTEK

**NOT FOR PUBLIC RELEASE**

## Unmanaged Switch

# API Document

Ver 1.4.0

11<sup>th</sup> June, 2020



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](http://www.realtek.com)

## COPYRIGHT

© 2020 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 Switch 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
1.4.0	2020/06/11	First Release	Realtek

Realtek Confidential files  
The document authorized to  
windy(e-link)  
2021-04-09 10:57:16

1.	Module acl.h - Unmanaged switch high-level API.....	19
1.1.	rtk_filter_igrAcl_init .....	20
1.2.	rtk_filter_igrAcl_field_add .....	20
1.3.	rtk_filter_igrAcl_cfg_add.....	21
1.4.	rtk_filter_igrAcl_cfg_del.....	22
1.5.	rtk_filter_igrAcl_cfg_delAll .....	22
1.6.	rtk_filter_igrAcl_cfg_get.....	22
1.7.	rtk_filter_igrAcl_unmatchAction_set .....	23
1.8.	rtk_filter_igrAcl_unmatchAction_get .....	24
1.9.	rtk_filter_igrAcl_state_set.....	24
1.10.	rtk_filter_igrAcl_state_get .....	25
1.11.	rtk_filter_igrAcl_template_set .....	25
1.12.	rtk_filter_igrAcl_template_get.....	26
1.13.	rtk_filter_igrAcl_field_sel_set .....	26
1.14.	rtk_filter_igrAcl_field_sel_get .....	27
1.15.	rtk_filter_iprange_set .....	27
1.16.	rtk_filter_iprange_get .....	28
1.17.	rtk_filter_vidrange_set .....	29
1.18.	rtk_filter_vidrange_get .....	29
1.19.	rtk_filter_portrange_set .....	30
1.20.	rtk_filter_portrange_get.....	31
1.21.	rtk_filter_igrAclPolarity_set .....	31
1.22.	rtk_filter_igrAclPolarity_get .....	32
2.	Module cpu.h - Unmanaged switch high-level API.....	32
2.1.	rtk_cpu_enable_set .....	33
2.2.	rtk_cpu_enable_get .....	33
2.3.	rtk_cpu_tagPort_set .....	34
2.4.	rtk_cpu_tagPort_get .....	34

2.5.	rtk_cpu_awarePort_set .....	35
2.6.	rtk_cpu_awarePort_get.....	35
2.7.	rtk_cpu_tagPosition_set .....	36
2.8.	rtk_cpu_tagPosition_get .....	36
2.9.	rtk_cpu_tagLength_set .....	37
2.10.	rtk_cpu_tagLength_get.....	37
2.11.	rtk_cpu_acceptLength_set.....	38
2.12.	rtk_cpu_acceptLength_get .....	38
2.13.	rtk_cpu_priRemap_set.....	38
2.14.	rtk_cpu_priRemap_get .....	39
3.	Module dot1x.h - Unmanaged switch high-level API .....	40
3.1.	rtk_dot1x_unauthPacketOper_set .....	40
3.2.	rtk_dot1x_unauthPacketOper_get.....	41
3.3.	rtk_dot1x_eapolFrame2CpuEnable_set .....	42
3.4.	rtk_dot1x_eapolFrame2CpuEnable_get.....	42
3.5.	rtk_dot1x_portBasedEnable_set.....	43
3.6.	rtk_dot1x_portBasedEnable_get .....	43
3.7.	rtk_dot1x_portBasedAuthStatus_set .....	44
3.8.	rtk_dot1x_portBasedAuthStatus_get.....	45
3.9.	rtk_dot1x_portBasedDirection_set.....	45
3.10.	rtk_dot1x_portBasedDirection_get .....	46
3.11.	rtk_dot1x_macBasedEnable_set .....	46
3.12.	rtk_dot1x_macBasedEnable_get .....	47
3.13.	rtk_dot1x_macBasedAuthMac_add .....	48
3.14.	rtk_dot1x_macBasedAuthMac_del .....	48
3.15.	rtk_dot1x_macBasedDirection_set .....	49
3.16.	rtk_dot1x_macBasedDirection_get .....	49
3.17.	rtk_dot1x_guestVlan_set.....	50

3.18.	rtk_dot1x_guestVlan_get .....	50
3.19.	rtk_dot1x_guestVlan2Auth_set.....	51
3.20.	rtk_dot1x_guestVlan2Auth_get .....	51
4.	Module eee.h - Unmanaged switch high-level API.....	52
4.1.	rtk_eee_init .....	52
4.2.	rtk_eee_portEnable_set .....	53
4.3.	rtk_eee_portEnable_get.....	53
5.	Module igmp.h - Unmanaged switch high-level API.....	54
5.1.	rtk_igmp_init .....	55
5.2.	rtk_igmp_state_set.....	55
5.3.	rtk_igmp_state_get .....	56
5.4.	rtk_igmp_static_router_port_set .....	56
5.5.	rtk_igmp_static_router_port_get.....	57
5.6.	rtk_igmp_protocol_set .....	57
5.7.	rtk_igmp_protocol_get .....	58
5.8.	rtk_igmp_fastLeave_set .....	58
5.9.	rtk_igmp_fastLeave_get.....	59
5.10.	rtk_igmp_maxGroup_set.....	59
5.11.	rtk_igmp_maxGroup_get .....	60
5.12.	rtk_igmp_currentGroup_get.....	60
5.13.	rtk_igmp_tableFullAction_set .....	61
5.14.	rtk_igmp_tableFullAction_get .....	61
5.15.	rtk_igmp_checksumErrorAction_set .....	62
5.16.	rtk_igmp_checksumErrorAction_get.....	62
5.17.	rtk_igmp_leaveTimer_set.....	63
5.18.	rtk_igmp_leaveTimer_get .....	63
5.19.	rtk_igmp_queryInterval_set .....	63
5.20.	rtk_igmp_queryInterval_get.....	64

5.21.	rtk_igmp_robustness_set .....	64
5.22.	rtk_igmp_robustness_get.....	65
5.23.	rtk_igmp_dynamicRouterPortAllow_set .....	65
5.24.	rtk_igmp_dynamicRouterPortAllow_get.....	66
5.25.	rtk_igmp_dynamicRouterPort_get .....	66
5.26.	rtk_igmp_suppressionEnable_set.....	67
5.27.	rtk_igmp_suppressionEnable_get .....	67
5.28.	rtk_igmp_portRxPktEnable_set .....	68
5.29.	rtk_igmp_portRxPktEnable_get .....	68
5.30.	rtk_igmp_groupInfo_get.....	69
5.31.	rtk_igmp_ReportLeaveFwdAction_set .....	69
5.32.	rtk_igmp_ReportLeaveFwdAction_get.....	70
5.33.	rtk_igmp_dropLeaveZeroEnable_set .....	70
5.34.	rtk_igmp_dropLeaveZeroEnable_get.....	71
5.35.	rtk_igmp_bypassGroupRange_set.....	71
5.36.	rtk_igmp_bypassGroupRange_get .....	72
6.	Module interrupt.h - Unmanaged switch high-level API.....	72
6.1.	rtk_int_polarity_set.....	73
6.2.	rtk_int_polarity_get .....	73
6.3.	rtk_int_control_set.....	74
6.4.	rtk_int_control_get .....	74
6.5.	rtk_int_status_set .....	75
6.6.	rtk_int_status_get.....	76
6.7.	rtk_int_advanceInfo_get.....	77
7.	Module l2.h - Unmanaged switch high-level API .....	77
7.1.	rtk_l2_init .....	79
7.2.	rtk_l2_addr_add.....	79
7.3.	rtk_l2_addr_get.....	80

7.4.	rtk_l2_addr_next_get.....	80
7.5.	rtk_l2_addr_del.....	81
7.6.	rtk_l2_mcastAddr_add .....	82
7.7.	rtk_l2_mcastAddr_get .....	82
7.8.	rtk_l2_mcastAddr_next_get .....	83
7.9.	rtk_l2_mcastAddr_del .....	84
7.10.	rtk_l2_ipMcastAddr_add.....	84
7.11.	rtk_l2_ipMcastAddr_get.....	85
7.12.	rtk_l2_ipMcastAddr_next_get.....	85
7.13.	rtk_l2_ipMcastAddr_del.....	86
7.14.	rtk_l2_ipVidMcastAddr_add.....	86
7.15.	rtk_l2_ipVidMcastAddr_get .....	87
7.16.	rtk_l2_ipVidMcastAddr_next_get .....	87
7.17.	rtk_l2_ipVidMcastAddr_del .....	88
7.18.	rtk_l2_icastAddr_flush.....	88
7.19.	rtk_l2_table_clear.....	89
7.20.	rtk_l2_table_clearStatus_get .....	90
7.21.	rtk_l2_flushLinkDownPortAddrEnable_set .....	90
7.22.	rtk_l2_flushLinkDownPortAddrEnable_get .....	91
7.23.	rtk_l2_agingEnable_set .....	91
7.24.	rtk_l2_agingEnable_get.....	92
7.25.	rtk_l2_limitLearningCnt_set .....	92
7.26.	rtk_l2_limitLearningCnt_get .....	93
7.27.	rtk_l2_limitSystemLearningCnt_set .....	93
7.28.	rtk_l2_limitSystemLearningCnt_get .....	94
7.29.	rtk_l2_limitLearningCntAction_set .....	94
7.30.	rtk_l2_limitLearningCntAction_get .....	95
7.31.	rtk_l2_limitSystemLearningCntAction_set .....	95

7.32.	rtk_l2_limitSystemLearningCntAction_get .....	96
7.33.	rtk_l2_limitSystemLearningCntPortMask_set .....	97
7.34.	rtk_l2_limitSystemLearningCntPortMask_get .....	97
7.35.	rtk_l2_learningCnt_get .....	98
7.36.	rtk_l2_floodPortMask_set .....	98
7.37.	rtk_l2_floodPortMask_get .....	99
7.38.	rtk_l2_localPktPermit_set .....	99
7.39.	rtk_l2_localPktPermit_get .....	100
7.40.	rtk_l2_aging_set .....	100
7.41.	rtk_l2_aging_get .....	101
7.42.	rtk_l2_ipMcastAddrLookup_set .....	101
7.43.	rtk_l2_ipMcastAddrLookup_get .....	102
7.44.	rtk_l2_ipMcastForwardRouterPort_set .....	102
7.45.	rtk_l2_ipMcastForwardRouterPort_get .....	103
7.46.	rtk_l2_ipMcastGroupEntry_add .....	103
7.47.	rtk_l2_ipMcastGroupEntry_del .....	104
7.48.	rtk_l2_ipMcastGroupEntry_get .....	104
7.49.	rtk_l2_entry_get .....	105
7.50.	rtk_l2_lookupHitIsolationAction_set .....	105
7.51.	rtk_l2_lookupHitIsolationAction_get .....	106
8.	Module leaky.h - Unmanaged switch high-level API .....	106
8.1.	rtk_leaky_vlan_set .....	107
8.2.	rtk_leaky_vlan_get .....	108
8.3.	rtk_leaky_portIsolation_set .....	110
8.4.	rtk_leaky_portIsolation_get .....	112
9.	Module led.h - Unmanaged switch high-level API .....	114
9.1.	rtk_led_enable_set .....	114
9.2.	rtk_led_enable_get .....	115

9.3.	rtk_led_operation_set .....	115
9.4.	rtk_led_operation_get.....	116
9.5.	rtk_led_modeForce_set .....	116
9.6.	rtk_led_modeForce_get.....	117
9.7.	rtk_led_blinkRate_set.....	118
9.8.	rtk_led_blinkRate_get .....	118
9.9.	rtk_led_groupConfig_set.....	119
9.10.	rtk_led_groupConfig_get .....	120
9.11.	rtk_led_serialMode_set .....	120
9.12.	rtk_led_serialMode_get.....	121
9.13.	rtk_led_OutputEnable_set .....	121
9.14.	rtk_led_OutputEnable_get.....	122
9.15.	rtk_led_groupAbility_set.....	122
9.16.	rtk_led_groupAbility _get .....	123
9.17.	rtk_led_serialModePortmask_set .....	123
9.18.	rtk_led_serialModePortmask_get.....	124
10.	Module mirror.h - Unmanaged switch high-level API .....	124
10.1.	rtk_mirror_portBased_set.....	125
10.2.	rtk_mirror_portBased_get .....	125
10.3.	rtk_mirror_portIso_set.....	126
10.4.	rtk_mirror_portIso_get .....	126
10.5.	rtk_mirror_vlanLeaky_set .....	127
10.6.	rtk_mirror_vlanLeaky_get .....	127
10.7.	rtk_mirror_isolationLeaky_set .....	128
10.8.	rtk_mirror_isolationLeaky_get.....	128
10.9.	rtk_mirror_keep_set .....	129
10.10.	rtk_mirror_keep_get .....	129
10.11.	rtk_mirror_override_set.....	130

10.12. rtk_mirror_override_get .....	130
11. Module oam.h - Unmanaged switch high-level API.....	131
11.1. rtk_oam_init.....	132
11.2. rtk_oam_state_set .....	132
11.3. rtk_oam_state_get.....	132
11.4. rtk_oam_parserAction_set.....	133
11.5. rtk_oam_parserAction_get .....	133
11.6. rtk_oam_multiplexerAction_set .....	134
11.7. rtk_oam_multiplexerAction_get .....	134
12. Module port.h - Unmanaged switch high-level API.....	135
12.1. rtk_port_phyAutoNegoAbility_set.....	136
12.2. rtk_port_phyAutoNegoAbility_get .....	137
12.3. rtk_port_phyForceModeAbility_set .....	137
12.4. rtk_port_phyForceModeAbility_get .....	138
12.5. rtk_port_phyStatus_get .....	138
12.6. rtk_port_macForceLink_set .....	139
12.7. rtk_port_macForceLink_get .....	140
12.8. rtk_port_macForceLinkExt_set .....	140
12.9. rtk_port_macForceLinkExt_get .....	141
12.10. rtk_port_macStatus_get .....	142
12.11. rtk_port_macLocalLoopbackEnable_set .....	142
12.12. rtk_port_macLocalLoopbackEnable_get .....	143
12.13. rtk_port_phyReg_set .....	143
12.14. rtk_port_phyReg_get .....	144
12.15. rtk_port_backpressureEnable_set .....	144
12.16. rtk_port_backpressureEnable_get .....	145
12.17. rtk_port_adminEnable_set .....	146
12.18. rtk_port_adminEnable_get .....	146

12.19. rtk_port_isolation_set .....	147
12.20. rtk_port_isolation_get.....	147
12.21. rtk_port_rgmiiDelayExt_set.....	148
12.22. rtk_port_rgmiiDelayExt_get .....	149
12.23. rtk_port_phyEnableAll_set .....	149
12.24. rtk_port_phyEnableAll_get .....	150
12.25. rtk_port_efid_set.....	150
12.26. rtk_port_efid_get .....	151
12.27. rtk_port_phyComboPortMedia_set.....	151
12.28. rtk_port_phyComboPortMedia_get .....	152
12.29. rtk_port_rtctEnable_set .....	152
12.30. rtk_port_rtctDisable_set .....	153
12.31. rtk_port_rtctResult_get.....	153
12.32. rtk_port_sds_reset.....	154
12.33. rtk_port_sgmiiLinkStatus_get .....	154
12.34. rtk_port_sgmiiNway_set .....	155
12.35. rtk_port_sgmiiNway_get.....	155
13. Module ptp.h - Unmanaged switch high-level API .....	156
13.1. rtk_ptp_init .....	157
13.2. rtk_ptp_mac_set.....	157
13.3. rtk_ptp_mac_get .....	158
13.4. rtk_ptp_tpid_set.....	158
13.5. rtk_ptp_tpid_get.....	159
13.6. rtk_ptp_refTime_set .....	159
13.7. rtk_ptp_refTime_get .....	160
13.8. rtk_ptp_refTimeAjust_set .....	160
13.9. rtk_ptp_refTimeEnable_set .....	161
13.10. rtk_ptp_refTimeEnable_get.....	161

13.11.	rtk_ptp_portEnable_set.....	162
13.12.	rtk_ptp_portEnable_get .....	162
13.13.	rtk_ptp_portTimestamp_get .....	163
13.14.	rtk_ptp_intControl_set.....	163
13.15.	rtk_ptp_intControl_get .....	164
13.16.	rtk_ptp_intStatus_get.....	165
13.17.	rtk_ptp_portIntStatus_set .....	165
13.18.	rtk_ptp_portIntStatus_get .....	166
13.19.	rtk_ptp_portTrap_set .....	167
13.20.	rtk_ptp_portTrap_get.....	167
14.	Module qos.h - Unmanaged switch high-level API.....	168
14.1.	rtk_qos_init.....	169
14.2.	rtk_qos_priSel_set .....	169
14.3.	rtk_qos_priSel_get.....	170
14.4.	rtk_qos_1pPriRemap_set.....	171
14.5.	rtk_qos_1pPriRemap_get .....	171
14.6.	rtk_qos_1pRemarkSrcSel_set.....	172
14.7.	rtk_qos_1pRemarkSrcSel_get .....	172
14.8.	rtk_qos_dscpPriRemap_set .....	173
14.9.	rtk_qos_dscpPriRemap_get .....	174
14.10.	rtk_qos_portPri_set.....	174
14.11.	rtk_qos_portPri_get .....	175
14.12.	rtk_qos_queueNum_set .....	175
14.13.	rtk_qos_queueNum_get.....	176
14.14.	rtk_qos_priMap_set .....	176
14.15.	rtk_qos_priMap_get.....	177
14.16.	rtk_qos_schedulingQueue_set .....	178
14.17.	rtk_qos_schedulingQueue_get .....	178

14.18. rtk_qos_1pRemarkEnable_set.....	179
14.19. rtk_qos_1pRemarkEnable_get .....	180
14.20. rtk_qos_1pRemark_set.....	180
14.21. rtk_qos_1pRemark_get.....	181
14.22. rtk_qos_dscpRemarkEnable_set .....	181
14.23. rtk_qos_dscpRemarkEnable_get .....	182
14.24. rtk_qos_dscpRemark_set.....	182
14.25. rtk_qos_dscpRemark_get .....	183
14.26. rtk_qos_dscpRemarkSrcSel_set .....	183
14.27. rtk_qos_dscpRemarkSrcSel_get.....	184
14.28. rtk_qos_dscpRemark2Dscp_set .....	185
14.29. rtk_qos_dscpRemark2Dscp_get.....	185
14.30. rtk_qos_portPriSelIndex_set .....	186
14.31. rtk_qos_portPriSelIndex_get .....	186
14.32. rtk_qos_schedulingType_set .....	187
14.33. rtk_qos_schedulingType_get.....	187
15. Module rate.h - Unmanaged switch high-level API .....	188
15.1. rtk_rate_shareMeter_set .....	188
15.2. rtk_rate_shareMeter_get .....	189
15.3. rtk_rate_shareMeterBucket_set .....	190
15.4. rtk_rate_shareMeterBucket_get .....	190
15.5. rtk_rate_igrBandwidthCtrlRate_set .....	191
15.6. rtk_rate_igrBandwidthCtrlRate_get .....	192
15.7. rtk_rate_egrBandwidthCtrlRate_set .....	192
15.8. rtk_rate_egrBandwidthCtrlRate_get .....	193
15.9. rtk_rate_egrQueueBwCtrlEnable_set .....	194
15.10. rtk_rate_egrQueueBwCtrlEnable_get .....	194
15.11. rtk_rate_egrQueueBwCtrlRate_set .....	195

15.12. rtk_rate_egressQueueBwCtrlRate_get .....	195
16. Module rldp.h - Declaration of RLDP and RLPP API .....	196
16.1. rtk_rldp_config_set.....	197
16.2. rtk_rldp_config_get .....	197
16.3. rtk_rldp_portConfig_set .....	198
16.4. rtk_rldp_portConfig_get.....	198
16.5. rtk_rldp_status_get .....	199
16.6. rtk_rldp_portStatus_get .....	199
16.7. rtk_rldp_portStatus_set.....	200
16.8. rtk_rldp_portLoopPair_get .....	200
17. Module rtk_switch.h - Definition function prototype of RTK switch API .....	201
17.1. rtk_switch_probe .....	202
17.2. rtk_switch_initialState_set .....	202
17.3. rtk_switch_initialState_get .....	202
17.4. rtk_switch_logicalPortCheck .....	203
17.5. rtk_switch_isUtpPort .....	203
17.6. rtk_switch_isExtPort .....	204
17.7. rtk_switch_isHsgPort.....	204
17.8. rtk_switch_isComboPort .....	204
17.9. rtk_switch_ComboPort_get.....	205
17.10. rtk_switch_port_L2P_get .....	205
17.11. rtk_switch_port_P2L_get .....	206
17.12. rtk_switch_isPortMaskValid .....	206
17.13. rtk_switch_isPortMaskUtp .....	206
17.14. rtk_switch_isPortMaskExt .....	207
17.15. rtk_switch_portmask_L2P_get.....	207
17.16. rtk_switch_portmask_P2L_get.....	208
17.17. rtk_switch_phyPortMask_get.....	208

17.18. rtk_switch_logPortMask_get.....	209
17.19. rtk_switch_init.....	209
17.20. rtk_switch_portMaxPktLen_set .....	210
17.21. rtk_switch_portMaxPktLen_get.....	210
17.22. rtk_switch_maxPktLenCfg_set .....	211
17.23. rtk_switch_maxPktLenCfg_get.....	211
17.24. rtk_switch_greenEthernet_set .....	212
17.25. rtk_switch_greenEthernet_get.....	212
17.26. rtk_switch_maxLogicalPort_get .....	213
18. Module stat.h - Unmanaged switch high-level API.....	213
18.1. rtk_stat_global_reset.....	214
18.2. rtk_stat_port_reset .....	214
18.3. rtk_stat_queueManage_reset .....	215
18.4. rtk_stat_global_get .....	215
18.5. rtk_stat_global_getAll .....	216
18.6. rtk_stat_port_get .....	216
18.7. rtk_stat_port_getAll.....	217
18.8. rtk_stat_logging_counterCfg_set .....	217
18.9. rtk_stat_logging_counterCfg_get.....	218
18.10. rtk_stat_logging_counter_reset .....	218
18.11. rtk_stat_logging_counter_get .....	219
18.12. rtk_stat_lengthMode_set .....	219
18.13. rtk_stat_lengthMode_get .....	220
19. Module storm.h - Unmanaged switch high-level API .....	220
19.1. rtk_rate_stormControlMeterIdx_set.....	221
19.2. rtk_rate_stormControlMeterIdx_get .....	221
19.3. rtk_rate_stormControlPortEnable_set .....	222
19.4. rtk_rate_stormControlPortEnable_get .....	223

19.5.	rtk_storm_bypass_set .....	223
19.6.	rtk_storm_bypass_get.....	225
19.7.	rtk_rate_stormControlExtPortmask_set .....	226
19.8.	rtk_rate_stormControlExtPortmask_get.....	227
19.9.	rtk_rate_stormControlExtEnable_set .....	227
19.10.	rtk_rate_stormControlExtEnable_get.....	228
19.11.	rtk_rate_stormControlExtMeterIdx_set .....	228
19.12.	rtk_rate_stormControlExtMeterIdx_get.....	229
20.	Module svlan.h - Unmanaged switch high-level API .....	230
20.1.	rtk_svlan_init .....	231
20.2.	rtk_svlan_servicePort_add .....	231
20.3.	rtk_svlan_servicePort_get .....	232
20.4.	rtk_svlan_servicePort_del .....	232
20.5.	rtk_svlan_tpidEntry_set.....	233
20.6.	rtk_svlan_tpidEntry_get .....	233
20.7.	rtk_svlan_priorityRef_set.....	234
20.8.	rtk_svlan_priorityRef_get.....	234
20.9.	rtk_svlan_memberPortEntry_set .....	235
20.10.	rtk_svlan_memberPortEntry_get.....	235
20.11.	rtk_svlan_memberPortEntry_adv_set .....	236
20.12.	rtk_svlan_memberPortEntry_adv_get.....	237
20.13.	rtk_svlan_defaultSvlan_set.....	237
20.14.	rtk_svlan_defaultSvlan_get .....	238
20.15.	rtk_svlan_c2s_add .....	238
20.16.	rtk_svlan_c2s_del .....	239
20.17.	rtk_svlan_c2s_get .....	240
20.18.	rtk_svlan_untag_action_set .....	240
20.19.	rtk_svlan_untag_action_get.....	241

20.20. rtk_svlan_unmatch_action_set .....	242
20.21. rtk_svlan_unmatch_action_get .....	242
20.22. rtk_svlan_dmac_vidsel_set .....	243
20.23. rtk_svlan_dmac_vidsel_get .....	244
20.24. rtk_svlan_ipmc2s_add .....	244
20.25. rtk_svlan_ipmc2s_del .....	245
20.26. rtk_svlan_ipmc2s_get .....	246
20.27. rtk_svlan_l2mc2s_add .....	246
20.28. rtk_svlan_l2mc2s_del .....	247
20.29. rtk_svlan_l2mc2s_get .....	247
20.30. rtk_svlan_sp2c_add .....	248
20.31. rtk_svlan_sp2c_get .....	249
20.32. rtk_svlan_sp2c_del .....	249
20.33. rtk_svlan_lookupType_set .....	250
20.34. rtk_svlan_lookupType_get .....	250
20.35. rtk_svlan_trapPri_set .....	251
20.36. rtk_svlan_trapPri_get .....	251
20.37. rtk_svlan_unassign_action_set .....	252
20.38. rtk_svlan_unassign_action_get .....	252
20.39. rtk_svlan_checkAndCreateMbr .....	253
21. Module trap.h - Unmanaged switch high-level API .....	253
21.1. rtk_trap_unknownUnicastPktAction_set .....	254
21.2. rtk_trap_unknownUnicastPktAction_get .....	255
21.3. rtk_trap_unknownMacPktAction_set .....	255
21.4. rtk_trap_unknownMacPktAction_get .....	256
21.5. rtk_trap_unmatchMacPktAction_set .....	256
21.6. rtk_trap_unmatchMacPktAction_get .....	257
21.7. rtk_trap_unmatchMacMoving_set .....	258

21.8. rtk_trap_unmatchMacMoving_get.....	258
21.9. rtk_trap_unknownMcastPktAction_set.....	259
21.10. rtk_trap_unknownMcastPktAction_get .....	260
21.11. rtk_trap_lldpEnable_set.....	260
21.12. rtk_trap_lldpEnable_get .....	261
21.13. rtk_trap_reasonTrapToCpuPriority_set.....	262
21.14. rtk_trap_reasonTrapToCpuPriority_get .....	262
21.15. rtk_trap_rmaAction_set .....	263
21.16. rtk_trap_rmaAction_get .....	265
21.17. rtk_trap_rmaKeepFormat_set.....	267
21.18. rtk_trap_rmaKeepFormat_get .....	268
21.19. rtk_trap_portUnknownMacPktAction_set .....	270
21.20. rtk_trap_portUnknownMacPktAction_get .....	271
21.21. rtk_trap_portUnmatchMacPktAction_set .....	271
21.22. rtk_trap_portUnmatchMacPktAction_get .....	272
22. Module trunk.h - Unmanaged switch high-level API .....	272
22.1. rtk_trunk_port_set .....	273
22.2. rtk_trunk_port_get.....	274
22.3. rtk_trunk_distributionAlgorithm_set.....	274
22.4. rtk_trunk_distributionAlgorithm_get .....	275
22.5. rtk_trunk_queueEmptyStatus_get .....	276
22.6. rtk_trunk_trafficSeparate_set .....	276
22.7. rtk_trunk_trafficSeparate_get.....	277
22.8. rtk_trunk_mode_set .....	277
22.9. rtk_trunk_mode_get.....	278
22.10. rtk_trunk_trafficPause_set.....	278
22.11. rtk_trunk_trafficPause_get .....	279
22.12. rtk_trunk_hashMappingTable_set.....	279

22.13. rtk_trunk_hashMappingTable_get .....	280
22.14. rtk_trunk_portQueueEmpty_get.....	280
23. Module vlan.h - Unmanaged switch high-level API .....	281
23.1. rtk_vlan_init.....	282
23.2. rtk_vlan_set .....	282
23.3. rtk_vlan_get .....	283
23.4. rtk_vlan_egressFilterEnable_set .....	283
23.5. rtk_vlan_egressFilterEnable_get.....	284
23.6. rtk_vlan_mbrCfg_set.....	284
23.7. rtk_vlan_mbrCfg_get .....	285
23.8. rtk_vlan_portPvid_set.....	285
23.9. rtk_vlan_portPvid_get .....	286
23.10. rtk_vlan_portIgrFilterEnable_set .....	287
23.11. rtk_vlan_portIgrFilterEnable_get.....	287
23.12. rtk_vlan_portAcceptFrameType_set.....	288
23.13. rtk_vlan_portAcceptFrameType_get .....	289
23.14. rtk_vlan_tagMode_set .....	289
23.15. rtk_vlan_tagMode_get.....	290
23.16. rtk_vlan_transparent_set.....	291
23.17. rtk_vlan_transparent_get .....	291
23.18. rtk_vlan_keep_set.....	292
23.19. rtk_vlan_keep_get .....	292
23.20. rtk_vlan_stg_set.....	293
23.21. rtk_vlan_stg_get .....	293
23.22. rtk_vlan_protoAndPortBasedVlan_add .....	294
23.23. rtk_vlan_protoAndPortBasedVlan_get .....	295
23.24. rtk_vlan_protoAndPortBasedVlan_del .....	295
23.25. rtk_vlan_protoAndPortBasedVlan_deleteAll .....	296

23.26. rtk_vlan_portFid_set.....	297
23.27. rtk_vlan_portFid_get .....	297
23.28. rtk_vlan_UntagDscpPriorityEnable_set.....	298
23.29. rtk_vlan_UntagDscpPriorityEnable_get .....	298
23.30. rtk_stp_mstpState_set.....	299
23.31. rtk_stp_mstpState_get .....	300
23.32. rtk_vlan_checkAndCreateMbr .....	300
23.33. rtk_vlan_reservedVidAction_set.....	301
23.34. rtk_vlan_reservedVidAction_get .....	301
23.35. rtk_vlan_realKeepRemarkEnable_set.....	302
23.36. rtk_vlan_realKeepRemarkEnable_get.....	302
23.37. rtk_vlan_reset .....	303
24. Module i2c.h - Unmanaged switch high-level API.....	303
24.1. rtk_i2c_data_read .....	304
24.2. rtk_i2c_data_write .....	304
24.3. rtk_i2c_init.....	305
24.4. rtk_i2c_mode_set.....	305
24.5. rtk_i2c_mode_get .....	306
24.6. rtk_i2c_gpioPinGroup_set.....	306
24.7. rtk_i2c_gpioPinGroup_get .....	306

---

# 1. Module acl.h - Unmanaged switch high-level API

Filename: acl.h

## Description

The file includes ACL module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

## List of Symbols

Here is a list of all functions and variables in this module

acl.h - Unmanaged switch high-level API  
rtk\_filter\_igrAcl\_init  
rtk\_filter\_igrAcl\_field\_add  
rtk\_filter\_igrAcl\_cfg\_add  
rtk\_filter\_igrAcl\_cfg\_del  
rtk\_filter\_igrAcl\_cfg\_delAll  
rtk\_filter\_igrAcl\_cfg\_get  
rtk\_filter\_igrAcl\_unmatchAction\_set  
rtk\_filter\_igrAcl\_unmatchAction\_get  
rtk\_filter\_igrAcl\_state\_set  
rtk\_filter\_igrAcl\_state\_get  
rtk\_filter\_igrAcl\_template\_set  
rtk\_filter\_igrAcl\_template\_get  
rtk\_filter\_igrAcl\_field\_sel\_set  
rtk\_filter\_igrAcl\_field\_sel\_get  
rtk\_filter\_iprange\_set  
rtk\_filter\_iprange\_get  
rtk\_filter\_vidrange\_set  
rtk\_filter\_vidrange\_get  
rtk\_filter\_portrange\_set  
rtk\_filter\_portrange\_get  
rtk\_filter\_igrAclPolarity\_set  
rtk\_filter\_igrAclPolarity\_get

---

## 1.1. rtk\_filter\_igrAcl\_init

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_init( void)**

ACL initialization function

Defined in: acl.h

**Parameters**

*void*

**Comments**

This function enable and initialize ACL function

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Pointer pFilter_field or pFilter_cfg point to NULL.

---

## 1.2. rtk\_filter\_igrAcl\_field\_add

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_field\_add(rtк\_filter\_cfg\_t \*pFilter\_cfg,  
rtк\_filter\_field\_t \*pFilter\_field)**

Add comparison rule to an ACL configuration

Defined in: acl.h

**Parameters**

*\*pFilter\_cfg*

The ACL configuration that this function will add comparison rule

*\*pFilter\_field*

The comparison rule that will be added.

This function add a comparison rule (*\*pFilter\_field*) to an ACL configuration (*\*pFilter\_cfg*). Pointer pFilter\_cfg points to an ACL configuration structure, this structure keeps multiple ACL comparison rules by means of linked list. Pointer pFilter\_field will be added to linked list kepted by structure that pFilter\_cfg points to.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Pointer pFilter_field or pFilter_cfg point to NULL.

---

RT_ERR_INPUT	Invalid input parameters.
--------------	---------------------------

---

### 1.3. rtk\_filter\_igrAcl\_cfg\_add

`rtk_api_ret_t rtk_filter_igrAcl_cfg_add(rtk_filter_id_t filter_id,  
rtk_filter_cfg_t *pFilter_cfg, rtk_filter_action_t *pAction,  
rtk_filter_number_t *ruleNum)`

Add an ACL configuration to ASIC

Defined in: acl.h

#### Parameters

*filter\_id*

Start index of ACL configuration.

*\*pFilter\_cfg*

The ACL configuration that this function will add comparison rule

*\*pAction*

Action(s) of ACL configuration.

*\*ruleNum*

number of rules written in acl table

#### Comments

This function store pFilter\_cfg, pFilter\_action into ASIC. The starting index(es) is filter\_id.

#### Return Codes

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_NULL\_POINTER

Pointer pFilter\_field or pFilter\_cfg point to NULL.

RT\_ERR\_INPUT

Invalid input parameters.

RT\_ERR\_ENTRY\_INDEX

Invalid filter\_id .

RT\_ERR\_NULL\_POINTER

Pointer pFilter\_action or pFilter\_cfg point to NULL.

RT\_ERR\_FILTER\_INACL\_ACT\_NOT\_SUPPORT

Action is not supported in this chip.

RT\_ERR\_FILTER\_INACL\_RULE\_NO\_T\_SUPPORT

Rule is not supported.

---

## 1.4. rtk\_filter\_igrAcl\_cfg\_del

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_cfg\_del(rtk\_filter\_id\_t filter\_id)**

Delete an ACL configuration from ASIC

Defined in: acl.h

**Parameters**

*filter\_id*

Start index of ACL configuration.

**Comments**

This function delete a group of ACL rules starting from filter\_id.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_FILTER\_ENTRYIDX

Invalid filter\_id.

---

## 1.5. rtk\_filter\_igrAcl\_cfg\_delAll

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_cfg\_delAll( void)**

Delete all ACL entries from ASIC

Defined in: acl.h

**Parameters**

*void*

**Comments**

This function delete all ACL configuration from ASIC.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 1.6. rtk\_filter\_igrAcl\_cfg\_get

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_cfg\_get(rtk\_filter\_id\_t filter\_id,  
rtk\_filter\_cfg\_raw\_t \*pFilter\_cfg, rtk\_filter\_action\_t \*pAction)**

---

	Get one ingress acl configuration from ASIC.											
	Defined in: acl.h											
<b>Parameters</b>	<p><i>filter_id</i> Start index of ACL configuration.</p> <p><i>*pFilter_cfg</i> buffer pointer of ingress acl data</p> <p><i>*pAction</i> buffer pointer of ingress acl action</p>											
<b>Comments</b>	This function delete all ACL configuration from ASIC.											
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_NULL_POINTER</td> <td>Pointer pFilter_action or pFilter_cfg point to NULL.</td> </tr> <tr> <td>RT_ERR_FILTER_ENTRYIDX</td> <td>Invalid entry index.</td> </tr> </table>		RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_NULL_POINTER	Pointer pFilter_action or pFilter_cfg point to NULL.	RT_ERR_FILTER_ENTRYIDX	Invalid entry index.
RT_ERR_OK	ok											
RT_ERR_FAILED	failed											
RT_ERR_SMI	SMI access error											
RT_ERR_NULL_POINTER	Pointer pFilter_action or pFilter_cfg point to NULL.											
RT_ERR_FILTER_ENTRYIDX	Invalid entry index.											

---

## 1.7. rtk\_filter\_igrAcl\_unmatchAction\_set

`rtk_api_ret_t rtk_filter_igrAcl_unmatchAction_set(rtk_port_t port,  
rtk_filter_unmatch_action_t action)`

Set action to packets when no ACL configuration match

	Defined in: acl.h											
<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>action</i> Action.</p>											
<b>Comments</b>	This function sets action of packets when no ACL configuration matches.											
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port id.</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>		RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port id.	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok											
RT_ERR_FAILED	failed											
RT_ERR_SMI	SMI access error											
RT_ERR_PORT_ID	Invalid port id.											
RT_ERR_INPUT	Invalid input parameters.											

## 1.8. rtk\_filter\_igrAcl\_unmatchAction\_get

`rtk_api_ret_t rtk_filter_igrAcl_unmatchAction_get(rtk_port_t port,  
rtk_filter_unmatch_action_t* action)`

Get action to packets when no ACL configuration match

Defined in: acl.h

**Parameters**

*port*  
Port id.

*action*  
Action.

**Comments**

This function gets action of packets when no ACL configuration matches.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port id.
RT_ERR_INPUT	Invalid input parameters.

## 1.9. rtk\_filter\_igrAcl\_state\_set

`rtk_api_ret_t rtk_filter_igrAcl_state_set(rtk_port_t port, rtk_filter_state_t state)`

Set state of ingress ACL.

Defined in: acl.h

**Parameters**

*port*  
Port id.

*state*  
Ingress ACL state.

**Comments**

This function gets action of packets when no ACL configuration matches.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port id.

---

RT_ERR_INPUT	Invalid input parameters.
--------------	---------------------------

---

## 1.10. rtk\_filter\_igrAcl\_state\_get

`rtk_api_ret_t rtk_filter_igrAcl_state_get(rtк_port_t port, rtk_filter_state_t* state)`

Get state of ingress ACL.

Defined in: acl.h

**Parameters**

*port*

Port id.

*state*

Ingress ACL state.

**Comments**

This function gets action of packets when no ACL configuration matches.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port id.

RT\_ERR\_INPUT

Invalid input parameters.

---

## 1.11. rtk\_filter\_igrAcl\_template\_set

`rtk_api_ret_t rtk_filter_igrAcl_template_set(rtк_filter_template_t* aclTemplate)`

Set template of ingress ACL.

Defined in: acl.h

**Parameters**

*\*aclTemplate*

Ingress ACL template

**Comments**

This function set ACL template.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_INPUT

Invalid input parameters.

## 1.12. rtk\_filter\_igrAcl\_template\_get

`rtk_api_ret_t rtk_filter_igrAcl_template_get(rtk_filter_template_t *aclTemplate)`

Get template of ingress ACL.

Defined in: acl.h

**Parameters**

*\*aclTemplate*  
Ingress ACL template

**Comments**

This function gets template of ACL.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

## 1.13. rtk\_filter\_igrAcl\_field\_sel\_set

`rtk_api_ret_t rtk_filter_igrAcl_field_sel_set(rtк_uint32 index, rtk_field_sel_t format, rtк_uint32 offset)`

Set user defined field selectors in HSB

Defined in: acl.h

**Parameters**

*index*  
index of field selector 0  
*format*  
Format of field selector  
*offset*  
Retrieving data offset

**Comments**

System support 16 user defined field selctors. Each selector can be enabled or disable. User can defined retrieving 16-bits in many predefiend standard l2/l3/l4 payload.

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 1.14. rtk\_filter\_igrAcl\_field\_sel\_get

**rtk\_api\_ret\_t rtk\_filter\_igrAcl\_field\_sel\_get(rtk\_uint32 *index*, rtk\_field\_sel\_t \**pFormat*, rtk\_uint32 \**pOffset*)**

Get user defined field selectors in HSB

Defined in: acl.h

**Parameters**

*index*  
index of field selector 0  
*\*pFormat*  
Format of field selector  
*\*pOffset*  
Retrieving data offset

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 1.15. rtk\_filter\_iprange\_set

**rtk\_api\_ret\_t rtk\_filter\_iprange\_set(rtk\_uint32 *index*, rtk\_filter\_iprange\_t *type*, ipaddr\_t *upperIp*, ipaddr\_t *lowerIp*)**

Set IP Range check

Defined in: acl.h

**Parameters**

*index*  
index of IP Range 0  
*type*  
IP Range check type, 0:Delete a entry, 1: IPv4\_SIP, 2: IPv4\_DIP, 3:IPv6\_SIP,  
4:IPv6\_DIP

<i>upperIp</i>	The upper bound of IP range
<i>lowerIp</i>	The lower Bound of IP range
<b>Comments</b>	upperIp must be larger or equal than lowerIp.
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_OUT_OF_RANGE	The parameter is out of range
RT_ERR_INPUT	Input error

---

## 1.16. rtk\_filter\_iprange\_get

`rtk_api_ret_t rtk_filter_iprange_get(rtk_uint32 index, rtk_filter_iprange_t *pType, ipaddr_t *pUpperIp, ipaddr_t *pLowerIp)`

Set IP Range check

Defined in: acl.h

<b>Parameters</b>	
<i>index</i>	index of IP Range 0
<i>*pType</i>	IP Range check type, 0:Delete a entry, 1: IPv4_SIP, 2: IPv4_DIP, 3:IPv6_SIP, 4:IPv6_DIP
<i>*pUpperIp</i>	The upper bound of IP range
<i>*pLowerIp</i>	The lower Bound of IP range
<b>Comments</b>	upperIp must be larger or equal than lowerIp.
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_OUT_OF_RANGE	The parameter is out of range

---

## 1.17. rtk\_filter\_vidrange\_set

`rtk_api_ret_t rtk_filter_vidrange_set(rtk_uint32 index, rtk_filter_vidrange_t type, rtk_uint32 upperVid, rtk_uint32 lowerVid)`

Set VID Range check

Defined in: acl.h

### Parameters

*index*  
index of VID Range 0

*type*  
IP Range check type, 0:Delete a entry, 1:CVID, 2:SVID

*upperVid*  
The upper bound of VID range

*lowerVid*  
The lower Bound of VID range

### Comments

upperVid must be larger or equal than lowerVid.

### Return Codes

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_OUT_OF_RANGE</code>	The parameter is out of range
<code>RT_ERR_INPUT</code>	Input error

---

## 1.18. rtk\_filter\_vidrange\_get

`rtk_api_ret_t rtk_filter_vidrange_get(rtk_uint32 index, rtk_filter_vidrange_t *pType, rtk_uint32 *pUpperVid, rtk_uint32 *pLowerVid)`

Get VID Range check

Defined in: acl.h

### Parameters

*index*  
index of VID Range 0

*\*pType*  
IP Range check type, 0:Unused, 1:CVID, 2:SVID

*\*pUpperVid*  
The upper bound of VID range

<i>*pLowerVid</i>	The lower Bound of VID range								
<b>Comments</b>	None.								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_OUT_OF_RANGE</td><td>The parameter is out of range</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_OUT_OF_RANGE	The parameter is out of range
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_OUT_OF_RANGE	The parameter is out of range								

---

### 1.19. rtk\_filter\_portrange\_set

`rtk_api_ret_t rtk_filter_portrange_set(rtк_uint32 index,  
rtk_filter_portrange_t type, rtк_uint32 upperPort, rtк_uint32 lowerPort)`

Set Port Range check

Defined in: acl.h

#### Parameters

*index*  
index of Port Range 0

*type*  
IP Range check type, 0:Delete a entry, 1: Source Port, 2: Destination Port

*upperPort*  
The upper bound of Port range

*lowerPort*  
The lower Bound of Port range  
upperPort must be larger or equal than lowerPort.

#### Comments

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_OUT_OF_RANGE	The parameter is out of range
RT_ERR_INPUT	Input error

---

---

## 1.20. rtk\_filter\_portrange\_get

`rtk_api_ret_t rtk_filter_portrange_get(rtk_uint32 index,  
rtk_filter_portrange_t *pType, rtk_uint32 *pUpperPort, rtk_uint32  
*pLowerPort)`

Set Port Range check

Defined in: acl.h

**Parameters**

*index*  
index of Port Range 0

*\*pType*  
IP Range check type, 0:Delete a entry, 1: Source Port, 2: Destination Port

*\*pUpperPort*  
The upper bound of Port range

*\*pLowerPort*  
The lower Bound of Port range

**Comments**

None.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_OUT_OF_RANGE</code>	The parameter is out of range
<code>RT_ERR_INPUT</code>	Input error

---

## 1.21. rtk\_filter\_ligrAclPolarity\_set

`rtk_api_ret_t rtk_filter_ligrAclPolarity_set(rtk_uint32 polarity)`

Set ACL Goip control polarity

Defined in: acl.h

**Parameters**

*polarity*  
1: High, 0: Low

**Comments**

none

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_SMI</code>	SMI access error

---

## 1.22. rtk\_filter\_igrAclPolarity\_get

`rtk_api_ret_t rtk_filter_igrAclPolarity_get(rtk_uint32* pPolarity)`

Get ACL Goip control polarity

Defined in: acl.h

**Parameters**

*pPolarity*

1: High, 0: Low

**Comments**

none

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_SMI

SMI access error

---

## 2. Module cpu.h - Unmanaged switch high-level API

Filename: cpu.h

**Description**

The file includes CPU module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

cpu.h - Unmanaged switch high-level API  
rtk\_cpu\_enable\_set  
rtk\_cpu\_enable\_get  
rtk\_cpu\_tagPort\_set  
rtk\_cpu\_tagPort\_get  
rtk\_cpu\_awarePort\_set  
rtk\_cpu\_awarePort\_get  
rtk\_cpu\_tagPosition\_set  
rtk\_cpu\_tagPosition\_get  
rtk\_cpu\_tagLength\_set  
rtk\_cpu\_tagLength\_get

---

rtk\_cpu\_acceptLength\_set  
rtk\_cpu\_acceptLength\_get  
rtk\_cpu\_priRemap\_set  
rtk\_cpu\_priRemap\_get

---

## 2.1. rtk\_cpu\_enable\_set

`rtk_api_ret_t rtk_cpu_enable_set(rtk_enable_t enable)`

Set CPU port function enable/disable.

Defined in: cpu.h

**Parameters**

*enable*  
CPU port function enable

**Comments**

The API can set CPU port function enable/disable.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_PORT_ID	Invalid port number.

---

## 2.2. rtk\_cpu\_enable\_get

`rtk_api_ret_t rtk_cpu_enable_get(rtk_enable_t *pEnable)`

Get CPU port and its setting.

Defined in: cpu.h

**Parameters**

*\*pEnable*  
CPU port function enable

**Comments**

The API can get CPU port function enable/disable.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

RT_ERR_L2_NO_CPU_PORT	CPU port is not exist
-----------------------	-----------------------

### 2.3. rtk\_cpu\_tagPort\_set

**rtk\_api\_ret\_t rtk\_cpu\_tagPort\_set(rtk\_port\_t *port*, rtk\_cpu\_insert\_t *mode*)**

Set CPU port and CPU tag insert mode.

Defined in: cpu.h

#### Parameters

*port*  
Port id.

*mode*  
CPU tag insert for packets egress from CPU port.

#### Comments

The API can set CPU port and inserting proprietary CPU tag mode (Length/Type 0x8899) to the frame that transmitting to CPU port. The inset cpu tag mode is as following:

- CPU\_INSERT\_TO\_ALL
- CPU\_INSERT\_TO\_TRAPPING
- CPU\_INSERT\_TO\_NONE

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_PORT_ID	Invalid port number.

### 2.4. rtk\_cpu\_tagPort\_get

**rtk\_api\_ret\_t rtk\_cpu\_tagPort\_get(rtk\_port\_t \**pPort*, rtk\_cpu\_insert\_t \**pMode*)**

Get CPU port and CPU tag insert mode.

Defined in: cpu.h

#### Parameters

\**pPort*  
Port id.

---

	<i>*pMode</i>	CPU tag insert for packets egress from CPU port, 0:all insert 1:Only for trapped packets 2:no insert.
<b>Comments</b>	The API can get configured CPU port and its setting. The inset cpu tag mode is as following:	
	- CPU_INSERT_TO_ALL	
	- CPU_INSERT_TO_TRAPPING	
	- CPU_INSERT_TO_NONE	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT RT_ERR_L2_NO_CPU_PORT	ok failed SMI access error Invalid input parameters. CPU port is not exist

---

## 2.5. rtk\_cpu\_awarePort\_set

`rtk_api_ret_t rtk_cpu_awarePort_set(rtk_portmask_t *pPortmask)`

Set CPU aware port mask.

Defined in: cpu.h

<b>Parameters</b>	<i>*pPortmask</i>	Port mask.
<b>Comments</b>	The API can set configured CPU aware port mask.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_MASK	ok failed SMI access error Invalid port mask.

---

## 2.6. rtk\_cpu\_awarePort\_get

`rtk_api_ret_t rtk_cpu_awarePort_get(rtk_portmask_t *pPortmask)`

Get CPU aware port mask.

Defined in: cpu.h

<b>Parameters</b>	<i>*pPortmask</i> Port mask.						
<b>Comments</b>	The API can get configured CPU aware port mask.						
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error
RT_ERR_OK	ok						
RT_ERR_FAILED	failed						
RT_ERR_SMI	SMI access error						

---

## 2.7. rtk\_cpu\_tagPosition\_set

`rtk_api_ret_t rtk_cpu_tagPosition_set(rtk_cpu_position_t position)`

Set CPU tag position.

Defined in: cpu.h

<b>Parameters</b>	<i>position</i> CPU tag position.								
<b>Comments</b>	The API can set CPU tag position.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input.								

---

## 2.8. rtk\_cpu\_tagPosition\_get

`rtk_api_ret_t rtk_cpu_tagPosition_get(rtk_cpu_position_t *pPosition)`

Get CPU tag position.

Defined in: cpu.h

<b>Parameters</b>	<i>*pPosition</i> CPU tag position.				
<b>Comments</b>	The API can get CPU tag position.				
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed
RT_ERR_OK	ok				
RT_ERR_FAILED	failed				

---

RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input.

---

## 2.9. rtk\_cpu\_tagLength\_set

`rtk_api_ret_t rtk_cpu_tagLength_set(rtk_cpu_tag_length_t length)`

Set CPU tag length.

Defined in: cpu.h

**Parameters**

*length*  
CPU tag length.

**Comments**

The API can set CPU tag length.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input.

## 2.10. rtk\_cpu\_tagLength\_get

`rtk_api_ret_t rtk_cpu_tagLength_get(rtk_cpu_tag_length_t *pLength)`

Get CPU tag length.

Defined in: cpu.h

**Parameters**

*\*pLength*  
CPU tag length.

**Comments**

The API can get CPU tag length.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input.

---

## 2.11. rtk\_cpu\_acceptLength\_set

`rtk_api_ret_t rtk_cpu_acceptLength_set(rtk_cpu_rx_length_t length)`

Set CPU accept length.

Defined in: cpu.h

**Parameters**

*length*  
CPU tag length.

**Comments**

The API can set CPU accept length.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input.

---

## 2.12. rtk\_cpu\_acceptLength\_get

`rtk_api_ret_t rtk_cpu_acceptLength_get(rtk_cpu_rx_length_t *pLength)`

Get CPU accept length.

Defined in: cpu.h

**Parameters**

*\*pLength*  
CPU tag length.

**Comments**

The API can get CPU accept length.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input.

---

## 2.13. rtk\_cpu\_priRemap\_set

`rtk_api_ret_t rtk_cpu_priRemap_set(rtk_pri_t int_pri, rtk_pri_t new_pri)`

---

	Configure CPU priorities mapping to internal absolute priority.	
	Defined in: cpu.h	
<b>Parameters</b>	<i>int_pri</i> internal priority value.  <i>new_pri</i> new internal priority value.	
<b>Comments</b>	Priority of CPU tag assignment for internal asic priority, and it is used for queue usage and packet scheduling.	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters. RT_ERR_VLAN_PRIORITY Invalid 1p priority. RT_ERR_QOS_INT_PRIORITY Invalid priority.	

---

## 2.14. rtk\_cpu\_priRemap\_get

`rtk_api_ret_t rtk_cpu_priRemap_get(rtk_pri_t int_pri, rtk_pri_t *pNew_pri)`

Configure CPU priorities mapping to internal absolute priority.

Defined in: cpu.h

<b>Parameters</b>	<i>int_pri</i> internal priority value.	<i>*pNew_pri</i> new internal priority value.
<b>Comments</b>	Priority of CPU tag assignment for internal asic priority, and it is used for queue usage and packet scheduling.	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters. RT_ERR_VLAN_PRIORITY Invalid 1p priority. RT_ERR_QOS_INT_PRIORITY Invalid priority.	

---

### 3. Module dot1x.h - Unmanaged switch high-level API

Filename: dot1x.h

**Description** The file includes 1X module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

dot1x.h - Unmanaged switch high-level API  
rtk\_dot1x\_unauthPacketOper\_set  
rtk\_dot1x\_unauthPacketOper\_get  
rtk\_dot1x\_eapolFrame2CpuEnable\_set  
rtk\_dot1x\_eapolFrame2CpuEnable\_get  
rtk\_dot1x\_portBasedEnable\_set  
rtk\_dot1x\_portBasedEnable\_get  
rtk\_dot1x\_portBasedAuthStatus\_set  
rtk\_dot1x\_portBasedAuthStatus\_get  
rtk\_dot1x\_portBasedDirection\_set  
rtk\_dot1x\_portBasedDirection\_get  
rtk\_dot1x\_macBasedEnable\_set  
rtk\_dot1x\_macBasedEnable\_get  
rtk\_dot1x\_macBasedAuthMac\_add  
rtk\_dot1x\_macBasedAuthMac\_del  
rtk\_dot1x\_macBasedDirection\_set  
rtk\_dot1x\_macBasedDirection\_get  
rtk\_dot1x\_guestVlan\_set  
rtk\_dot1x\_guestVlan\_get  
rtk\_dot1x\_guestVlan2Auth\_set  
rtk\_dot1x\_guestVlan2Auth\_get

---

#### 3.1. rtk\_dot1x\_unauthPacketOper\_set

**rtk\_api\_ret\_t rtk\_dot1x\_unauthPacketOper\_set(rtk\_port\_t port,  
rtk\_dot1x\_unauth\_action\_t unauth\_action)**

Set 802.1x unauth action configuration.

---

	Defined in: dot1x.h										
<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>unauth_action</i> 802.1X unauth action.</p>										
<b>Comments</b>	This API can set 802.1x unauth action configuration. The unauth action is as following: <ul style="list-style-type: none"> <li>- DOT1X_ACTION_DROP</li> <li>- DOT1X_ACTION_TRAP2CPU</li> <li>- DOT1X_ACTION_GUESTVLAN</li> </ul>										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameter.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_INPUT	Invalid input parameter.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_PORT_ID	Invalid port number.										
RT_ERR_INPUT	Invalid input parameter.										

---

### 3.2. rtk\_dot1x\_unauthPacketOper\_get

```
rtk_api_ret_t rtk_dot1x_unauthPacketOper_get(rtk_port_t port,
                                             rtk_dot1x_unauth_action_t *pUnauth_action)
```

Get 802.1x unauth action configuration.

Defined in: dot1x.h

<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>*pUnauth_action</i> 802.1X unauth action.</p>								
<b>Comments</b>	This API can get 802.1x unauth action configuration. The unauth action is as following: <ul style="list-style-type: none"> <li>- DOT1X_ACTION_DROP</li> <li>- DOT1X_ACTION_TRAP2CPU</li> <li>- DOT1X_ACTION_GUESTVLAN</li> </ul>								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

RT_ERR_PORT_ID	Invalid port number.
----------------	----------------------

### 3.3. rtk\_dot1x\_eapolFrame2CpuEnable\_set

**rtk\_api\_ret\_t rtk\_dot1x\_eapolFrame2CpuEnable\_set(rtk\_enable\_t enable)**

Set 802.1x EAPOL packet trap to CPU configuration

Defined in: dot1x.h

**Parameters**

*enable*

The status of 802.1x EAPOL packet.

**Comments**

To support 802.1x authentication functionality, EAPOL frame (ether type = 0x888E) has to be trapped to CPU. The status of EAPOL frame trap to CPU is as following:

- DISABLED
- ENABLED

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_ENABLE

Invalid enable input.

### 3.4. rtk\_dot1x\_eapolFrame2CpuEnable\_get

**rtk\_api\_ret\_t rtk\_dot1x\_eapolFrame2CpuEnable\_get(rtk\_enable\_t \*pEnable)**

Get 802.1x EAPOL packet trap to CPU configuration

Defined in: dot1x.h

**Parameters**

*\*pEnable*

The status of 802.1x EAPOL packet.

**Comments**

To support 802.1x authentication functionality, EAPOL frame (ether type = 0x888E) has to be trapped to CPU. The status of EAPOL frame trap to CPU is as following:

- DISABLED
- ENABLED

**Return Codes**

RT\_ERR\_OK

ok

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	invalid input parameters.

---

### 3.5. rtk\_dot1x\_portBasedEnable\_set

`rtk_api_ret_t rtk_dot1x_portBasedEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set 802.1x port-based enable configuration

Defined in: dot1x.h

#### Parameters

*port*  
Port id.

*enable*  
The status of 802.1x port.

#### Comments

The API can update the port-based port enable register content. If a port is 802.1x port based network access control "enabled", it should be authenticated so packets from that port won't be dropped or trapped to CPU. The status of 802.1x port-based network access control is as following:

- DISABLED
- ENABLED

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_ENABLE	Invalid enable input.
RT_ERR_DOT1X_PORTBASEDPNEN	802.1X port

---

### 3.6. rtk\_dot1x\_portBasedEnable\_get

`rtk_api_ret_t rtk_dot1x_portBasedEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get 802.1x port-based enable configuration

Defined in: dot1x.h

<b>Parameters</b>	<i>port</i> Port id.  <i>*pEnable</i> The status of 802.1x port.										
<b>Comments</b>	The API can get the 802.1x port-based port status.										
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_INPUT	Invalid input parameters.										
RT_ERR_PORT_ID	Invalid port number.										

---

### 3.7. rtk\_dot1x\_portBasedAuthStatus\_set

`rtk_api_ret_t rtk_dot1x_portBasedAuthStatus_set(rtk_port_t port,  
rtk_dot1x_auth_status_t port_auth)`

Set 802.1x port-based auth. port configuration

Defined in: dot1x.h

<b>Parameters</b>	<i>port</i> Port id.  <i>port_auth</i> The status of 802.1x port.										
<b>Comments</b>	The authenticated status of 802.1x port-based network access control is as following: - UNAUTH - AUTH										
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> <tr> <td>RT_ERR_DOT1X_PORTBASEDAUTH</td><td>802.1X port</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_DOT1X_PORTBASEDAUTH	802.1X port
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_PORT_ID	Invalid port number.										
RT_ERR_DOT1X_PORTBASEDAUTH	802.1X port										

---

### 3.8. rtk\_dot1x\_portBasedAuthStatus\_get

`rtk_api_ret_t rtk_dot1x_portBasedAuthStatus_get(rtk_port_t port,  
rtk_dot1x_auth_status_t *pPort_auth)`

Get 802.1x port-based auth. port configuration

Defined in: dot1x.h

**Parameters**

*port*

Port id.

*\*pPort\_auth*

The status of 802.1x port.

**Comments**

The API can get 802.1x port-based port auth.information.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_INPUT

Invalid input parameters.

RT\_ERR\_PORT\_ID

Invalid port number.

---

### 3.9. rtk\_dot1x\_portBasedDirection\_set

`rtk_api_ret_t rtk_dot1x_portBasedDirection_set(rtk_port_t port,  
rtk_dot1x_direction_t port_direction)`

Set 802.1x port-based operational direction configuration

Defined in: dot1x.h

**Parameters**

*port*

Port id.

*port\_direction*

Operation direction

**Comments**

The operate controlled direction of 802.1x port-based network access control is as following:

- BOTH
- IN

**Return Codes**

RT\_ERR\_OK

ok

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_DOT1X_PORTBASEDOPDI R	802.1X port

### 3.10. rtk\_dot1x\_portBasedDirection\_get

`rtk_api_ret_t rtk_dot1x_portBasedDirection_get(rtk_port_t port,  
rtk_dot1x_direction_t *pPort_direction)`

Get 802.1X port-based operational direction configuration

Defined in: dot1x.h

**Parameters**

*port*  
Port id.

*\*pPort\_direction*  
Operation direction

**Comments**

The API can get 802.1x port-based operational direction information.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port number.

### 3.11. rtk\_dot1x\_macBasedEnable\_set

`rtk_api_ret_t rtk_dot1x_macBasedEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set 802.1x mac-based port enable configuration

Defined in: dot1x.h

**Parameters**

*port*  
Port id.

---

*enable*

The status of 802.1x port.

**Comments**

If a port is 802.1x MAC based network access control "enabled", the incoming packets should be authenticated so packets from that port won't be dropped or trapped to CPU. The status of 802.1x MAC-based network access control is as following:

- DISABLED
- ENABLED

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_ENABLE	Invalid enable input.
RT_ERR_DOT1X_MACBASEDPNEN	802.1X mac

---

### 3.12. rtk\_dot1x\_macBasedEnable\_get

`rtk_api_ret_t rtk_dot1x_macBasedEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get 802.1x mac-based port enable configuration

Defined in: dot1x.h

**Parameters**

*port*

Port id.

*\*pEnable*

The status of 802.1x port.

**Comments**

If a port is 802.1x MAC based network access control "enabled", the incoming packets should be authenticated so packets from that port won't be dropped or trapped to CPU. The status of 802.1x MAC-based network access control is as following:

- DISABLED
- ENABLED

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port number.

---

### **3.13. rtk\_dot1x\_macBasedAuthMac\_add**

**rtk\_api\_ret\_t rtk\_dot1x\_macBasedAuthMac\_add(rtk\_port\_t port, rtk\_mac\_t \*pAuth\_mac, rtk\_fid\_t fid)**

Add an authenticated MAC to ASIC

Defined in: dot1x.h

**Parameters**

*port*

Port id.

*\*pAuth\_mac*

The authenticated MAC.

*fid*

filtering database.

**Comments**

The API can add a 802.1x authenticated MAC address to port. If the MAC does not exist in LUT, user can't add this MAC to auth status.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_ENABLE	Invalid enable input.
RT_ERR_DOT1X_MACBASEDPNEN	802.1X mac

---

### **3.14. rtk\_dot1x\_macBasedAuthMac\_del**

**rtk\_api\_ret\_t rtk\_dot1x\_macBasedAuthMac\_del(rtk\_port\_t port, rtk\_mac\_t \*pAuth\_mac, rtk\_fid\_t fid)**

Delete an authenticated MAC to ASIC

Defined in: dot1x.h

**Parameters**

*port*

Port id.

*\*pAuth\_mac*

The authenticated MAC.

---

<i>fid</i>	filtering database.										
<b>Comments</b>	The API can delete a 802.1x authenticated MAC address to port. It only change the auth status of the MAC and won't delete it from LUT.										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_MAC</td><td>Invalid MAC address.</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_MAC	Invalid MAC address.	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_MAC	Invalid MAC address.										
RT_ERR_PORT_ID	Invalid port number.										

---

### 3.15. rtk\_dot1x\_macBasedDirection\_set

`rtk_api_ret_t rtk_dot1x_macBasedDirection_set(rtk_dot1x_direction_t  
mac_direction)`

Set 802.1x mac-based operational direction configuration

Defined in: dot1x.h

*mac\_direction*

Operation direction

**Comments**  
The operate controlled direction of 802.1x mac-based network access control is as following:

- BOTH
- IN

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_DOT1X_MACBASEDOPDIR	802.1X mac

---

### 3.16. rtk\_dot1x\_macBasedDirection\_get

`rtk_api_ret_t rtk_dot1x_macBasedDirection_get(rtk_dot1x_direction_t  
*pMac_direction)`

Get 802.1x mac-based operational direction configuration

Defined in: dot1x.h

**Parameters**

*\*pMac\_direction*  
Port id.

**Comments**

The API can get 802.1x mac-based operational direction information.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

### 3.17. rtk\_dot1x\_guestVlan\_set

*rtk\_api\_ret\_t rtk\_dot1x\_guestVlan\_set(rtk\_vlan\_t vid)*

Set 802.1x mac-based operational direction configuration

Defined in: dot1x.h

**Parameters**

*vid*  
802.1x guest VLAN ID

**Comments**

The operate controlled 802.1x guest VLAN

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.

---

### 3.18. rtk\_dot1x\_guestVlan\_get

*rtk\_api\_ret\_t rtk\_dot1x\_guestVlan\_get(rtk\_vlan\_t \*pVid)*

Get 802.1x guest VLAN configuration

Defined in: dot1x.h

**Parameters**

*\*pVid*  
802.1x guest VLAN ID

---

<b>Comments</b>	The API can get 802.1x guest VLAN information.
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

### 3.19. rtk\_dot1x\_guestVlan2Auth\_set

`rtk_api_ret_t rtk_dot1x_guestVlan2Auth_set(rtk_enable_t enable)`

Set 802.1x guest VLAN to auth host configuration

Defined in: dot1x.h

<b>Parameters</b>	<i>enable</i>	
		The status of guest VLAN to auth host.
<b>Comments</b>	The operational direction of 802.1x guest VLAN to auth host control is as following:	
	- ENABLED	
	- DISABLED	
<b>Return Codes</b>		
RT_ERR_OK	ok	
RT_ERR_FAILED	failed	
RT_ERR_SMI	SMI access error	
RT_ERR_INPUT	Invalid input parameter.	

---

### 3.20. rtk\_dot1x\_guestVlan2Auth\_get

`rtk_api_ret_t rtk_dot1x_guestVlan2Auth_get(rtk_enable_t *pEnable)`

Get 802.1x guest VLAN to auth host configuration

Defined in: dot1x.h

<b>Parameters</b>	<i>*pEnable</i>	
		The status of guest VLAN to auth host.
<b>Comments</b>	The API can get 802.1x guest VLAN to auth host information.	
<b>Return Codes</b>		
RT_ERR_OK	ok	

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 4. Module eee.h - Unmanaged switch high-level API

Filename: eee.h

**Description**

The file includes EEE module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

eee.h - Unmanaged switch high-level API

rtk\_eee\_init

rtk\_eee\_portEnable\_set

rtk\_eee\_portEnable\_get

---

### 4.1. rtk\_eee\_init

**rtk\_api\_ret\_t rtk\_eee\_init( void )**

EEE function initialization.

Defined in: eee.h

**Parameters**

*void*

**Comments**

This API is used to initialize EEE status.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

---

## 4.2. rtk\_eee\_portEnable\_set

`rtk_api_ret_t rtk_eee_portEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set enable status of EEE function.

Defined in: eee.h

**Parameters**

*port*

port id.

*enable*

enable EEE status.

**Comments**

This API can set EEE function to the specific port. The configuration of the port is as following:

- DISABLE
- ENABLE

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_ENABLE

Invalid enable input.

---

## 4.3. rtk\_eee\_portEnable\_get

`rtk_api_ret_t rtk_eee_portEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get port admin configuration of the specific port.

Defined in: eee.h

**Parameters**

*port*

Port id.

*\*pEnable*

Back pressure status.

**Comments**

This API can set EEE function to the specific port. The configuration of the port is as following:

- DISABLE
- ENABLE

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID	ok failed SMI access error Invalid port number.
---------------------	--	--

---

## 5. Module igmp.h - Unmanaged switch high-level API

Filename: igmp.h

**Description** The file includes IGMP module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

igmp.h - Unmanaged switch high-level API  
 rtk\_igmp\_init  
 rtk\_igmp\_state\_set  
 rtk\_igmp\_state\_get  
 rtk\_igmp\_static\_router\_port\_set  
 rtk\_igmp\_static\_router\_port\_get  
 rtk\_igmp\_protocol\_set  
 rtk\_igmp\_protocol\_get  
 rtk\_igmp\_fastLeave\_set  
 rtk\_igmp\_fastLeave\_get  
 rtk\_igmp\_maxGroup\_set  
 rtk\_igmp\_maxGroup\_get  
 rtk\_igmp\_currentGroup\_get  
 rtk\_igmp\_tableFullAction\_set  
 rtk\_igmp\_tableFullAction\_get  
 rtk\_igmp\_checksumErrorAction\_set  
 rtk\_igmp\_checksumFrrorAction\_get  
 rtk\_igmp\_leaveTimer\_set  
 rtk\_igmp\_leaveTimer\_get  
 rtk\_igmp\_queryInterval\_set  
 rtk\_igmp\_queryInterval\_get  
 rtk\_igmp\_robustness\_set  
 rtk\_igmp\_robustness\_get  
 rtk\_igmp\_dynamicRouterPortAllow\_set

---

```
rtk_igmp_dynamicRouterPortAllow_get  
rtk_igmp_dynamicRouterPort_get  
rtk_igmp_suppressionEnable_set  
rtk_igmp_suppressionEnable_get  
rtk_igmp_portRxPktEnable_set  
rtk_igmp_portRxPktEnable_get  
rtk_igmp_groupInfo_get  
rtk_igmp_ReportLeaveFwdAction_set  
rtk_igmp_ReportLeaveFwdAction_get  
rtk_igmp_dropLeaveZeroEnable_set  
rtk_igmp_dropLeaveZeroEnable_get  
rtk_igmp_bypassGroupRange_set  
rtk_igmp_bypassGroupRange_get
```

---

## 5.1. rtk\_igmp\_init

`rtk_api_ret_t rtk_igmp_init( void )`

This API enables H/W IGMP and set a default initial configuration.

Defined in: igmp.h

**Parameters**

`void`

**Comments**

This API enables H/W IGMP and set a default initial configuration.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 5.2. rtk\_igmp\_state\_set

`rtk_api_ret_t rtk_igmp_state_set(rtk_enable_t enabled)`

This API set H/W IGMP state.

Defined in: igmp.h

**Parameters**

<code>enabled</code>	
	H/W IGMP state

<b>Comments</b>	This API set H/W IGMP state.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error parameter

---

### 5.3. rtk\_igmp\_state\_get

`rtk_api_ret_t rtk_igmp_state_get(rtk_enable_t *pEnabled)`

This API get H/W IGMP state.

Defined in: igmp.h

<b>Parameters</b>	<code>*pEnabled</code>
	H/W IGMP state

**Comments** This API set current H/W IGMP state.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error parameter

---

### 5.4. rtk\_igmp\_static\_router\_port\_set

`rtk_api_ret_t rtk_igmp_static_router_port_set(rtk_portmask_t *pPortmask)`

Configure static router port

Defined in: igmp.h

<b>Parameters</b>	<code>*pPortmask</code>
	Static Port mask

**Comments** This API set static router port

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

RT_ERR_PORT_MASK	Error parameter
------------------	-----------------

---

## 5.5. rtk\_igmp\_static\_router\_port\_get

`rtk_api_ret_t rtk_igmp_static_router_port_get(rtk_portmask_t *pPortmask)`

Get static router port

Defined in: igmp.h

**Parameters** *\*pPortmask*  
Static port mask

**Comments** This API get static router port

**Return Codes** RT\_ERR\_OK ok  
RT\_ERR\_FAILED failed  
RT\_ERR\_SMI SMI access error  
RT\_ERR\_PORT\_MASK Error parameter

---

## 5.6. rtk\_igmp\_protocol\_set

`rtk_api_ret_t rtk_igmp_protocol_set(rtk_port_t port, rtk_igmp_protocol_t protocol, rtk_igmp_action_t action)`

set IGMP/MLD protocol action

Defined in: igmp.h

**Parameters** *port*  
Port ID  
*protocol*  
IGMP/MLD protocol  
*action*  
Per

**Comments** This API set IGMP/MLD protocol action

**Return Codes** RT\_ERR\_OK ok  
RT\_ERR\_FAILED failed  
RT\_ERR\_SMI SMI access error

RT\_ERR\_PORT\_MASK Error parameter

---

## 5.7. rtk\_igmp\_protocol\_get

`rtk_api_ret_t rtk_igmp_protocol_get(rtk_port_t port, rtk_igmp_protocol_t protocol, rtk_igmp_action_t *pAction)`

set IGMP/MLD protocol action

Defined in: igmp.h

### Parameters

*port*  
Port ID

*protocol*  
IGMP/MLD protocol

*\*pAction*  
Per

### Comments

This API set IGMP/MLD protocol action

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	Error parameter

---

## 5.8. rtk\_igmp\_fastLeave\_set

`rtk_api_ret_t rtk_igmp_fastLeave_set(rtk_enable_t state)`

set IGMP/MLD FastLeave state

Defined in: igmp.h

### Parameters

*state*  
ENABLED: Enable FastLeave, DISABLED: disable FastLeave

### Comments

This API set IGMP/MLD FastLeave state

### Return Codes

RT_ERR_OK	ok
RT_ERR_INPUT	Error Input
RT_ERR_FAILED	failed

---

RT_ERR_SMI	SMI access error
------------	------------------

---

## 5.9. rtk\_igmp\_fastLeave\_get

`rtk_api_ret_t rtk_igmp_fastLeave_get(rtk_enable_t *pState)`

get IGMP/MLD FastLeave state

Defined in: igmp.h

<b>Parameters</b>	<i>*pState</i>	
		ENABLED: Enable FastLeave, DISABLED: disable FastLeave
<b>Comments</b>	This API get IGMP/MLD FastLeave state	
<b>Return Codes</b>		
	RT_ERR_OK	ok
	RT_ERR_NULL_POINTER	NULL pointer
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 5.10. rtk\_igmp\_maxGroup\_set

`rtk_api_ret_t rtk_igmp_maxGroup_set(rtk_port_t port, rtk_uint32 group)`

Set per port multicast group learning limit.

Defined in: igmp.h

<b>Parameters</b>	<i>port</i>	
	Port ID	
	<i>group</i>	
		The number of multicast group learning limit.
<b>Comments</b>	This API set per port multicast group learning limit.	
<b>Return Codes</b>		
	RT_ERR_OK	ok
	RT_ERR_PORT_ID	Error Port ID
	RT_ERR_OUT_OF_RANGE	parameter out of range
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 5.11.rtk\_igmp\_maxGroup\_get

`rtk_api_ret_t rtk_igmp_maxGroup_get(rtk_port_t port, rtk_uint32 *pGroup)`

Get per port multicast group learning limit.

Defined in: igmp.h

**Parameters**

*port*  
Port ID

*\*pGroup*  
The number of multicast group learning limit.

**Comments**

This API get per port multicast group learning limit.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_PORT_ID</code>	Error Port ID
<code>RT_ERR_NULL_POINTER</code>	Null pointer
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 5.12.rtk\_igmp\_currentGroup\_get

`rtk_api_ret_t rtk_igmp_currentGroup_get(rtk_port_t port, rtk_uint32 *pGroup)`

Get per port multicast group learning count.

Defined in: igmp.h

**Parameters**

*port*  
Port ID

*\*pGroup*  
The number of multicast group learning count.

**Comments**

This API get per port multicast group learning count.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_PORT_ID</code>	Error Port ID
<code>RT_ERR_NULL_POINTER</code>	Null pointer
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 5.13. rtk\_igmp\_tableFullAction\_set

`rtk_api_ret_t rtk_igmp_tableFullAction_set(rtk_igmp_tableFullAction_t action)`

set IGMP/MLD Table Full Action

Defined in: igmp.h

**Parameters**

*action*  
Table Full Action

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_INPUT	Error Input
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 5.14. rtk\_igmp\_tableFullAction\_get

`rtk_api_ret_t rtk_igmp_tableFullAction_get(rtk_igmp_tableFullAction_t *pAction)`

get IGMP/MLD Table Full Action

Defined in: igmp.h

**Parameters**

*\*pAction*  
Table Full Action

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_NULL_POINTER	Null pointer
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 5.15.rtk\_igmp\_checksumErrorAction\_set

**rtk\_api\_ret\_t**  
**rtk\_igmp\_checksumErrorAction\_set(rtk\_igmp\_checksumErrorAction\_t  
action)**

set IGMP/MLD Checksum Error Action

Defined in: igmp.h

**Parameters** *action*  
Checksum error Action

**Comments** This API get per port multicast group learning count.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_INPUT	Error Input
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 5.16.rtk\_igmp\_checksumErrorAction\_get

**rtk\_api\_ret\_t**  
**rtk\_igmp\_checksumErrorAction\_get(rtk\_igmp\_checksumErrorAction\_t  
\*pAction)**

get IGMP/MLD Checksum Error Action

Defined in: igmp.h

**Parameters** *\*pAction*  
Checksum error Action

**Comments** This API get per port multicast group learning count.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_NULL_POINTER	Null pointer
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

---

## 5.17. rtk\_igmp\_leaveTimer\_set

`rtk_api_ret_t rtk_igmp_leaveTimer_set(rtk_uint32 timer)`

set IGMP/MLD Leave timer

Defined in: igmp.h

**Parameters**

*timer*

Leave timer

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_INPUT

Error Input

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 5.18. rtk\_igmp\_leaveTimer\_get

`rtk_api_ret_t rtk_igmp_leaveTimer_get(rtk_uint32 *pTimer)`

get IGMP/MLD Leave timer

Defined in: igmp.h

**Parameters**

*\*pTimer*

Leave Timer.

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_NULL\_POINTER

Null pointer

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 5.19. rtk\_igmp\_queryInterval\_set

`rtk_api_ret_t rtk_igmp_queryInterval_set(rtk_uint32 interval)`

set IGMP/MLD Query Interval

Defined in: igmp.h

**Parameters**

*interval*

Query Interval

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_INPUT

Error Input

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 5.20.rtk\_igmp\_queryInterval\_get

`rtk_api_ret_t rtk_igmp_queryInterval_get(rtk_uint32 *pInterval)`

get IGMP/MLD Query Interval

Defined in: igmp.h

**Parameters**

*\*pInterval*

Query Interval

**Comments**

This API get per port multicast group learning count.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_NULL\_POINTER

Null pointer

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 5.21.rtk\_igmp\_robustness\_set

`rtk_api_ret_t rtk_igmp_robustness_set(rtk_uint32 robustness)`

set IGMP/MLD Robustness value

Defined in: igmp.h

**Parameters**

*robustness*

Robustness value

---

<b>Comments</b>	This API get per port multicast group learning count.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_INPUT	Error Input
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 5.22. rtk\_igmp\_robustness\_get

`rtk_api_ret_t rtk_igmp_robustness_get(rtk_uint32 *pRobustness)`

get IGMP/MLD Robustness value

Defined in: igmp.h

<b>Parameters</b>	<i>*pRobustness</i> Robustness value.	
<b>Comments</b>	This API get per port multicast group learning count.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_NULL_POINTER	Null pointer
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 5.23. rtk\_igmp\_dynamicRouterPortAllow\_set

`rtk_api_ret_t rtk_igmp_dynamicRouterPortAllow_set(rtk_portmask_t *pPortmask)`

Configure dynamic router port allow option

Defined in: igmp.h

<b>Parameters</b>	<i>*pPortmask</i> Dynamic Port allow mask	
<b>Comments</b>		
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed

RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	Error parameter

## 5.24. rtk\_igmp\_dynamicRouterPortAllow\_get

**rtk\_api\_ret\_t rtk\_igmp\_dynamicRouterPortAllow\_get(rtk\_portmask\_t \*pPortmask)**

Get dynamic router port allow option

Defined in: igmp.h

**Parameters**

\**pPortmask*  
Dynamic Port allow mask

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	Error parameter

## 5.25. rtk\_igmp\_dynamicRouterPort\_get

**rtk\_api\_ret\_t  
rtk\_igmp\_dynamicRouterPort\_get(rtk\_igmp\_dynamicRouterPort\_t \*pDynamicRouterPort)**

Get dynamic router port

Defined in: igmp.h

**Parameters**

\**pDynamicRouterPort*  
Dynamic Router Port

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Null pointer
RT_ERR_SMI	SMI access error

---

RT\_ERR\_PORT\_MASK                              Error parameter

---

## 5.26. rtk\_igmp\_suppressionEnable\_set

`rtk_api_ret_t rtk_igmp_suppressionEnable_set(rtk_enable_t  
reportSuppression, rtk_enable_t leaveSuppression)`

Configure IGMPv1/v2 & MLDv1 Report/Leave/Done suppression

Defined in: igmp.h

**Parameters**

*reportSuppression*  
Report suppression  
*leaveSuppression*  
Leave suppression

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error Input

## 5.27. rtk\_igmp\_suppressionEnable\_get

`rtk_api_ret_t rtk_igmp_suppressionEnable_get(rtk_enable_t  
*pReportSuppression, rtk_enable_t *pLeaveSuppression)`

Get IGMPv1/v2 & MLDv1 Report/Leave/Done suppression

Defined in: igmp.h

**Parameters**

*\*pReportSuppression*  
Report suppression  
*\*pLeaveSuppression*  
Leave suppression

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Null pointer

---

## 5.28. rtk\_igmp\_portRxPktEnable\_set

**rtk\_api\_ret\_t rtk\_igmp\_portRxPktEnable\_set(rtk\_port\_t *port*,  
rtk\_igmp\_rxPktEnable\_t \**pRxCfg*)**

Configure IGMP/MLD RX Packet configuration

Defined in: igmp.h

**Parameters**

*port*  
Port ID  
*\*pRxCfg*  
RX Packet Configuration

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error Input
RT_ERR_NULL_POINTER	Null pointer

---

## 5.29. rtk\_igmp\_portRxPktEnable\_get

**rtk\_api\_ret\_t rtk\_igmp\_portRxPktEnable\_get(rtk\_port\_t *port*,  
rtk\_igmp\_rxPktEnable\_t \**pRxCfg*)**

Get IGMP/MLD RX Packet configuration

Defined in: igmp.h

**Parameters**

*port*  
Port ID  
*\*pRxCfg*  
RX Packet Configuration

**Comments**

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input
	RT_ERR_NULL_POINTER	Null pointer

---

### 5.30. rtk\_igmp\_groupInfo\_get

`rtk_api_ret_t rtk_igmp_groupInfo_get(rtk_uint32 index,  
rtk_igmp_groupInfo_t *pGroup)`

Get IGMP/MLD Group database

Defined in: igmp.h

**Parameters**

*index*  
Index (0~255)

*\*pGroup*  
Group database information.

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error Input
RT_ERR_NULL_POINTER	Null pointer

---

### 5.31. rtk\_igmp\_ReportLeaveFwdAction\_set

`rtk_api_ret_t  
rtk_igmp_ReportLeaveFwdAction_set(rtk_igmp_ReportLeaveFwdAct_t  
action)`

Set Report Leave packet forwarding action

Defined in: igmp.h

**Parameters**

*action*  
Action

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Error Input
---------------------	--	---

---

**5.32. rtk\_igmp\_ReportLeaveFwdAction\_get**

`rtk_api_ret_t  
rtk_igmp_ReportLeaveFwdAction_get(rtk_igmp_ReportLeaveFwdAct_t  
*pAction)`

Get Report Leave packet forwarding action

Defined in: igmp.h

**Parameters**

`*pAction`  
Action

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT RT_ERR_NULL_POINTER	ok failed SMI access error Error Input Null Pointer
---------------------	---	---

---

**5.33. rtk\_igmp\_dropLeaveZeroEnable\_set**

`rtk_api_ret_t rtk_igmp_dropLeaveZeroEnable_set(rtk_enable_t enabled)`

Set the function of dropping Leave packet with group IP = 0.0.0.0

Defined in: igmp.h

**Parameters**

`enabled`  
Action 1: drop, 0:pass

**Comments****Return Codes**



RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error Input

---

### 5.36.rtk\_igmp\_bypassGroupRange\_get

`rtk_api_ret_t rtk_igmp_bypassGroupRange_get(rtk_igmp_bypassGroup_t group, rtk_enable_t *pEnable)`

get Bypass group

Defined in: igmp.h

**Parameters**

*group*  
bypassed group

*\*pEnable*  
enabled 1: Bypassed, 0: not bypass

**Comments**

**Return Codes**

RT\_ERR\_OK  
RT\_ERR\_FAILED  
RT\_ERR\_SMI  
RT\_ERR\_INPUT  
RT\_ERR\_NULL\_POINTER

ok  
failed  
SMI access error  
Error Input  
Null Pointer

## 6. Module interrupt.h - Unmanaged switch high-level API

Filename: interrupt.h

**Description**

The file includes Interrupt module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

interrupt.h - Unmanaged switch high-level API

---

```
rtk_int_polarity_set  
rtk_int_polarity_get  
rtk_int_control_set  
rtk_int_control_get  
rtk_int_status_set  
rtk_int_status_get  
rtk_int_advanceInfo_get
```

---

## 6.1. rtk\_int\_polarity\_set

`rtk_api_ret_t rtk_int_polarity_set(rtk_int_polarity_t type)`

Set interrupt polarity configuration.

Defined in: interrupt.h

**Parameters**

*type*  
Interruptpolarity type.

**Comments**

The API can set interrupt polarity configuration.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 6.2. rtk\_int\_polarity\_get

`rtk_api_ret_t rtk_int_polarity_get(rtk_int_polarity_t *pType)`

Get interrupt polarity configuration.

Defined in: interrupt.h

**Parameters**

*\*pType*  
Interruptpolarity type.

**Comments**

The API can get interrupt polarity configuration.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 6.3. rtk\_int\_control\_set

`rtk_api_ret_t rtk_int_control_set(rtk_int_type_t type, rtk_enable_t enable)`

Set interrupt trigger status configuration.

Defined in: interrupt.h

**Parameters**

*type*

Interrupt type.

*enable*

Interrupt status.

**Comments**

The API can set interrupt status configuration. The interrupt trigger status is shown in the following:

- INT\_TYPE\_LINK\_STATUS
- INT\_TYPE\_METER\_EXCEED
- INT\_TYPE\_LEARN\_LIMIT
- INT\_TYPE\_LINK\_SPEED
- INT\_TYPE\_CONGEST
- INT\_TYPE\_GREEN\_FEATURE
- INT\_TYPE\_LOOP\_DETECT
- INT\_TYPE\_8051,
- INT\_TYPE\_CABLE\_DIAG,
- INT\_TYPE\_ACL,
- INT\_TYPE\_SLIENT

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_SMI`

SMI access error

`RT_ERR_INPUT`

Invalid input parameters.

`RT_ERR_ENABLE`

Invalid enable input.

---

## 6.4. rtk\_int\_control\_get

`rtk_api_ret_t rtk_int_control_get(rtk_int_type_t type, rtk_enable_t* pEnable)`

Get interrupt trigger status configuration.

---

	Defined in: interrupt.h								
<b>Parameters</b>	<p><i>type</i> Interrupt type.</p> <p><i>pEnable</i> Interrupt status.</p>								
<b>Comments</b>	The API can get interrupt status configuration. The interrupt trigger status is shown in the following: - INT_TYPE_LINK_STATUS - INT_TYPE_METER_EXCEED - INT_TYPE_LEARN_LIMIT - INT_TYPE_LINK_SPEED - INT_TYPE_CONGEST - INT_TYPE_GREEN_FEATURE - INT_TYPE_LOOP_DETECT - INT_TYPE_8051, - INT_TYPE_CABLE_DIAG, - INT_TYPE_ACL, - INT_TYPE_SLINK								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

---

## 6.5. rtk\_int\_status\_set

`rtk_api_ret_t rtk_int_status_set(rtk_int_status_t *pStatusMask)`

Set interrupt trigger status to clean.

Defined in: interrupt.h

<b>Parameters</b>	<i>*pStatusMask</i> Interrupt status bit mask.
<b>Comments</b>	The API can clean interrupt trigger status when interrupt happened. The interrupt trigger status is shown in the following: - INT_TYPE_LINK_STATUS (value[0] (Bit0)) - INT_TYPE_METER_EXCEED (value[0] (Bit1)) - INT_TYPE_LEARN_LIMIT (value[0] (Bit2)) - INT_TYPE_LINK_SPEED (value[0] (Bit3)) - INT_TYPE_CONGEST (value[0] (Bit4))

- INT\_TYPE\_GREEN\_FEATURE (value[0] (Bit5))
- INT\_TYPE\_LOOP\_DETECT (value[0] (Bit6))
- INT\_TYPE\_8051 (value[0] (Bit7))
- INT\_TYPE\_CABLE\_DIAG (value[0] (Bit8))
- INT\_TYPE\_ACL (value[0] (Bit9))
- INT\_TYPE\_SLIENT (value[0] (Bit11)) The status will be cleared after execute this API.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

## 6.6. rtk\_int\_status\_get

`rtk_api_ret_t rtk_int_status_get(rtk_int_status_t* pStatusMask)`

Get interrupt trigger status.

Defined in: interrupt.h

*pStatusMask*

Interrupt status bit mask.

The API can get interrupt trigger status when interrupt happened. The interrupt trigger status is shown in the following:

- INT\_TYPE\_LINK\_STATUS (value[0] (Bit0))
- INT\_TYPE\_METER\_EXCEED (value[0] (Bit1))
- INT\_TYPE\_LEARN\_LIMIT (value[0] (Bit2))
- INT\_TYPE\_LINK\_SPEED (value[0] (Bit3))
- INT\_TYPE\_CONGEST (value[0] (Bit4))
- INT\_TYPE\_GREEN\_FEATURE (value[0] (Bit5))
- INT\_TYPE\_LOOP\_DETECT (value[0] (Bit6))
- INT\_TYPE\_8051 (value[0] (Bit7))
- INT\_TYPE\_CABLE\_DIAG (value[0] (Bit8))
- INT\_TYPE\_ACL (value[0] (Bit9))
- INT\_TYPE\_SLIENT (value[0] (Bit11))

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

## 6.7. rtk\_int\_advanceInfo\_get

```
rtk_api_ret_t rtk_int_advanceInfo_get(rtk_int_advType_t adv_type,  
rtk_int_info_t* info)
```

Get interrupt advanced information.

Defined in: interrupt.h

**Parameters**

*adv\_type*  
Advanced interrupt type.

*info*  
Information per type.

**Comments**

This API can get advanced information when interrupt happened. The status will be cleared after execute this API.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 7. Module l2.h - Unmanaged switch high-level API

Filename: l2.h

**Description**

The file includes L2 module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

**List of Symbols**

Here is a list of all functions and variables in this module

l2.h - Unmanaged switch high-level API  
rtk\_l2\_init  
rtk\_l2\_addr\_add  
rtk\_l2\_addr\_get  
rtk\_l2\_addr\_next\_get  
rtk\_l2\_addr\_del  
rtk\_l2\_mcastAddr\_add  
rtk\_l2\_mcastAddr\_get

rtk\_l2\_mcastAddr\_next\_get  
rtk\_l2\_mcastAddr\_del  
rtk\_l2\_ipMcastAddr\_add  
rtk\_l2\_ipMcastAddr\_get  
rtk\_l2\_ipMcastAddr\_next\_get  
rtk\_l2\_ipMcastAddr\_del  
rtk\_l2\_ipVidMcastAddr\_add  
rtk\_l2\_ipVidMcastAddr\_get  
rtk\_l2\_ipVidMcastAddr\_next\_get  
rtk\_l2\_ipVidMcastAddr\_del  
rtk\_l2\_unicastAddr\_flush  
rtk\_l2\_table\_clear  
rtk\_l2\_table\_clearStatus\_get  
rtk\_l2\_flushLinkDownPortAddrEnable\_set  
rtk\_l2\_flushLinkDownPortAddrEnable\_get  
rtk\_l2\_agingEnable\_set  
rtk\_l2\_agingEnable\_get  
rtk\_l2\_limitLearningCnt\_set  
rtk\_l2\_limitLearningCnt\_get  
rtk\_l2\_limitSystemLearningCnt\_set  
rtk\_l2\_limitSystemLearningCnt\_get  
rtk\_l2\_limitLearningCntAction\_set  
rtk\_l2\_limitLearningCntAction\_get  
rtk\_l2\_limitSystemLearningCntAction\_set  
rtk\_l2\_limitSystemLearningCntAction\_get  
rtk\_l2\_limitSystemLearningCntPortMask\_set  
rtk\_l2\_limitSystemLearningCntPortMask\_get  
rtk\_l2\_learningCnt\_get  
rtk\_l2\_floodPortMask\_set  
rtk\_l2\_floodPortMask\_get  
rtk\_l2\_localPktPermit\_set  
rtk\_l2\_localPktPermit\_get  
rtk\_l2\_aging\_set  
rtk\_l2\_aging\_get  
rtk\_l2\_ipMcastAddrLookup\_set  
rtk\_l2\_ipMcastAddrLookup\_get  
rtk\_l2\_ipMcastForwardRouterPort\_set  
rtk\_l2\_ipMcastForwardRouterPort\_get  
rtk\_l2\_ipMcastGroupEntry\_add  
rtk\_l2\_ipMcastGroupEntry\_del  
rtk\_l2\_ipMcastGroupEntry\_get  
rtk\_l2\_entry\_get  
rtk\_l2\_lookupHitIsolationAction\_set  
rtk\_l2\_lookupHitIsolationAction\_get

---

## 7.1. rtk\_l2\_init

`rtk_api_ret_t rtk_l2_init( void)`

Initialize l2 module of the specified device.

Defined in: l2.h

**Parameters** `void`

**Comments** Initialize l2 module before calling any l2 APIs.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 7.2. rtk\_l2\_addr\_add

`rtk_api_ret_t rtk_l2_addr_add(rtk_mac_t *pMac, rtk_l2_unicastAddr_t *pL2_data)`

Add LUT unicast entry.

Defined in: l2.h

**Parameters**

<code>*pMac</code>	6 bytes unicast(I/G bit is 0) mac address to be written into LUT.
<code>*pL2_data</code>	Unicast entry parameter

**Comments** If the unicast mac address already existed in LUT, it will update the status of the entry. Otherwise, it will find an empty or asic auto learned entry to write. If all the entries with the same hash value can't be replaced, ASIC will return a `RT_ERR_L2_INDEXTBL_FULL` error.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.
<code>RT_ERR_MAC</code>	Invalid MAC address.
<code>RT_ERR_L2_FID</code>	Invalid FID .

RT_ERR_L2_INDEXTBL_FULL	hashed index is full of entries.
RT_ERR_INPUT	Invalid input parameters.

---

### 7.3. rtk\_l2\_addr\_get

**rtk\_api\_ret\_t rtk\_l2\_addr\_get(rtk\_mac\_t \*pMac, rtk\_l2\_unicastAddr\_t \*pL2\_data)**

Get LUT unicast entry.

Defined in: l2.h

#### Parameters

\**pMac*  
6 bytes unicast(I/G bit is 0) mac address to be written into LUT.  
\**pL2\_data*  
Unicast entry parameter

#### Comments

If the unicast mac address existed in LUT, it will return the port and fid where the mac is learned. Otherwise, it will return a RT\_ERR\_L2\_ENTRY\_NOTFOUND error.

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_MAC	Invalid MAC address.
RT_ERR_L2_FID	Invalid FID .
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
RT_ERR_INPUT	Invalid input parameters.

### 7.4. rtk\_l2\_addr\_next\_get

**rtk\_api\_ret\_t rtk\_l2\_addr\_next\_get(rtk\_l2\_read\_method\_t read\_method, rtk\_port\_t port, rtk\_uint32 \*pAddress, rtk\_l2\_unicastAddr\_t \*pL2\_data)**

Get Next LUT unicast entry.

Defined in: l2.h

#### Parameters

---

<i>read_method</i>	The reading method.																
<i>port</i>	The port number if the read_method is READMETHOD_NEXT_L2UCSPA																
<i>*pAddress</i>	The Address ID																
<i>*pL2_data</i>	Unicast entry parameter																
<b>Comments</b>	Get the next unicast entry after the current entry pointed by pAddress. The address of next entry is returned by pAddress. User can use (address + 1) as pAddress to call this API again for dumping all entries in LUT.																
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> <tr> <td>RT_ERR_MAC</td><td>Invalid MAC address.</td></tr> <tr> <td>RT_ERR_L2_FID</td><td>Invalid FID .</td></tr> <tr> <td>RT_ERR_L2_ENTRY_NOTFOUND</td><td>No such LUT entry.</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_MAC	Invalid MAC address.	RT_ERR_L2_FID	Invalid FID .	RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok																
RT_ERR_FAILED	failed																
RT_ERR_SMI	SMI access error																
RT_ERR_PORT_ID	Invalid port number.																
RT_ERR_MAC	Invalid MAC address.																
RT_ERR_L2_FID	Invalid FID .																
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.																
RT_ERR_INPUT	Invalid input parameters.																

---

## 7.5. rtk\_l2\_addr\_del

**rtk\_api\_ret\_t rtk\_l2\_addr\_del(rtk\_mac\_t \*pMac, rtk\_l2\_unicastAddr\_t \*pL2\_data)**

Delete LUT unicast entry.

Defined in: l2.h

<b>Parameters</b>	<i>*pMac</i> 6 bytes unicast(I/G bit is 0) mac address to be written into LUT.  <i>*pL2_data</i> Filtering database
-------------------	---

**Comments** If the mac has existed in the LUT, it will be deleted. Otherwise, it will return RT\_ERR\_L2\_ENTRY\_NOTFOUND.

<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error
RT_ERR_OK	ok						
RT_ERR_FAILED	failed						
RT_ERR_SMI	SMI access error						

RT_ERR_PORT_ID	Invalid port number.
RT_ERR_MAC	Invalid MAC address.
RT_ERR_L2_FID	Invalid FID .
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
RT_ERR_INPUT	Invalid input parameters.

## 7.6. rtk\_l2\_mcastAddr\_add

`rtk_api_ret_t rtk_l2_mcastAddr_add(rtk_l2_mcastAddr_t *pMcastAddr)`

Add LUT multicast entry.

Defined in: l2.h

### Parameters

`*pMcastAddr`  
L2 multicast entry structure

### Comments

If the multicast mac address already existed in the LUT, it will update the port mask of the entry. Otherwise, it will find an empty or asic auto learned entry to write. If all the entries with the same hash value can't be replaced, ASIC will return a RT\_ERR\_L2\_INDEXTBL\_FULL error.

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_MAC	Invalid MAC address.
RT_ERR_L2_FID	Invalid FID .
RT_ERR_L2_VID	Invalid VID .
RT_ERR_L2_INDEXTBL_FULL	hashed index is full of entries.
RT_ERR_PORT_MASK	Invalid portmask.
RT_ERR_INPUT	Invalid input parameters.

## 7.7. rtk\_l2\_mcastAddr\_get

`rtk_api_ret_t rtk_l2_mcastAddr_get(rtk_l2_mcastAddr_t *pMcastAddr)`

Get LUT multicast entry.

---

	Defined in: l2.h	
<b>Parameters</b>	* <i>pMcastAddr</i> L2 multicast entry structure	
<b>Comments</b>	If the multicast mac address existed in the LUT, it will return the port where the mac is learned. Otherwise, it will return a RT_ERR_L2_ENTRY_NOTFOUND error.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_MAC RT_ERR_L2_FID RT_ERR_L2_VID RT_ERR_L2_ENTRY_NOTFOUND RT_ERR_INPUT	ok failed SMI access error Invalid MAC address. Invalid FID . Invalid VID . No such LUT entry. Invalid input parameters.

---

## 7.8. rtk\_l2\_mcastAddr\_next\_get

```
rtk_api_ret_t rtk_l2_mcastAddr_next_get(rtk_uint32 *pAddress,  
rtk_l2_mcastAddr_t *pMcastAddr)
```

Get Next L2 Multicast entry.

Defined in: l2.h

<b>Parameters</b>	* <i>pAddress</i> The Address ID  * <i>pMcastAddr</i> L2 multicast entry structure	
<b>Comments</b>	Get the next L2 multicast entry after the current entry pointed by pAddress. The address of next entry is returned by pAddress. User can use (address + 1) as pAddress to call this API again for dumping all multicast entries in LUT.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_L2_ENTRY_NOTFOUND RT_ERR_INPUT	ok failed SMI access error No such LUT entry. Invalid input parameters.

---

## 7.9. rtk\_l2\_mcastAddr\_del

`rtk_api_ret_t rtk_l2_mcastAddr_del(rtk_l2_mcastAddr_t *pMcastAddr)`

Delete LUT multicast entry.

Defined in: l2.h

**Parameters**

`*pMcastAddr`  
L2 multicast entry structure

**Comments**

If the mac has existed in the LUT, it will be deleted. Otherwise, it will return RT\_ERR\_L2\_ENTRY\_NOTFOUND.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_MAC</code>	Invalid MAC address.
<code>RT_ERR_L2_FID</code>	Invalid FID .
<code>RT_ERR_L2_VID</code>	Invalid VID .
<code>RT_ERR_L2_ENTRY_NOTFOUND</code>	No such LUT entry.
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 7.10. rtk\_l2\_ipMcastAddr\_add

`rtk_api_ret_t rtk_l2_ipMcastAddr_add(rtk_l2_ipMcastAddr_t  
*pIpMcastAddr)`

Add Lut IP multicast entry

Defined in: l2.h

**Parameters**

`*pIpMcastAddr`  
IP Multicast entry

**Comments**

System supports L2 entry with IP multicast DIP/SIP to forward IP multicasting frame as user desired. If this function is enabled, then system will be looked up L2 IP multicast entry to forward IP multicast frame directly without flooding.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

RT_ERR_PORT_ID	Invalid port number.
RT_ERR_L2_INDEXTBL_FULL	hashed index is full of entries.
RT_ERR_PORT_MASK	invalid portmask.
RT_ERR_INPUT	Invalid input parameters.

---

## 7.11. rtk\_l2\_ipMcastAddr\_get

`rtk_api_ret_t rtk_l2_ipMcastAddr_get(rtk_l2_ipMcastAddr_t *pIpMcastAddr)`

Get LUT IP multicast entry.

Defined in: l2.h

**Parameters**

`*pIpMcastAddr`  
IP Multicast entry

**Comments**

The API can get Lut table of IP multicast entry.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
RT_ERR_INPUT	Invalid input parameters.

## 7.12. rtk\_l2\_ipMcastAddr\_next\_get

`rtk_api_ret_t rtk_l2_ipMcastAddr_next_get(rtk_uint32 *pAddress, rtk_l2_ipMcastAddr_t *pIpMcastAddr)`

Get Next IP Multicast entry.

Defined in: l2.h

**Parameters**

`*pAddress`  
The Address ID  
`*pIpMcastAddr`  
IP Multicast entry

**Comments**

Get the next IP multicast entry after the current entry pointed by pAddress. The address of next entry is returned by pAddress. User can use (address + 1) as pAddress to call this API again for dumping all IP multicast entries in LUT.

Return Codes	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
RT_ERR_INPUT	Invalid input parameters.

---

## 7.13.rtk\_l2\_ipMcastAddr\_del

```
rtk_api_ret_t rtk_l2_ipMcastAddr_del(rtk_l2_ipMcastAddr_t  
*pIpMcastAddr)
```

Delete a ip multicast address entry from the specified device.

Defined in: l2.h

Parameters	
*pIpMcastAddr	IP Multicast entry

Comments

The API can delete a IP multicast address entry from the specified device.

Return Codes	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
RT_ERR_INPUT	Invalid input parameters.

---

## 7.14.rtk\_l2\_ipVidMcastAddr\_add

```
rtk_api_ret_t rtk_l2_ipVidMcastAddr_add(rtk_l2_ipVidMcastAddr_t  
*pIpVidMcastAddr)
```

Add Lut IP multicast+VID entry

Defined in: l2.h

Parameters

---

*\*pIpVidMcastAddr*  
IP & VID multicast Entry

**Comments**

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.
	RT_ERR_L2_INDEXTBL_FULL	hashed index is full of entries.
	RT_ERR_PORT_MASK	Invalid portmask
	RT_ERR_INPUT	Invalid input parameters.

---

## 7.15. rtk\_l2\_ipVidMcastAddr\_get

`rtk_api_ret_t rtk_l2_ipVidMcastAddr_get(rtk_l2_ipVidMcastAddr_t  
*pIpVidMcastAddr)`

Get LUT IP multicast+VID entry.

Defined in: l2.h

<b>Parameters</b>	<i>*pIpVidMcastAddr</i> IP & VID multicast Entry
-------------------	---

**Comments**

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.
	RT_ERR_INPUT	Invalid input parameters.

---

## 7.16. rtk\_l2\_ipVidMcastAddr\_next\_get

`rtk_api_ret_t rtk_l2_ipVidMcastAddr_next_get(rtk_uint32 *pAddress,  
rtk_l2_ipVidMcastAddr_t *pIpVidMcastAddr)`

Get Next IP Multicast+VID entry.

	Defined in: l2.h										
<b>Parameters</b>	<p><i>*pAddress</i> The Address ID</p> <p><i>*pIpVidMcastAddr</i> IP &amp; VID multicast Entry</p>										
<b>Comments</b>	Get the next IP multicast entry after the current entry pointed by pAddress. The address of next entry is returned by pAddress. User can use (address + 1) as pAddress to call this API again for dumping all IP multicast entries in LUT.										
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_L2_ENTRY_NOTFOUND</td> <td>No such LUT entry.</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.										
RT_ERR_INPUT	Invalid input parameters.										

---

## 7.17.rtk\_l2\_ipVidMcastAddr\_del

**rtk\_api\_ret\_t rtk\_l2\_ipVidMcastAddr\_del(rtk\_l2\_ipVidMcastAddr\_t \*pIpVidMcastAddr)**

Delete a ip multicast+VID address entry from the specified device.

	Defined in: l2.h										
<b>Parameters</b>	<p><i>*pIpVidMcastAddr</i> IP &amp; VID multicast Entry</p>										
<b>Comments</b>											
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_L2_ENTRY_NOTFOUND</td> <td>No such LUT entry.</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_L2_ENTRY_NOTFOUND	No such LUT entry.										
RT_ERR_INPUT	Invalid input parameters.										

---

## 7.18.rtk\_l2\_icastAddr\_flush

**rtk\_api\_ret\_t rtk\_l2\_icastAddr\_flush(rtk\_l2\_flushCfg\_t \*pConfig)**

---

	Flush L2 mac address by type in the specified device (both dynamic and static).	
Defined in:	l2.h	
<b>Parameters</b>	<p><i>*pConfig</i> flush configuration</p>	
<b>Comments</b>	<p>flushByVid - 1: Flush by VID, 0: Don't flush by VID vid - VID (0 ~ 4095)</p> <p>flushByFid - 1: Flush by FID, 0: Don't flush by FID fid - FID (0 ~ 15)</p> <p>flushByPort - 1: Flush by Port, 0: port - Don't flush by Port</p> <p>flushStaticAddr - Not Supported</p> <p>flushBoth - Flush both Static and Dynamic entries, 0: Flush only Dynamic entries</p> <p>flushAddrOnAllPorts - 1: Flush VID-matched entries at all ports, 0: Flush VID-matched entries per port.</p>	
<b>Return Codes</b>	<p>RT_ERR_OK ok</p> <p>RT_ERR_FAILED failed</p> <p>RT_ERR_SMI SMI access error</p> <p>RT_ERR_PORT_ID Invalid port number.</p> <p>RT_ERR_VLAN_VID Invalid VID parameter.</p> <p>RT_ERR_INPUT Invalid input parameters.</p>	

---

## 7.19. rtk\_l2\_table\_clear

`rtk_api_ret_t rtk_l2_table_clear( void )`

Flush all static & dynamic entries in LUT.

Defined in: l2.h

<b>Parameters</b>	<i>void</i>						
<b>Comments</b>							
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error
RT_ERR_OK	ok						
RT_ERR_FAILED	failed						
RT_ERR_SMI	SMI access error						

---

## 7.20.rtk\_l2\_table\_clearStatus\_get

`rtk_api_ret_t rtk_l2_table_clearStatus_get(rtk_l2_clearStatus_t *pStatus)`

Get table clear status

Defined in: l2.h

**Parameters**

*\*pStatus*  
Clear status, 1:Busy, 0:finish

**Comments**

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 7.21.rtk\_l2\_flushLinkDownPortAddrEnable\_set

`rtk_api_ret_t rtk_l2_flushLinkDownPortAddrEnable_set(rtk_port_t port,  
rtk_enable_t enable)`

Set HW flush linkdown port mac configuration of the specified device.

Defined in: l2.h

**Parameters**

*port*  
Port id.  
*enable*  
link down flush status

**Comments**

The status of flush linkdown port address is as following:

- DISABLED
- ENABLED

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.
<code>RT_ERR_ENABLE</code>	Invalid enable input.

---

---

## 7.22. rtk\_l2\_flushLinkDownPortAddrEnable\_get

`rtk_api_ret_t rtk_l2_flushLinkDownPortAddrEnable_get(rtk_port_t port,  
                  rtk_enable_t *pEnable)`

Get HW flush linkdown port mac configuration of the specified device.

Defined in: l2.h

**Parameters**

*port*

Port id.

*\*pEnable*

link down flush status

**Comments**

The status of flush linkdown port address is as following:

- DISABLED
- ENABLED

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

---

## 7.23. rtk\_l2\_agingEnable\_set

`rtk_api_ret_t rtk_l2_agingEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set L2 LUT aging status per port setting.

Defined in: l2.h

**Parameters**

*port*

Port id.

*enable*

Aging status

**Comments**

This API can be used to set L2 LUT aging status per port.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT_ERR_ENABLE	Invalid enable input.
---------------	-----------------------

## 7.24. rtk\_l2\_agingEnable\_get

`rtk_api_ret_t rtk_l2_agingEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get L2 LUT aging status per port setting.

Defined in: l2.h

**Parameters**

*port*  
Port id.

*\*pEnable*  
Aging status

**Comments**

This API can be used to get L2 LUT aging function per port.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

## 7.25. rtk\_l2\_limitLearningCnt\_set

`rtk_api_ret_t rtk_l2_limitLearningCnt_set(rtk_port_t port, rtk_mac_cnt_t mac_cnt)`

Set per-Port auto learning limit number

Defined in: l2.h

**Parameters**

*port*  
Port id.

*mac\_cnt*  
Auto learning entries limit number

**Comments**

The API can set per-port ASIC auto learning limit number from 0(disable learning) to 8k.

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	invalid port number.
RT_ERR_LIMITED_L2ENTRY_NUM	Invalid auto learning limit number

---

## 7.26. rtk\_l2\_limitLearningCnt\_get

`rtk_api_ret_t rtk_l2_limitLearningCnt_get(rtк_port_t port, rtк_mac_cnt_t *pMac_cnt)`

Get per-Port auto learning limit number

Defined in: l2.h

**Parameters**

*port*  
Port id.

*\*pMac\_cnt*  
Auto learning entries limit number

**Comments**

The API can get per-port ASIC auto learning limit number.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

---

## 7.27. rtk\_l2\_limitSystemLearningCnt\_set

`rtk_api_ret_t rtk_l2_limitSystemLearningCnt_set(rtк_mac_cnt_t mac_cnt)`

Set System auto learning limit number

Defined in: l2.h

**Parameters**

*mac\_cnt*  
Auto learning entries limit number

**Comments**

The API can set system ASIC auto learning limit number from 0(disable learning) to 2112.

**Return Codes**

RT_ERR_OK	ok
-----------	----

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_LIMITED_L2ENTRY_NUM	Invalid auto learning limit number

---

## 7.28.rtk\_l2\_limitSystemLearningCnt\_get

`rtk_api_ret_t rtk_l2_limitSystemLearningCnt_get(rtk_mac_cnt_t *pMac_cnt)`

Get System auto learning limit number

Defined in: l2.h

**Parameters**

`*pMac_cnt`  
Auto learning entries limit number

**Comments**

The API can get system ASIC auto learning limit number.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

---

## 7.29.rtk\_l2\_limitLearningCntAction\_set

`rtk_api_ret_t rtk_l2_limitLearningCntAction_set(rtk_port_t port, rtk_l2_limitLearnCntAction_t action)`

Configure auto learn over limit number action.

Defined in: l2.h

**Parameters**

`port`  
Port id.  
`action`  
Auto learning entries limit number

**Comments**

The API can set SA unknown packet action while auto learn limit number is over  
The action symbol as following:  
- LIMIT\_LEARN\_CNT\_ACTION\_DROP,

---

	- LIMIT_LEARN_CNT_ACTION_FORWARD, - LIMIT_LEARN_CNT_ACTION_TO_CPU,
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port number. RT_ERR_NOT_ALLOWED Invalid learn over action

---

### 7.30. rtk\_l2\_limitLearningCntAction\_get

`rtk_api_ret_t rtk_l2_limitLearningCntAction_get(rtk_port_t port,  
rtk_l2_limitLearnCntAction_t *pAction)`

Get auto learn over limit number action.

Defined in: l2.h

#### Parameters

*port*  
Port id.

*\*pAction*  
Learn over action

#### Comments

The API can get SA unknown packet action while auto learn limit number is over  
The action symbol as following:

- LIMIT\_LEARN\_CNT\_ACTION\_DROP,
- LIMIT\_LEARN\_CNT\_ACTION\_FORWARD,
- LIMIT\_LEARN\_CNT\_ACTION\_TO\_CPU,

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

### 7.31. rtk\_l2\_limitSystemLearningCntAction\_set

`rtk_api_ret_t  
rtk_l2_limitSystemLearningCntAction_set(rtk_l2_limitLearnCntAction_t  
action)`

Configure system auto learn over limit number action.

Defined in: l2.h

**Parameters**

*action*

Port id.

**Comments**

The API can set SA unknown packet action while auto learn limit number is over  
The action symbol as following:

- LIMIT\_LEARN\_CNT\_ACTION\_DROP,
- LIMIT\_LEARN\_CNT\_ACTION\_FORWARD,
- LIMIT\_LEARN\_CNT\_ACTION\_TO\_CPU,

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_NOT\_ALLOWED

Invalid learn over action

---

## 7.32. rtk\_l2\_limitSystemLearningCntAction\_get

**rtk\_api\_ret\_t**  
**rtk\_l2\_limitSystemLearningCntAction\_get(rtk\_l2\_limitLearnCntAction\_t**  
*\*pAction*)

Get system auto learn over limit number action.

Defined in: l2.h

**Parameters**

*\*pAction*

Learn over action

**Comments**

The API can get SA unknown packet action while auto learn limit number is over  
The action symbol as following:

- LIMIT\_LEARN\_CNT\_ACTION\_DROP,
- LIMIT\_LEARN\_CNT\_ACTION\_FORWARD,
- LIMIT\_LEARN\_CNT\_ACTION\_TO\_CPU,

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

---

## 7.33. rtk\_l2\_limitSystemLearningCntPortMask\_set

`rtk_api_ret_t rtk_l2_limitSystemLearningCntPortMask_set(rtk_portmask_t *pPortmask)`

Configure system auto learn portmask

Defined in: l2.h

**Parameters**

`*pPortmask`  
Port Mask

**Comments**

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_MASK</code>	Invalid port mask.

---

## 7.34. rtk\_l2\_limitSystemLearningCntPortMask\_get

`rtk_api_ret_t rtk_l2_limitSystemLearningCntPortMask_get(rtk_portmask_t *pPortmask)`

get system auto learn portmask

Defined in: l2.h

**Parameters**

`*pPortmask`  
Port Mask

**Comments**

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_NULL_POINTER</code>	Null pointer.

---

## 7.35.rtk\_l2\_learningCnt\_get

`rtk_api_ret_t rtk_l2_learningCnt_get(rtk_port_t port, rtk_mac_cnt_t *pMac_cnt)`

Get per-Port current auto learning number

Defined in: l2.h

**Parameters**

*port*  
Port id.

*\*pMac\_cnt*  
ASIC auto learning entries number

**Comments**

The API can get per-port ASIC auto learning number

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.

---

## 7.36.rtk\_l2\_floodPortMask\_set

`rtk_api_ret_t rtk_l2_floodPortMask_set(rtk_l2_flood_type_t floood_type, rtk_portmask_t *pFlood_portmask)`

Set flooding portmask

Defined in: l2.h

**Parameters**

*floood\_type*  
flooding type.  
*\*pFlood\_portmask*  
flooding porkmask

**Comments**

This API can set the flooding mask. The flooding type is as following:

- FLOOD\_UNKNOWNDA
- FLOOD\_UNKNOWNMIC
- FLOOD\_BC

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed

---

RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	Invalid portmask.
RT_ERR_INPUT	invalid input parameters.

---

## 7.37. rtk\_l2\_floodPortMask\_get

`rtk_api_ret_t rtk_l2_floodPortMask_get(rtk_l2_flood_type_t floood_type,  
rtk_portmask_t *pFlood_portmask)`

Get flooding portmask

Defined in: l2.h

**Parameters**

*flood\_type*  
flooding type.

*\*pFlood\_portmask*  
flooding portmask

**Comments**

This API can get the flooding mask. The flooding type is as following:

- FLOOD\_UNKNOWNDA
- FLOOD\_UNKNOWNMIC
- FLOOD\_BC

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

## 7.38. rtk\_l2\_localPktPermit\_set

`rtk_api_ret_t rtk_l2_localPktPermit_set(rtk_port_t port, rtk_enable_t permit)`

Set permission of frames if source port and destination port are the same.

Defined in: l2.h

**Parameters**

*port*  
Port id.

*permit*  
permission status

<b>Comments</b>	This API is setted to permit frame if its source port is equal to destination port.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.
	RT_ERR_ENABLE	Invalid permit value.

---

## 7.39.rtk\_l2\_localPktPermit\_get

`rtk_api_ret_t rtk_l2_localPktPermit_get(rtk_port_t port, rtk_enable_t *pPermit)`

Get permission of frames if source port and destination port are the same.

Defined in: l2.h

<b>Parameters</b>	<i>port</i>	Port id.
	<i>*pPermit</i>	permittion status
<b>Comments</b>	This API is to get permittion status for frames if its source port is equal to destination port.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.

---

## 7.40.rtk\_l2\_aging\_set

`rtk_api_ret_t rtk_l2_aging_set(rtk_l2_age_time_t aging_time)`

Set LUT agging out speed

Defined in: l2.h

<b>Parameters</b>	<i>aging_time</i>	Agging out time.
-------------------	-------------------	------------------

---

<b>Comments</b>	The API can set LUT agging out period for each entry and the range is from 14s to 800s.								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_OUT_OF_RANGE</td> <td>input out of range.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_OUT_OF_RANGE	input out of range.								

---

## 7.41. rtk\_l2\_aging\_get

`rtk_api_ret_t rtk_l2_aging_get(rtk_l2_age_time_t *pAging_time)`

Get LUT agging out time

Defined in: l2.h

<b>Parameters</b>	<code>*pAging_time</code> Aging status
-------------------	---

**Comments** The API can get LUT agging out period for each entry.

<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port number.								

---

## 7.42. rtk\_l2\_ipMcastAddrLookup\_set

`rtk_api_ret_t rtk_l2_ipMcastAddrLookup_set(rtk_l2_ipmc_lookup_type_t type)`

Set Lut IP multicast lookup function

Defined in: l2.h

<b>Parameters</b>	<code>type</code> Lookup type for IPMC packet.
-------------------	---

**Comments** This API can work with rtk\_l2\_ipMcastAddrLookupException\_add. If users set the lookup type to DIP, the group in exception table will be lookup by DIP+SIP If

users set the lookup type to DIP+SIP, the group in exception table will be lookup by only DIP

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 7.43.rtk\_l2\_ipMcastAddrLookup\_get

`rtk_api_ret_t rtk_l2_ipMcastAddrLookup_get(rtk_l2_ipmc_lookup_type_t *pType)`

Get Lut IP multicast lookup function

Defined in: l2.h

<b>Parameters</b>	<i>*pType</i>	Lookup type for IPMC packet.
-------------------	---------------	------------------------------

**Comments** None.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 7.44.rtk\_l2\_ipMcastForwardRouterPort\_set

`rtk_api_ret_t rtk_l2_ipMcastForwardRouterPort_set(rtk_enable_t enabled)`

Set IPMC packet forward to router port also or not

Defined in: l2.h

<b>Parameters</b>	<i>enabled</i>	1: Include router port, 0, exclude router port
-------------------	----------------	--

**Comments**

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

---

## 7.45. rtk\_l2\_ipMcastForwardRouterPort\_get

`rtk_api_ret_t rtk_l2_ipMcastForwardRouterPort_get(rtk_enable_t *pEnabled)`

Get IPMC packet forward to router port also or not

Defined in: l2.h

**Parameters**

`*pEnabled`  
1: Include router port, 0, exclude router port

**Comments**

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_NULL_POINTER</code>	Null pointer

---

## 7.46. rtk\_l2\_ipMcastGroupEntry\_add

`rtk_api_ret_t rtk_l2_ipMcastGroupEntry_add(ipaddr_t ip_addr, rtk_uint32 vid, rtk_portmask_t *pPortmask)`

Add an IP Multicast entry to group table

Defined in: l2.h

**Parameters**

`ip_addr`  
IP address

`vid`  
VLAN ID

`*pPortmask`  
portmask

**Comments**

Add an entry to IP Multicast Group table.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_TBL_FULL</code>	Table Full

---

## 7.47.rtk\_l2\_ipMcastGroupEntry\_del

`rtk_api_ret_t rtk_l2_ipMcastGroupEntry_del(ipaddr_t ip_addr, rtk_uint32 vid)`

Delete an entry from IP Multicast group table

Defined in: l2.h

**Parameters** *ip\_addr*  
IP address

*vid*  
VLAN ID

**Comments** Delete an entry from IP Multicast group table.

**Return Codes** RT\_ERR\_OK ok  
RT\_ERR\_FAILED failed  
RT\_ERR\_SMI SMI access error  
RT\_ERR\_TBL\_FULL Table Full

---

## 7.48.rtk\_l2\_ipMcastGroupEntry\_get

`rtk_api_ret_t rtk_l2_ipMcastGroupEntry_get(ipaddr_t ip_addr, rtk_uint32 vid, rtk_portmask_t *pPortmask)`

get an entry from IP Multicast group table

Defined in: l2.h

**Parameters** *ip\_addr*  
IP address  
*vid*  
VLAN ID  
*\*pPortmask*  
member port mask

**Comments** Delete an entry from IP Multicast group table.

**Return Codes** RT\_ERR\_OK ok  
RT\_ERR\_FAILED failed  
RT\_ERR\_SMI SMI access error

## 7.49. rtk\_l2\_entry\_get

`rtk_api_ret_t rtk_l2_entry_get(rtk_l2_addr_table_t *pL2_entry)`

Get LUT unicast entry.

Defined in: l2.h

**Parameters**

`*pL2_entry`  
Index field in the structure.

**Comments**

This API is used to get address by index from 0~2111.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_L2_EMPTY_ENTRY</code>	Empty LUT entry.
<code>RT_ERR_INPUT</code>	Invalid input parameters.

## 7.50. rtk\_l2\_lookupHitIsolationAction\_set

`rtk_api_ret_t  
rtk_l2_lookupHitIsolationAction_set(rtk_l2_lookupHitIsolationAction_t  
action)`

Set action of lookup hit & isolation.

Defined in: l2.h

**Parameters**

`action`  
The action

**Comments**

This API is used to configure the action of packet which is lookup hit in L2 table but the destination port/portmask are not in the port isolation group.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters

---

## 7.51.rtk\_l2\_lookupHitIsolationAction\_get

```
rtk_api_ret_t  
rtk_l2_lookupHitIsolationAction_fet(rtk_l2_lookupHitIsolationAction_t  
*pAction)  
  
get action of lookup hit & isolation.
```

Defined in: l2.h

**Parameters**

*pAction*

The action

**Comments**

This API is used to get the action of packet which is lookup hit in L2 table but the destination port/portmask are not in the port isolation group.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters

---

## 8. Module leaky.h - Unmanaged switch high-level API

Filename: leaky.h

**Description**

The file includes Leaky module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

**List of Symbols**

Here is a list of all functions and variables in this module

leaky.h - Unmanaged switch high-level API  
rtk\_leaky\_vlan\_set  
rtk\_leaky\_vlan\_get  
rtk\_leaky\_portIsolation\_set  
rtk\_leaky\_portIsolation\_get

---

## 8.1. rtk\_leaky\_vlan\_set

`rtk_api_ret_t rtk_leaky_vlan_set(rtk_leaky_type_t type, rtk_enable_t enable)`

Set VLAN leaky.

Defined in: leaky.h

**Parameters**

*type*

Packet type for VLAN leaky.

*enable*

Leaky status.

**Comments**

This API can set VLAN leaky for RMA ,IGMP/MLD, CDP, CSSTP, and LLDP packets. The leaky frame types are as following:

- LEAKY\_BRG\_GROUP,
- LEAKY\_FD\_PAUSE,
- LEAKY\_SP\_MCAST,
- LEAKY\_1X\_PAE,
- LEAKY\_UNDEF\_BRG\_04,
- LEAKY\_UNDEF\_BRG\_05,
- LEAKY\_UNDEF\_BRG\_06,
- LEAKY\_UNDEF\_BRG\_07,
- LEAKY\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- LEAKY\_UNDEF\_BRG\_09,
- LEAKY\_UNDEF\_BRG\_0A,
- LEAKY\_UNDEF\_BRG\_0B,
- LEAKY\_UNDEF\_BRG\_0C,
- LEAKY\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- LEAKY\_8021AB,
- LEAKY\_UNDEF\_BRG\_0F,
- LEAKY\_BRG\_MNGEMENT,
- LEAKY\_UNDEFINED\_11,
- LEAKY\_UNDEFINED\_12,
- LEAKY\_UNDEFINED\_13,
- LEAKY\_UNDEFINED\_14,
- LEAKY\_UNDEFINED\_15,
- LEAKY\_UNDEFINED\_16,
- LEAKY\_UNDEFINED\_17,
- LEAKY\_UNDEFINED\_18,
- LEAKY\_UNDEFINED\_19,
- LEAKY\_UNDEFINED\_1A,
- LEAKY\_UNDEFINED\_1B,

- LEAKY\_UNDEFINED\_1C,  
- LEAKY\_UNDEFINED\_1D,  
- LEAKY\_UNDEFINED\_1E,  
- LEAKY\_UNDEFINED\_1F,  
- LEAKY\_GMRP,  
- LEAKY\_GVRP,  
- LEAKY\_UNDEF\_GARP\_22,  
- LEAKY\_UNDEF\_GARP\_23,  
- LEAKY\_UNDEF\_GARP\_24,  
- LEAKY\_UNDEF\_GARP\_25,  
- LEAKY\_UNDEF\_GARP\_26,  
- LEAKY\_UNDEF\_GARP\_27,  
- LEAKY\_UNDEF\_GARP\_28,  
- LEAKY\_UNDEF\_GARP\_29,  
- LEAKY\_UNDEF\_GARP\_2A,  
- LEAKY\_UNDEF\_GARP\_2B,  
- LEAKY\_UNDEF\_GARP\_2C,  
- LEAKY\_UNDEF\_GARP\_2D,  
- LEAKY\_UNDEF\_GARP\_2E,  
- LEAKY\_UNDEF\_GARP\_2F,  
- LEAKY\_IGMP,  
- LEAKY\_IPMULTICAST.  
- LEAKY\_CDP,  
- LEAKY\_CSSTP,  
- LEAKY\_LLDP.

Return Codes	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_ENABLE	Invalid enable input

## 8.2. rtk\_leaky\_vlan\_get

```
rtk_api_ret_t rtk_leaky_vlan_get(rtk_leaky_type_t type, rtk_enable_t  
*pEnable)
```

Get VLAN leaky.

Defined in: leaky.h

### Parameters

*type*  
Packet type for VLAN leaky.

---

*\*pEnable*

Leaky status.

**Comments**

This API can get VLAN leaky status for RMA ,IGMP/MLD, CDP, CSSTP, and LLDP packets. The leaky frame types are as following:

- LEAKY\_BRG\_GROUP,
- LEAKY\_FD\_PAUSE,
- LEAKY\_SP\_MCAST,
- LEAKY\_1X\_PAE,
- LEAKY\_UNDEF\_BRG\_04,
- LEAKY\_UNDEF\_BRG\_05,
- LEAKY\_UNDEF\_BRG\_06,
- LEAKY\_UNDEF\_BRG\_07,
- LEAKY\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- LEAKY\_UNDEF\_BRG\_09,
- LEAKY\_UNDEF\_BRG\_0A,
- LEAKY\_UNDEF\_BRG\_0B,
- LEAKY\_UNDEF\_BRG\_0C,
- LEAKY\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- LEAKY\_8021AB,
- LEAKY\_UNDEF\_BRG\_0F,
- LEAKY\_BRG\_MNGEMENT,
- LEAKY\_UNDEFINED\_11,
- LEAKY\_UNDEFINED\_12,
- LEAKY\_UNDEFINED\_13,
- LEAKY\_UNDEFINED\_14,
- LEAKY\_UNDEFINED\_15,
- LEAKY\_UNDEFINED\_16,
- LEAKY\_UNDEFINED\_17,
- LEAKY\_UNDEFINED\_18,
- LEAKY\_UNDEFINED\_19,
- LEAKY\_UNDEFINED\_1A,
- LEAKY\_UNDEFINED\_1B,
- LEAKY\_UNDEFINED\_1C,
- LEAKY\_UNDEFINED\_1D,
- LEAKY\_UNDEFINED\_1E,
- LEAKY\_UNDEFINED\_1F,
- LEAKY\_GMRP,
- LEAKY\_GVRP,
- LEAKY\_UNDEF\_GARP\_22,
- LEAKY\_UNDEF\_GARP\_23,
- LEAKY\_UNDEF\_GARP\_24,
- LEAKY\_UNDEF\_GARP\_25,
- LEAKY\_UNDEF\_GARP\_26,
- LEAKY\_UNDEF\_GARP\_27,
- LEAKY\_UNDEF\_GARP\_28,

- LEAKY\_UNDEF\_GARP\_29,
- LEAKY\_UNDEF\_GARP\_2A,
- LEAKY\_UNDEF\_GARP\_2B,
- LEAKY\_UNDEF\_GARP\_2C,
- LEAKY\_UNDEF\_GARP\_2D,
- LEAKY\_UNDEF\_GARP\_2E,
- LEAKY\_UNDEF\_GARP\_2F,
- LEAKY\_IGMP,
- LEAKY\_IPMULTICAST,
- LEAKY\_CDP,
- LEAKY\_CSSTP,
- LEAKY\_LLDP.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

### 8.3. rtk\_leaky\_portIsolation\_set

`rtk_api_ret_t rtk_leaky_portIsolation_set(rtk_leaky_type_t type,  
rtk_enable_t enable)`

Set port isolation leaky.

Defined in: leaky.h

#### Parameters

*type*  
Packet type for port isolation leaky.

*enable*  
Leaky status.

#### Comments

This API can set port isolation leaky for RMA ,IGMP/MLD, CDP, CSSTP, and LLDP packets. The leaky frame types are as following:

- LEAKY\_BRG\_GROUP,
- LEAKY\_FD\_PAUSE,
- LEAKY\_SP\_MCAST,
- LEAKY\_1X\_PAE,
- LEAKY\_UNDEF\_BRG\_04,
- LEAKY\_UNDEF\_BRG\_05,
- LEAKY\_UNDEF\_BRG\_06,
- LEAKY\_UNDEF\_BRG\_07,
- LEAKY\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,

---

- LEAKY\_UNDEF\_BRG\_09,
- LEAKY\_UNDEF\_BRG\_0A,
- LEAKY\_UNDEF\_BRG\_0B,
- LEAKY\_UNDEF\_BRG\_0C,
- LEAKY\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- LEAKY\_8021AB,
- LEAKY\_UNDEF\_BRG\_0F,
- LEAKY\_BRG\_MNGEMENT,
- LEAKY\_UNDEFINED\_11,
- LEAKY\_UNDEFINED\_12,
- LEAKY\_UNDEFINED\_13,
- LEAKY\_UNDEFINED\_14,
- LEAKY\_UNDEFINED\_15,
- LEAKY\_UNDEFINED\_16,
- LEAKY\_UNDEFINED\_17,
- LEAKY\_UNDEFINED\_18,
- LEAKY\_UNDEFINED\_19,
- LEAKY\_UNDEFINED\_1A,
- LEAKY\_UNDEFINED\_1B,
- LEAKY\_UNDEFINED\_1C,
- LEAKY\_UNDEFINED\_1D,
- LEAKY\_UNDEFINED\_1E,
- LEAKY\_UNDEFINED\_1F,
- LEAKY\_GMRP,
- LEAKY\_GVRP,
- LEAKY\_UNDEF\_GARP\_22,
- LEAKY\_UNDEF\_GARP\_23,
- LEAKY\_UNDEF\_GARP\_24,
- LEAKY\_UNDEF\_GARP\_25,
- LEAKY\_UNDEF\_GARP\_26,
- LEAKY\_UNDEF\_GARP\_27,
- LEAKY\_UNDEF\_GARP\_28,
- LEAKY\_UNDEF\_GARP\_29,
- LEAKY\_UNDEF\_GARP\_2A,
- LEAKY\_UNDEF\_GARP\_2B,
- LEAKY\_UNDEF\_GARP\_2C,
- LEAKY\_UNDEF\_GARP\_2D,
- LEAKY\_UNDEF\_GARP\_2E,
- LEAKY\_UNDEF\_GARP\_2F,
- LEAKY\_IGMP,
- LEAKY\_IPMULTICAST,
- LEAKY\_CDP,
- LEAKY\_CSSTP,
- LEAKY\_LLDP.

<b>Return Codes</b>	RT_ERR_OK	ok
---------------------	-----------	----

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_ENABLE	Invalid enable input

## 8.4. rtk\_leaky\_portIsolation\_get

`rtk_api_ret_t rtk_leaky_portIsolation_get(rtk_leaky_type_t type,  
rtk_enable_t *pEnable)`

Get port isolation leaky.

Defined in: leaky.h

### Parameters

*type*  
Packet type for port isolation leaky.

*\*pEnable*  
Leaky status.

### Comments

This API can get port isolation leaky status for RMA ,IGMP/MLD, CDP, CSSTP, and LLDP packets. The leaky frame types are as following:

- LEAKY\_BRG\_GROUP,
- LEAKY\_FD\_PAUSE,
- LEAKY\_SP\_MCAST,
- LEAKY\_1X\_PAE,
- LEAKY\_UNDEF\_BRG\_04,
- LEAKY\_UNDEF\_BRG\_05,
- LEAKY\_UNDEF\_BRG\_06,
- LEAKY\_UNDEF\_BRG\_07,
- LEAKY\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- LEAKY\_UNDEF\_BRG\_09,
- LEAKY\_UNDEF\_BRG\_0A,
- LEAKY\_UNDEF\_BRG\_0B,
- LEAKY\_UNDEF\_BRG\_0C,
- LEAKY\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- LEAKY\_8021AB,
- LEAKY\_UNDEF\_BRG\_0F,
- LEAKY\_BRG\_MNGEMENT,
- LEAKY\_UNDEFINED\_11,
- LEAKY\_UNDEFINED\_12,
- LEAKY\_UNDEFINED\_13,
- LEAKY\_UNDEFINED\_14,
- LEAKY\_UNDEFINED\_15,

---

- LEAKY\_UNDEFINED\_16,
- LEAKY\_UNDEFINED\_17,
- LEAKY\_UNDEFINED\_18,
- LEAKY\_UNDEFINED\_19,
- LEAKY\_UNDEFINED\_1A,
- LEAKY\_UNDEFINED\_1B,
- LEAKY\_UNDEFINED\_1C,
- LEAKY\_UNDEFINED\_1D,
- LEAKY\_UNDEFINED\_1E,
- LEAKY\_UNDEFINED\_1F,
- LEAKY\_GMRP,
- LEAKY\_GVRP,
- LEAKY\_UNDEF\_GARP\_22,
- LEAKY\_UNDEF\_GARP\_23,
- LEAKY\_UNDEF\_GARP\_24,
- LEAKY\_UNDEF\_GARP\_25,
- LEAKY\_UNDEF\_GARP\_26,
- LEAKY\_UNDEF\_GARP\_27,
- LEAKY\_UNDEF\_GARP\_28,
- LEAKY\_UNDEF\_GARP\_29,
- LEAKY\_UNDEF\_GARP\_2A,
- LEAKY\_UNDEF\_GARP\_2B,
- LEAKY\_UNDEF\_GARP\_2C,
- LEAKY\_UNDEF\_GARP\_2D,
- LEAKY\_UNDEF\_GARP\_2E,
- LEAKY\_UNDEF\_GARP\_2F,
- LEAKY\_IGMP,
- LEAKY\_IPMULTICAST,
- LEAKY\_CDP,
- LEAKY\_CSSTP,
- LEAKY\_LLDP.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 9. Module led.h - Unmanaged switch high-level API

Filename: led.h

**Description** The file includes LED module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

led.h - Unmanaged switch high-level API  
rtk\_led\_enable\_set  
rtk\_led\_enable\_get  
rtk\_led\_operation\_set  
rtk\_led\_operation\_get  
rtk\_led\_modeForce\_set  
rtk\_led\_modeForce\_get  
rtk\_led\_blinkRate\_set  
rtk\_led\_blinkRate\_get  
rtk\_led\_groupConfig\_set  
rtk\_led\_groupConfig\_get  
rtk\_led\_serialMode\_set  
rtk\_led\_serialMode\_get

---

### 9.1. rtk\_led\_enable\_set

**rtk\_api\_ret\_t rtk\_led\_enable\_set(rtk\_led\_group\_t group, rtk\_portmask\_t \*pPortmask)**

Set Led enable configuration

Defined in: led.h

**Parameters**

*group*  
LED group id.  
*\*pPortmask*  
LED enable port mask.

**Comments**

The API can be used to enable LED per port per group.

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

## 9.2. rtk\_led\_enable\_get

`rtk_api_ret_t rtk_led_enable_get(rtk_led_group_t group, rtk_portmask_t *pPortmask)`

Get Led enable congiuration

Defined in: led.h

**Parameters**

*group*  
LED group id.

*\*pPortmask*  
LED enable port mask.

**Comments**

The API can be used to get LED enable status.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 9.3. rtk\_led\_operation\_set

`rtk_api_ret_t rtk_led_operation_set(rtk_led_operation_t mode)`

Set Led operation mode

Defined in: led.h

**Parameters**

*mode*  
LED operation mode.

**Comments**

The API can set Led operation mode. The modes that can be set are as following:

- LED\_OP\_SCAN,
- LED\_OP\_PARALLEL,
- LED\_OP\_SERIAL,

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 9.4. rtk\_led\_operation\_get

`rtk_api_ret_t rtk_led_operation_get(rtk_led_operation_t *pMode)`

Get Led operation mode

Defined in: led.h

<b>Parameters</b>	<i>*pMode</i> Support LED operation mode.
-------------------	--

<b>Comments</b>	The API can get Led operation mode. The modes that can be set are as following: - LED_OP_SCAN, - LED_OP_PARALLEL, - LED_OP_SERIAL,
-----------------	---

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 9.5. rtk\_led\_modeForce\_set

`rtk_api_ret_t rtk_led_modeForce_set(rtk_port_t port, rtk_led_group_t group, rtk_led_force_mode_t mode)`

Set Led group to configuration force mode

Defined in: led.h

<b>Parameters</b>	<i>port</i> port ID <i>group</i> Support LED group id.
-------------------	---

---

	<i>mode</i>	Support LED force mode.
<b>Comments</b>	The API can force to one force mode. The force modes that can be set are as following:	
	- LED_FORCE_NORMAL,	
	- LED_FORCE_BLINK,	
	- LED_FORCE_OFF,	
	- LED_FORCE_ON.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.

---

## 9.6. rtk\_led\_modeForce\_get

`rtk_api_ret_t rtk_led_modeForce_get(rtk_port_t port, rtk_led_group_t group, rtk_led_force_mode_t *pMode)`

Get Led group to congiuration force mode

Defined in: led.h

<b>Parameters</b>	<i>port</i> port ID <i>group</i> Support LED group id. <i>*pMode</i> Support LED force mode.
-------------------	---

<b>Comments</b>	The API can get forced Led group mode. The force modes that can be set are as following: - LED_FORCE_NORMAL, - LED_FORCE_BLINK, - LED_FORCE_OFF, - LED_FORCE_ON.
-----------------	--

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 9.7. rtk\_led\_blinkRate\_set

`rtk_api_ret_t rtk_led_blinkRate_set(rtk_led_blink_rate_t blinkRate)`

Set LED blinking rate

Defined in: led.h

**Parameters**

*blinkRate*  
      blinking rate.

**Comments**

ASIC support 6 types of LED blinking rates at 43ms, 84ms, 120ms, 170ms, 340ms and 670ms.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 9.8. rtk\_led\_blinkRate\_get

`rtk_api_ret_t rtk_led_blinkRate_get(rtk_led_blink_rate_t *pBlinkRate)`

Get LED blinking rate at mode 0 to mode 3

Defined in: led.h

**Parameters**

*\*pBlinkRate*  
      blinking rate.

**Comments**

There are 6 types of LED blinking rates at 43ms, 84ms, 120ms, 170ms, 340ms and 670ms.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 9.9. rtk\_led\_groupConfig\_set

`rtk_api_ret_t rtk_led_groupConfig_set(rtk_led_group_t group,  
rtk_led_config_t config)`

Set per group Led to configuration mode

Defined in: led.h

### Parameters

*group*  
LED group.  
*config*  
LED configuration

### Comments

The API can set LED indicated information configuration for each LED group with 1 to 1 led mapping to each port.

- Definition	LED Statuses	Description
- 0000	LED_Off	LED pin Tri-State.
- 0001	Dup/Col	Collision, Full duplex Indicator.
- 0010	Link/Act	Link, Activity Indicator.
- 0011	Spd1000	1000Mb/s Speed Indicator.
- 0100	Spd100	100Mb/s Speed Indicator.
- 0101	Spd10	10Mb/s Speed Indicator.
- 0110	Spd1000/Act	1000Mb/s Speed/Activity Indicator.
- 0111	Spd100/Act	100Mb/s Speed/Activity Indicator.
- 1000	Spd10/Act	10Mb/s Speed/Activity Indicator.
- 1001	Spd100 (10)/Act	10/100Mb/s Speed/Activity Indicator.
- 1010	LoopDetect	LoopDetect Indicator.
- 1011	EEE	EEE Indicator.
- 1100	Link/Rx	Link, Activity Indicator.
- 1101	Link/Tx	Link, Activity Indicator.
- 1110	Master	Link on Master Indicator.
- 1111	Act	Activity Indicator. Low for link established.

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 9.10.rtk\_led\_groupConfig\_get

```
rtk_api_ret_t rtk_led_groupConfig_get(rtk_led_group_t group,  
                                     rtk_led_config_t *pConfig)
```

Get Led group configuration mode

Defined in: led.h

<b>Parameters</b>	<i>group</i> LED group. <i>*pConfig</i> LED configuration.
<b>Comments</b>	The API can get LED indicated information configuration for each LED group.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters.

---

## 9.11.rtk\_led\_serialMode\_set

```
rtk_api_ret_t rtk_led_serialMode_set(rtk_led_active_t active)
```

Set Led serial mode active configuration

Defined in: led.h

<b>Parameters</b>	<i>active</i> LED group.
<b>Comments</b>	The API can set LED serial mode active configuration.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters.

---

## 9.12. rtk\_led\_serialMode\_get

`rtk_api_ret_t rtk_led_serialMode_get(rtk_led_active_t *pActive)`

Get Led group congiuration mode

Defined in: led.h

**Parameters**      *\*pActive*  
                  LED group.

**Comments**      The API can get LED serial mode active configuration.

**Return Codes**      RT\_ERR\_OK                                 ok  
                          RT\_ERR\_FAILED                         failed  
                          RT\_ERR\_SMI                             SMI access error  
                          RT\_ERR\_INPUT                         Invalid input parameters.

---

## 9.13. rtk\_led\_OutputEnable\_set

`rtk_api_ret_t rtk_led_OutputEnable_set(rtk_enable_t state)`

This API set LED I/O state.

Defined in: led.h

**Parameters**      *state*  
                  LED I/O state

**Comments**      This API set LED I/O state.

**Return Codes**      RT\_ERR\_OK                                 ok  
                          RT\_ERR\_FAILED                         failed  
                          RT\_ERR\_SMI                             SMI access error  
                          RT\_ERR\_INPUT                         Error parameter

---

## 9.14.rtk\_led\_OutputEnable\_get

`rtk_api_ret_t rtk_led_OutputEnable_get(rtk_enable_t *pState)`

This API get LED I/O state.

Defined in: led.h

<b>Parameters</b>	<code>*pState</code> LED I/O state								
<b>Comments</b>	This API set current LED I/O state.								
<b>Return Codes</b>	<table><tr><td><code>RT_ERR_OK</code></td><td>ok</td></tr><tr><td><code>RT_ERR_FAILED</code></td><td>failed</td></tr><tr><td><code>RT_ERR_SMI</code></td><td>SMI access error</td></tr><tr><td><code>RT_ERR_INPUT</code></td><td>Error parameter</td></tr></table>	<code>RT_ERR_OK</code>	ok	<code>RT_ERR_FAILED</code>	failed	<code>RT_ERR_SMI</code>	SMI access error	<code>RT_ERR_INPUT</code>	Error parameter
<code>RT_ERR_OK</code>	ok								
<code>RT_ERR_FAILED</code>	failed								
<code>RT_ERR_SMI</code>	SMI access error								
<code>RT_ERR_INPUT</code>	Error parameter								

---

## 9.15.rtk\_led\_groupAbility\_set

`rtk_api_ret_t rtk_led_groupAbility_set(rtk_led_group_t group,  
rtk_led_ability_t *pAbility)`

Configure per group Led ability.

Defined in: led.h

<b>Parameters</b>	<code>group</code> LED group. <code>pAbility</code> LED Ability.								
<b>Comments</b>									
<b>Return Codes</b>	<table><tr><td><code>RT_ERR_OK</code></td><td>ok</td></tr><tr><td><code>RT_ERR_FAILED</code></td><td>failed</td></tr><tr><td><code>RT_ERR_SMI</code></td><td>SMI access error</td></tr><tr><td><code>RT_ERR_INPUT</code></td><td>Error parameter</td></tr></table>	<code>RT_ERR_OK</code>	ok	<code>RT_ERR_FAILED</code>	failed	<code>RT_ERR_SMI</code>	SMI access error	<code>RT_ERR_INPUT</code>	Error parameter
<code>RT_ERR_OK</code>	ok								
<code>RT_ERR_FAILED</code>	failed								
<code>RT_ERR_SMI</code>	SMI access error								
<code>RT_ERR_INPUT</code>	Error parameter								

---

## 9.16. rtk\_led\_groupAbility\_get

```
rtk_api_ret_t rtk_led_groupAbility_get(rtk_led_group_t group,  
                                      rtk_led_ability_t *pAbility)
```

This API get per group Led ability.

Defined in: led.h

**Parameters**

*group*  
LED group.  
*pAbility*  
LED Ability.

**Comments****Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error parameter

---

## 9.17. rtk\_led\_serialModePortmask\_set

```
rtk_api_ret_t rtk_led_serialModePortmask_set(rtk_led_serialOutput_t  
                                              output, rtk_portmask_t *pPortmask)
```

This API configure Serial LED output Group and portmask.

Defined in: led.h

**Parameters**

*output*  
output group.  
*pPortmask*  
output portmask.

**Comments****Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error parameter

---

## 9.18.rtk\_led\_serialModePortmask\_get

```
rtk_api_ret_t rtk_led_serialModePortmask_get(rtk_led_serialOutput_t  
output, rtk_portmask_t *pPortmask)
```

This API get serial LED output Group and portmask.

Defined in: led.h

**Parameters**

*output*  
output group.  
*pPortmask*  
output portmask.

**Comments****Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error parameter

---

## 10. Module mirror.h - Unmanaged switch high-level API

Filename: mirror.h

**Description**

The file includes Mirror module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

**List of Symbols**

Here is a list of all functions and variables in this module

mirror.h - Unmanaged switch high-level API  
rtk\_mirror\_portBased\_set  
rtk\_mirror\_portBased\_get  
rtk\_mirror\_portIso\_set  
rtk\_mirror\_portIso\_get  
rtk\_mirror\_vlanLeaky\_set

---

```
rtk_mirror_vlanLeaky_get  
rtk_mirror_isolationLeaky_set  
rtk_mirror_isolationLeaky_get  
rtk_mirror_keep_set  
rtk_mirror_keep_get
```

---

## 10.1. rtk\_mirror\_portBased\_set

`rtk_api_ret_t rtk_mirror_portBased_set(rtk_port_t mirroring_port,  
rtk_portmask_t *pMirrored_rx_portmask, rtk_portmask_t  
*pMirrored_tx_portmask)`

Set port mirror function.

Defined in: mirror.h

### Parameters

*mirroring\_port*

Monitor port.

\**pMirrored\_rx\_portmask*

Rx mirror port mask.

\**pMirrored\_tx\_portmask*

Tx mirror port mask.

### Comments

The API is to set mirror function of source port and mirror port. The mirror port can only be set to one port and the TX and RX mirror ports should be identical.

### Return Codes

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number

RT\_ERR\_PORT\_MASK

Invalid portmask.

---

## 10.2. rtk\_mirror\_portBased\_get

`rtk_api_ret_t rtk_mirror_portBased_get(rtk_port_t* pMirroring_port,  
rtk_portmask_t *pMirrored_rx_portmask, rtk_portmask_t  
*pMirrored_tx_portmask)`

Get port mirror function.

	Defined in: mirror.h								
<b>Parameters</b>	<p><i>pMirroring_port</i> Monitor port.</p> <p><i>*pMirrored_rx_portmask</i> Rx mirror port mask.</p> <p><i>*pMirrored_tx_portmask</i> Tx mirror port mask.</p>								
<b>Comments</b>	The API is to get mirror function of source port and mirror port.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

---

### 10.3. rtk\_mirror\_portIso\_set

`rtk_api_ret_t rtk_mirror_portIso_set(rtк_enable_t enable)`

Set mirror port isolation.

Defined in: mirror.h

<b>Parameters</b>	<i>enable</i> Monitor port.
<b>Comments</b>	The API is to set mirror isolation function that prevent normal forwarding packets to mirror port.

<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_ENABLE</td><td>Invalid enable input</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_ENABLE	Invalid enable input
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_ENABLE	Invalid enable input								

---

### 10.4. rtk\_mirror\_portIso\_get

`rtk_api_ret_t rtk_mirror_portIso_get(rtк_enable_t *pEnable)`

Get mirror port isolation.

---

	Defined in: mirror.h
<b>Parameters</b>	<i>*pEnable</i> Monitor port.
<b>Comments</b>	The API is to get mirror isolation status.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters.

---

## 10.5. rtk\_mirror\_vlanLeaky\_set

	<code>rtk_api_ret_t rtk_mirror_vlanLeaky_set(rtк_enable_t txenable, rtк_enable_t rxenable)</code>
	Set mirror VLAN leaky.
	Defined in: mirror.h
<b>Parameters</b>	<i>txenable</i> TX leaky enable. <i>rxenable</i> RX leaky enable.
<b>Comments</b>	The API is to set mirror VLAN leaky function forwarding packets to mirror port.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_ENABLE Invalid enable input

---

## 10.6. rtk\_mirror\_vlanLeaky\_get

	<code>rtk_api_ret_t rtk_mirror_vlanLeaky_get(rtк_enable_t *pTxenable, rtк_enable_t *pRxenable)</code>
	Get mirror VLAN leaky.
	Defined in: mirror.h

<b>Parameters</b>	<i>*pTxenable</i> TX leaky enable.								
	<i>*pRxenable</i> RX leaky enable.								
<b>Comments</b>	The API is to get mirror VLAN leaky status.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

---

## 10.7.rtk\_mirror\_isolationLeaky\_set

```
rtk_api_ret_t rtk_mirror_isolationLeaky_set(rtk_enable_t txenable,
                                            rtk_enable_t rxenable)
```

Set mirror Isolation leaky.

Defined in: mirror.h

<b>Parameters</b>	<i>txenable</i> TX leaky enable.								
	<i>rxenable</i> RX leaky enable.								
<b>Comments</b>	The API is to set mirror VLAN leaky function forwarding packets to miro port.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_ENABLE</td><td>Invalid enable input</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_ENABLE	Invalid enable input
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_ENABLE	Invalid enable input								

---

## 10.8.rtk\_mirror\_isolationLeaky\_get

```
rtk_api_ret_t rtk_mirror_isolationLeaky_get(rtk_enable_t *pTxenable,
                                             rtk_enable_t *pRxenable)
```

Get mirror isolation leaky.

Defined in: mirror.h

---

<b>Parameters</b>	<i>*pTxenable</i> TX leaky enable.								
	<i>*pRxenable</i> RX leaky enable.								
<b>Comments</b>	The API is to get mirror isolation leaky status.								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

---

## 10.9. rtk\_mirror\_keep\_set

`rtk_api_ret_t rtk_mirror_keep_set(rtk_mirror_keep_t mode)`

Set mirror packet format keep.

Defined in: mirror.h

<b>Parameters</b>	<i>mode</i>								
<b>Comments</b>	The API is to set -mirror keep mode. The mirror keep mode is as following: - MIRROR_FOLLOW_VLAN - MIRROR_KEEP_ORIGINAL - MIRROR_KEEP_END								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_ENABLE</td> <td>Invalid enable input</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_ENABLE	Invalid enable input
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_ENABLE	Invalid enable input								

---

## 10.10. rtk\_mirror\_keep\_get

`rtk_api_ret_t rtk_mirror_keep_get(rtk_mirror_keep_t *pMode)`

Get mirror packet format keep.

Defined in: mirror.h

<b>Parameters</b>	<i>*pMode</i> mirror keep mode.	
<b>Comments</b>	The API is to get mirror keep mode. - MIRROR_FOLLOW_VLAN - MIRROR_KEEP_ORIGINAL - MIRROR_KEEP_END	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.

## 10.11. rtk\_mirror\_override\_set

```
rtk_api_ret_t rtk_mirror_override_set(rtk_enable_t rxMirror, rtk_enable_t txMirror, rtk_enable_t aclMirror)
```

Set port mirror override function.

Defined in: mirror.h

<b>Parameters</b>	<i>rxMirror</i> 1: output mirrored packet, 0: output normal forward packet	
	<i>txMirror</i> 1: output mirrored packet, 0: output normal forward packet	
	<i>aclMirror</i> 1: output mirrored packet, 0: output normal forward packet	
<b>Comments</b>	The API is to set mirror override function. This function control the output format when a port output normal forward & mirrored packet at the same time.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI	ok failed SMI access error

---

## 10.12. rtk\_mirror\_override\_get

```
rtk_api_ret_t rtk_mirror_override_get(rtk_enable_t *pRxMirror,  
rtk_enable_t *pTxMirror, rtk_enable_t *pAclMirror)
```

Get port mirror override function.

---

	Defined in: mirror.h								
<b>Parameters</b>	<p><i>*pRxMirror</i> 1: output mirrored packet, 0: output normal forward packet</p> <p><i>*pTxMirror</i> 1: output mirrored packet, 0: output normal forward packet</p> <p><i>*pAclMirror</i> 1: output mirrored packet, 0: output normal forward packet</p>								
<b>Comments</b>	The API is to Get mirror override function. This function control the output format when a port output normal forward & mirrored packet at the same time.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_NULL_POINTER</td> <td>Null Pointer</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_NULL_POINTER	Null Pointer
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_NULL_POINTER	Null Pointer								

---

## 11. Module oam.h - Unmanaged switch high-level API

	Filename: oam.h
<b>Description</b>	<p>The file includes the following modules and sub-modules (1) OAM (802.3ah) configuration</p> <p>Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.</p>
	List of Symbols

Here is a list of all functions and variables in this module

oam.h - Unmanaged switch high-level API

*rtk\_oam\_init*  
*rtk\_oam\_state\_set*  
*rtk\_oam\_state\_get*  
*rtk\_oam\_parserAction\_set*  
*rtk\_oam\_parserAction\_get*  
*rtk\_oam\_multiplexerAction\_set*  
*rtk\_oam\_multiplexerAction\_get*

---

## 11.1.rtk\_oam\_init

`rtk_api_ret_t rtk_oam_init( void)`

Initialize oam module.

Defined in: oam.h

**Parameters**

*void*

**Comments**

Must initialize oam module before calling any oam APIs.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed

---

## 11.2.rtk\_oam\_state\_set

`rtk_api_ret_t rtk_oam_state_set(rt_k_enable_t enabled)`

This API set OAM state.

Defined in: oam.h

**Parameters**

*enabled*  
OAMstate

**Comments**

This API set OAM state.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Error parameter

---

## 11.3.rtk\_oam\_state\_get

`rtk_api_ret_t rtk_oam_state_get(rt_k_enable_t *pEnabled)`

This API get OAM state.

---

	Defined in: oam.h
<b>Parameters</b>	<i>*pEnabled</i> H/W IGMP state
<b>Comments</b>	This API set current OAM state.
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT
	ok failed SMI access error Error parameter

---

## 11.4. rtk\_oam\_parserAction\_set

	<code>rtk_api_ret_t rtk_oam_parserAction_set(rtk_port_t port, rtk_oam_parser_act_t action)</code>
	Set OAM parser action
	Defined in: oam.h
<b>Parameters</b>	<i>port</i> port id <i>action</i> parser action
<b>Comments</b>	None
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_PORT_ID
	ok failed invalid port id

---

## 11.5. rtk\_oam\_parserAction\_get

	<code>rtk_api_ret_t rtk_oam_parserAction_get(rtk_port_t port, rtk_oam_parser_act_t *pAction)</code>
	Get OAM parser action
	Defined in: oam.h
<b>Parameters</b>	

<i>port</i>	port id						
<i>*pAction</i>	parser action						
<b>Comments</b>	None						
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>invalid port id</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_PORT_ID	invalid port id
RT_ERR_OK	ok						
RT_ERR_FAILED	failed						
RT_ERR_PORT_ID	invalid port id						

---

## 11.6.rtk\_oam\_multiplexerAction\_set

`rtk_api_ret_t rtk_oam_multiplexerAction_set(rtk_port_t port,  
 rtk_oam_multiplexer_act_t action)`

Set OAM multiplexer action

Defined in: oam.h

<b>Parameters</b>	<i>port</i> port id <i>action</i> parser action						
<b>Comments</b>	None						
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>invalid port id</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_PORT_ID	invalid port id
RT_ERR_OK	ok						
RT_ERR_FAILED	failed						
RT_ERR_PORT_ID	invalid port id						

---

## 11.7.rtk\_oam\_multiplexerAction\_get

`rtk_api_ret_t rtk_oam_multiplexerAction_get(rtk_port_t port,  
 rtk_oam_multiplexer_act_t *pAction)`

Get OAM multiplexer action

Defined in: oam.h

<b>Parameters</b>
-------------------

---

<i>port</i>	port id
<i>*pAction</i>	parser action
<b>Comments</b>	None
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id

---

## 12. Module port.h - Unmanaged switch high-level API

Filename: port.h

**Description** The file includes port module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

port.h - Unmanaged switch high-level API  
 rtk\_port\_phyAutoNegoAbility\_set  
 rtk\_port\_phyAutoNegoAbility\_get  
 rtk\_port\_phyForceModeAbility\_set  
 rtk\_port\_phyForceModeAbility\_get  
 rtk\_port\_phyStatus\_get  
 rtk\_port\_macForceLink\_set  
 rtk\_port\_macForceLink\_get  
 rtk\_port\_macForceLinkExt\_set  
 rtk\_port\_macForceLinkExt\_get  
 rtk\_port\_macStatus\_get  
 rtk\_port\_macLocalLoopbackEnable\_set  
 rtk\_port\_macLocalLoopbackEnable\_get  
 rtk\_port\_phyReg\_set  
 rtk\_port\_phyReg\_get  
 rtk\_port\_backpressureEnable\_set  
 rtk\_port\_backpressureEnable\_get  
 rtk\_port\_adminEnable\_set  
 rtk\_port\_adminEnable\_get

```
rtk_port_isolation_set  
rtk_port_isolation_get  
rtk_port_rgmiiDelayExt_set  
rtk_port_rgmiiDelayExt_get  
rtk_port_phyEnableAll_set  
rtk_port_phyEnableAll_get  
rtk_port_efid_set  
rtk_port_efid_get  
rtk_port_phyComboPortMedia_set  
rtk_port_phyComboPortMedia_get  
rtk_port_rtctEnable_set  
rtk_port_rtctResult_get
```

## 12.1.rtk\_port\_phyAutoNegoAbility\_set

`rtk_api_ret_t rtk_port_phyAutoNegoAbility_set(rtk_port_t port,  
 rtk_port_phy_ability_t *pAbility)`

Set ethernet PHY auto-negotiation desired ability.

Defined in: port.h

### Parameters

*port*

port id.

*\*pAbility*

Ability structure

### Comments

If Full\_1000 bit is set to 1, the AutoNegotiation will be automatic set to 1. While both AutoNegotiation and Full\_1000 are set to 0, the PHY speed and duplex selection will be set as following 100F > 100H > 10F > 10H priority sequence.

### Return Codes

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.
<code>RT_ERR_PHY_REG_ID</code>	Invalid PHY address
<code>RT_ERR_INPUT</code>	Invalid input parameters.
<code>RT_ERR_BUSYWAIT_TIMEOUT</code>	PHY access busy

---

---

## 12.2. rtk\_port\_phyAutoNegoAbility\_get

`rtk_api_ret_t rtk_port_phyAutoNegoAbility_get(rtk_port_t port,  
rtk_port_phy_ability_t *pAbility)`

Get PHY ability through PHY registers.

Defined in: port.h

**Parameters**

*port*

Port id.

*\*pAbility*

Ability structure

**Comments**

Get the capability of specified PHY.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_PHY\_REG\_ID

Invalid PHY address

RT\_ERR\_INPUT

Invalid input parameters.

RT\_ERR\_BUSYWAIT\_TIMEOUT

PHY access busy

---

## 12.3. rtk\_port\_phyForceModeAbility\_set

`rtk_api_ret_t rtk_port_phyForceModeAbility_set(rtk_port_t port,  
rtk_port_phy_ability_t *pAbility)`

Set the port speed/duplex mode/pause/asy\_pause in the PHY force mode.

Defined in: port.h

**Parameters**

*port*

port id.

*\*pAbility*

Ability structure

**Comments**

If Full\_1000 bit is set to 1, the AutoNegotiation will be automatic set to 1. While both AutoNegotiation and Full\_1000 are set to 0, the PHY speed and duplex selection will be set as following 100F > 100H > 10F > 10H priority sequence.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_PHY_REG_ID RT_ERR_INPUT RT_ERR_BUSYWAIT_TIMEOUT	ok failed SMI access error Invalid port number. Invalid PHY address Invalid input parameters. PHY access busy
---------------------	--	---

---

## 12.4. rtk\_port\_phyForceModeAbility\_get

```
rtk_api_ret_t rtk_port_phyForceModeAbility_get(rtk_port_t port,
                                              rtk_port_phy_ability_t *pAbility)
```

Get PHY ability through PHY registers.

Defined in: port.h

<b>Parameters</b>	<i>port</i> Port id. <i>*pAbility</i> Ability structure
<b>Comments</b>	
Get the capability of specified PHY.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_PHY_REG_ID RT_ERR_INPUT RT_ERR_BUSYWAIT_TIMEOUT
	ok failed SMI access error Invalid port number. Invalid PHY address Invalid input parameters. PHY access busy

---

## 12.5. rtk\_port\_phyStatus\_get

```
rtk_api_ret_t rtk_port_phyStatus_get(rtk_port_t port, rtk_port_linkStatus_t
                                      *pLinkStatus, rtk_port_speed_t *pSpeed, rtk_port_duplex_t *pDuplex)
```

Get ethernet PHY linking status

---

	Defined in: port.h														
<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>*pLinkStatus</i> PHY link status</p> <p><i>*pSpeed</i> PHY link speed</p> <p><i>*pDuplex</i> PHY duplex mode</p>														
<b>Comments</b>	API will return auto negotiation status of phy.														
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> <tr> <td>RT_ERR_PHY_REG_ID</td> <td>Invalid PHY address</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> <tr> <td>RT_ERR_BUSYWAIT_TIMEOUT</td> <td>PHY access busy</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_PHY_REG_ID	Invalid PHY address	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_BUSYWAIT_TIMEOUT	PHY access busy
RT_ERR_OK	ok														
RT_ERR_FAILED	failed														
RT_ERR_SMI	SMI access error														
RT_ERR_PORT_ID	Invalid port number.														
RT_ERR_PHY_REG_ID	Invalid PHY address														
RT_ERR_INPUT	Invalid input parameters.														
RT_ERR_BUSYWAIT_TIMEOUT	PHY access busy														

---

## 12.6. rtk\_port\_macForceLink\_set

`rtk_api_ret_t rtk_port_macForceLink_set(rtk_port_t port,  
rtk_port_mac_ability_t *pPortability)`

Set port force linking configuration.

Defined in: port.h

<b>Parameters</b>	<p><i>port</i> port id.</p> <p><i>*pPortability</i> port ability configuration</p>								
<b>Comments</b>	This API can set Port/MAC force mode properties.								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port number.								

---

## 12.7.rtk\_port\_macForceLink\_get

`rtk_api_ret_t rtk_port_macForceLink_get(rtk_port_t port,  
rtk_port_mac_ability_t *pPortability)`

Get port force linking configuration.

Defined in: port.h

**Parameters**

*port*

Port id.

*\*pPortability*

port ability configuration

**Comments**

This API can get Port/MAC force mode properties.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_INPUT

Invalid input parameters.

---

## 12.8.rtk\_port\_macForceLinkExt\_set

`rtk_api_ret_t rtk_port_macForceLinkExt_set(rtk_port_t port,  
rtk_mode_ext_t mode, rtk_port_mac_ability_t *pPortability)`

Set external interface force linking configuration.

Defined in: port.h

**Parameters**

*port*

external port ID

*mode*

external interface mode

*\*pPortability*

port ability configuration

**Comments**

This API can set external interface force mode properties. The external interface can be set to:

- MODE\_EXT\_DISABLE,
- MODE\_EXT\_RGMII,

---

	- MODE_EXT_MII_MAC, - MODE_EXT_MII_PHY, - MODE_EXT_TMII_MAC, - MODE_EXT_TMII_PHY, - MODE_EXT_GMII, - MODE_EXT_RMII_MAC, - MODE_EXT_RMII_PHY, - MODE_EXT_SGMII, - MODE_EXT_HSGMII	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.

---

## 12.9. rtk\_port\_macForceLinkExt\_get

`rtk_api_ret_t rtk_port_macForceLinkExt_get(rtk_port_t port,  
rtk_mode_ext_t *pMode, rtk_port_mac_ability_t *pPortability)`

Set external interface force linking configuration.

Defined in: port.h

### Parameters

*port*  
external port ID  
*\*pMode*  
external interface mode  
*\*pPortability*  
port ability configuration

### Comments

This API can get external interface force mode properties.

### Return Codes

	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
--	--	---

---

## 12.10. rtk\_port\_macStatus\_get

```
rtk_api_ret_t rtk_port_macStatus_get(rtk_port_t port,  
                                     rtk_port_mac_ability_t *pPortstatus)
```

Get port link status.

Defined in: port.h

**Parameters**

*port*  
Port id.  
*\*pPortstatus*  
port ability configuration

**Comments**

This API can get Port/PHY properties.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

---

## 12.11. rtk\_port\_macLocalLoopbackEnable\_set

```
rtk_api_ret_t rtk_port_macLocalLoopbackEnable_set(rtk_port_t port,  
                                                 rtk_enable_t enable)
```

Set Port Local Loopback. (Redirect TX to RX.)

Defined in: port.h

**Parameters**

*port*  
Port id.  
*enable*  
Loopback state, 0:disable, 1:enable

**Comments**

This API can enable/disable Local loopback in MAC. For UTP port, This API will also enable the digital loopback bit in PHY register for sync of speed between PHY and MAC. For EXT port, users need to force the link state by themself.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

RT_ERR_PORT_ID	Invalid port number.
----------------	----------------------

---

## 12.12. rtk\_port\_macLocalLoopbackEnable\_get

`rtk_api_ret_t rtk_port_macLocalLoopbackEnable_get(rtk_port_t port,  
rtk_enable_t *pEnable)`

Get Port Local Loopback. (Redirect TX to RX.)

Defined in: port.h

**Parameters**

*port*  
Port id.  
*\*pEnable*  
Loopback state, 0:disable, 1:enable

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.

## 12.13. rtk\_port\_phyReg\_set

`rtk_api_ret_t rtk_port_phyReg_set(rtk_port_t port, rtk_port_phy_reg_t reg,  
rtk_port_phy_data_t value)`

Set PHY register data of the specific port.

Defined in: port.h

**Parameters**

*port*  
port id.  
*reg*  
Register id  
*value*  
Register data

**Comments**

This API can set PHY register data of the specific port.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_PHY_REG_ID RT_ERR_BUSYWAIT_TIMEOUT	ok failed SMI access error Invalid port number. Invalid PHY address PHY access busy
---------------------	--	--

---

## 12.14. rtk\_port\_phyReg\_get

`rtk_api_ret_t rtk_port_phyReg_get(rtk_port_t port, rtk_port_phy_reg_t reg,  
rtk_port_phy_data_t *pData)`

Get PHY register data of the specific port.

Defined in: port.h

<b>Parameters</b>	<i>port</i> Port id. <i>reg</i> Register id <i>*pData</i> Register data
<b>Comments</b>	This API can get PHY register data of the specific port.
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_PHY_REG_ID RT_ERR_BUSYWAIT_TIMEOUT

---

## 12.15. rtk\_port\_backpressureEnable\_set

`rtk_api_ret_t rtk_port_backpressureEnable_set(rtk_port_t port,  
rtk_enable_t enable)`

Set the half duplex backpressure enable status of the specific port.

---

	Defined in: port.h										
<b>Parameters</b>	<p><i>port</i> port id.</p> <p><i>enable</i> Back pressure status.</p>										
<b>Comments</b>	This API can set the half duplex backpressure enable status of the specific port. The half duplex backpressure enable status of the port is as following: - DISABLE - ENABLE										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> <tr> <td>RT_ERR_ENABLE</td> <td>Invalid enable input.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_ENABLE	Invalid enable input.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_PORT_ID	Invalid port number.										
RT_ERR_ENABLE	Invalid enable input.										

---

## 12.16. rtk\_port\_backpressureEnable\_get

```
rtk_api_ret_t rtk_port_backpressureEnable_get(rtk_port_t *port,
                                              rtk_enable_t *pEnable)
```

Get the half duplex backpressure enable status of the specific port.

Defined in: port.h

<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>*pEnable</i> Back pressure status.</p>								
<b>Comments</b>	This API can get the half duplex backpressure enable status of the specific port. The half duplex backpressure enable status of the port is as following: - DISABLE - ENABLE								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port number.								

---

## 12.17. rtk\_port\_adminEnable\_set

`rtk_api_ret_t rtk_port_adminEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set port admin configuration of the specific port.

Defined in: port.h

**Parameters**

*port*  
port id.

*enable*  
Back pressure status.

**Comments**

This API can set port admin configuration of the specific port. The port admin configuration of the port is as following:

- DISABLE
- ENABLE

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.
<code>RT_ERR_ENABLE</code>	Invalid enable input.

---

## 12.18. rtk\_port\_adminEnable\_get

`rtk_api_ret_t rtk_port_adminEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get port admin configuration of the specific port.

Defined in: port.h

**Parameters**

*port*  
Port id.  
*\*pEnable*  
Back pressure status.

**Comments**

This API can get port admin configuration of the specific port. The port admin configuration of the port is as following:

---

	- DISABLE	
	- ENABLE	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.

---

## 12.19. rtk\_port\_isolation\_set

`rtk_api_ret_t rtk_port_isolation_set(rtk_port_t port, rtk_portmask_t *pPortmask)`

Set permitted port isolation portmask

Defined in: port.h

<b>Parameters</b>	<i>port</i>	port id.
	<i>*pPortmask</i>	Permit port mask

**Comments** This API set the port mask that a port can transmit packet to of each port A port can only transmit packet to ports included in permitted portmask

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.
	RT_ERR_PORT_MASK	Invalid portmask.

---

## 12.20. rtk\_port\_isolation\_get

`rtk_api_ret_t rtk_port_isolation_get(rtk_port_t port, rtk_portmask_t *pPortmask)`

Get permitted port isolation portmask

Defined in: port.h

<b>Parameters</b>	<i>port</i> Port id.  <i>*pPortmask</i> Permit port mask
<b>Comments</b>	This API get the port mask that a port can trasmit packet to of each port A port can only transmit packet to ports included in permitted portmask
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port number.

---

## 12.21. rtk\_port\_rgmiiDelayExt\_set

`rtk_api_ret_t rtk_port_rgmiiDelayExt_set(rtk_port_t port, rtk_data_t txDelay, rtk_data_t rxDelay)`

Set RGMII interface delay value for TX and RX.

Defined in: port.h

<b>Parameters</b>	<i>port</i> TX delay value, 1 for delay 2ns and 0 for no  <i>txDelay</i> RX delay value, 0~7 for delay setup.  <i>rxDelay</i> Register data
<b>Comments</b>	This API can set external interface 2 RGMII delay. In TX delay, there are 2 selection: no-delay and 2ns delay. In RX dekay, there are 8 steps for delay tunning. 0 for no-delay, and 7 for maximum delay.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters.

---

---

## 12.22. rtk\_port\_rgmiiDelayExt\_get

`rtk_api_ret_t rtk_port_rgmiiDelayExt_get(rtk_port_t port, rtk_data_t *pTxDelay, rtk_data_t *pRxDelay)`

Get RGMII interface delay value for TX and RX.

Defined in: port.h

**Parameters**

*port*  
TX delay value  
*\*pTxDelay*  
RX delay value  
*\*pRxDelay*  
Register data

**Comments**

This API can set external interface 2 RGMII delay. In TX delay, there are 2 selection: no-delay and 2ns delay. In RX dekay, there are 8 steps for delay tunning. 0 for n0-delay, and 7 for maximum delay.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 12.23. rtk\_port\_phyEnableAll\_set

`rtk_api_ret_t rtk_port_phyEnableAll_set(rtk_enable_t enable)`

Set all PHY enable status.

Defined in: port.h

**Parameters**

*enable*  
PHY Enable State.

**Comments**

This API can set all PHY status. The configuration of all PHY is as following:

- DISABLE
- ENABLE

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_SMI	SMI access error
RT_ERR_ENABLE	Invalid enable input.

---

## 12.24. rtk\_port\_phyEnableAll\_get

`rtk_api_ret_t rtk_port_phyEnableAll_get(rtk_enable_t *pEnable)`

Get all PHY enable status.

Defined in: port.h

**Parameters**

*\*pEnable*  
PHY Enable State.

**Comments**

This API can set all PHY status. The configuration of all PHY is as following:

- DISABLE
- ENABLE

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 12.25. rtk\_port\_efid\_set

`rtk_api_ret_t rtk_port_efid_set(rtk_port_t port, rtk_data_t efid)`

Set port-based enhanced filtering database

Defined in: port.h

**Parameters**

*port*  
Port id.

*efid*  
Specified enhanced filtering database.

**Comments**

The API can set port-based enhanced filtering database.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_L2_FID	Invalid fid.

---

RT_ERR_INPUT	Invalid input parameter.
RT_ERR_PORT_ID	Invalid port ID.

---

## 12.26. rtk\_port\_efid\_get

`rtk_api_ret_t rtk_port_efid_get(rtk_port_t port, rtk_data_t *pEfid)`

Get port-based enhanced filtering database

Defined in: port.h

**Parameters**

*port*

Port id.

*\*pEfid*

Specified enhanced filtering database.

**Comments**

The API can get port-based enhanced filtering database status.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_INPUT

Invalid input parameters.

RT\_ERR\_PORT\_ID

Invalid port ID.

## 12.27. rtk\_port\_phyComboPortMedia\_set

`rtk_api_ret_t rtk_port_phyComboPortMedia_set(rtk_port_t port, rtk_port_media_t media)`

Set Combo port media type

Defined in: port.h

**Parameters**

*port*

Port id. (Should be Port 4)

*media*

Media (COPPER or FIBER)

**Comments**

The API can Set Combo port media type.

**Return Codes**

RT\_ERR\_OK

ok

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port ID.

## 12.28. rtk\_port\_phyComboPortMedia\_get

`rtk_api_ret_t rtk_port_phyComboPortMedia_get(rtk_port_t port,  
rtk_port_media_t *pMedia)`

Get Combo port media type

Defined in: port.h

**Parameters**

*port*  
Port id. (Should be Port 4)

*\*pMedia*  
Media (COPPER or FIBER)

**Comments**

The API can Set Combo port media type.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port ID.

## 12.29. rtk\_port\_rtctEnable\_set

`rtk_api_ret_t rtk_port_rtctEnable_set(rtk_portmask_t *pPortmask)`

Enable RTCT test

Defined in: port.h

**Parameters**

*\*pPortmask*  
Port mask of RTCT enabled port

**Comments**

The API can enable RTCT Test

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	invalid port mask.

---

## 12.30. rtk\_port\_rtctDisable\_set

`rtk_api_ret_t rtk_port_rtctDisable_set(rtk_portmask_t *pPortmask)`

Disable RTCT test

Defined in: port.h

**Parameters**      `*pPortmask`  
                  Port mask of RTCT disabled port

**Comments**      The API can disable RTCT Test

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_MASK	Invalid port mask.

---

## 12.31. rtk\_port\_rtctResult\_get

`rtk_api_ret_t rtk_port_rtctResult_get(rtk_port_t port, rtk_rtctResult_t *pRtctResult)`

Get the result of RTCT test

Defined in: port.h

**Parameters**      `port`  
                  Port ID  
`*pRtctResult`  
                  The result of RTCT result

**Comments**      The API can get RTCT test result. RTCT test may takes 4.8 seconds to finish its test at most. Thus, if this API return RT\_ERR\_PHY\_RTCT\_NOT\_FINISH or other error code, the result can not be referenced and user should call this API again until this API returns a RT\_ERR\_OK. The result is stored at

pRtctResult->ge\_result pRtctResult->linkType is unused. The unit of channel length is 2.5cm. Ex. 300 means  $300 * 2.5 = 750$ cm = 7.5M

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port ID.
	RT_ERR_PHY_RTCT_NOT_FINISH	Testing does not finish.

---

### 12.32. rtk\_port\_sds\_reset

`rtk_api_ret_t rtk_port_sds_reset(rtk_port_t port)`

Reset Serdes

Defined in: port.h

<b>Parameters</b>	<i>port</i>	
	Port ID	

<b>Comments</b>	The API can reset Serdes
-----------------	--------------------------

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port ID.

---

### 12.33. rtk\_port\_sgmiiLinkStatus\_get

`rtk_api_ret_t rtk_port_sgmiiLinkStatus_get(rtk_port_t port, rtk_data_t *pSignalDetect, rtk_data_t *pSync, rtk_port_linkStatus_t *pLink)`

Get SGMII status.

Defined in: port.h

<b>Parameters</b>	<i>port</i>	
	Port ID	
	<i>pSignalDetect</i>	Signal detect

---

<i>pSync</i>	Sync								
<i>pLink</i>	Link								
<b>Comments</b>	The API can get SGMII status								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port ID.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port ID.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port ID.								

---

### 12.34. rtk\_port\_sgmiiNway\_set

`rtk_api_ret_t rtk_port_sgmiiNway_set(rtk_port_t port, rtk_enable_t state)`

Configure SGMII/HSGMII port Nway state.

Defined in: port.h

<b>Parameters</b>	<table> <tr> <td><i>port</i></td><td>Port ID</td></tr> <tr> <td><i>state</i></td><td>Nway state</td></tr> </table>	<i>port</i>	Port ID	<i>state</i>	Nway state				
<i>port</i>	Port ID								
<i>state</i>	Nway state								
<b>Comments</b>	The API can configure SGMII/HSGMII port Nway state								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port ID.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port ID.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port ID.								

---

### 12.35. rtk\_port\_sgmiiNway\_get

`rtk_api_ret_t rtk_port_sgmiiNway_get(rtk_port_t port, rtk_enable_t *pState)`

Get SGMII/HSGMII port Nway state.

Defined in: port.h

<b>Parameters</b>	<i>port</i> Port ID								
	<i>pState</i> Nway state								
<b>Comments</b>	The API can get SGMII/HSGMII port Nway state								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port ID.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port ID.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port ID.								

---

## 13. Module ptp.h - Unmanaged switch high-level API

Filename: ptp.h

**Description** The file includes time module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

```

ptp.h - Unmanaged switch high-level API
rtk_ptp_init
rtk_ptp_mac_set
rtk_ptp_mac_get
rtk_ptp_tpid_set
rtk_ptp_tpid_get
rtk_ptp_refTime_set
rtk_ptp_refTime_get
rtk_ptp_refTimeAdjust_set
rtk_ptp_refTimeEnable_set
rtk_ptp_refTimeEnable_get
rtk_ptp_portEnable_set
rtk_ptp_portEnable_get
rtk_ptp_portTimestamp_get
rtk_ptp_intControl_set
rtk_ptp_intControl_get

```

---

```
rtk_ptp_intStatus_get  
rtk_ptp_portIntStatus_set  
rtk_ptp_portIntStatus_get  
rtk_ptp_portTrap_set  
rtk_ptp_portTrap_get
```

---

### 13.1. rtk\_ptp\_init

`rtk_api_ret_t rtk_ptp_init( void )`

PTP function initialization.

Defined in: ptph.h

**Parameters**

*void*

**Comments**

This API is used to initialize EEE status.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

### 13.2. rtk\_ptp\_mac\_set

`rtk_api_ret_t rtk_ptp_mac_set(rtk_mac_t mac)`

Configure PTP mac address.

Defined in: ptph.h

**Parameters**

*mac*  
mac address to parser PTP packets.

**Comments**

None

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameter.

---

### 13.3.rtk\_ptp\_mac\_get

`rtk_api_ret_t rtk_ptp_mac_get(rtk_mac_t *pMac)`

Get PTP mac address.

Defined in: ptp.h

**Parameters**      `*pMac`  
                          mac address to parser PTP packets.

**Comments**      None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.

---

### 13.4.rtk\_ptp\_tpid\_set

`rtk_api_ret_t rtk_ptp_tpid_set(rtk_ptp_tpid_t outerId, rtk_ptp_tpid_t innerId)`

Configure PTP accepted outer & inner tag TPID.

Defined in: ptp.h

**Parameters**      `outerId`  
                          Ether type of S

`innerId`  
                          Ether type of C

**Comments**      None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.

---

---

## 13.5. rtk\_ptp\_tpid\_get

`rtk_api_ret_t rtk_ptp_tpid_get(rtk_ptp_tpid_t *pOuterId, rtk_ptp_tpid_t *pInnerId)`

Get PTP accepted outer & inner tag TPID.

Defined in: ptph.h

**Parameters**

*\*pOuterId*  
Ether type of S  
*\*pInnerId*  
Ether type of C

**Comments**

None

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 13.6. rtk\_ptp\_refTime\_set

`rtk_api_ret_t rtk_ptp_refTime_set(rtk_ptp_timeStamp_t timeStamp)`

Set the reference time of the specified device.

Defined in: ptph.h

**Parameters**

*timeStamp*  
reference timestamp value

**Comments**

None

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_INPUT</code>	invalid input parameter
Applicable: 8390,8380	Invalid input parameter.

## 13.7. rtk\_ptp\_refTime\_get

```
rtk_api_ret_t rtk_ptp_refTime_get(rtk_ptp_timeStamp_t *pTimeStamp)
```

Get the reference time of the specified device.

Defined in: ptp.h

<b>Parameters</b>	<i>*pTimeStamp</i> pointer buffer of the reference time
<b>Comments</b>	None
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_UNIT_ID RT_ERR_NOT_INIT RT_ERR_NULL_POINTER Applicable: 8390,8380
	ok failed invalid unit id The module is not initial input parameter may be null pointer

## 13.8. rtk\_ptp\_refTimeAdjust\_set

```
rtk_api_ret_t rtk_ptp_refTimeAdjust_set(rtk_ptp_sys_adjust_t sign,  
rtk_ptp_timeStamp_t timeStamp)
```

Adjust the reference time.

Defined in: ptp.h

<b>Parameters</b>	<i>sign</i> unit id <i>timeStamp</i> significant
<b>Comments</b>	sign=0 for positive adjustment, sign=1 for negative adjustment.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_UNIT_ID invalid unit id RT_ERR_NOT_INIT The module is not initial RT_ERR_INPUT invalid input parameter

---

## 13.9. rtk\_ptp\_refTimeEnable\_set

`rtk_api_ret_t rtk_ptp_refTimeEnable_set(rtk_enable_t enable)`

Set the enable state of reference time of the specified device.

Defined in: ptp.h

**Parameters**      *enable*  
                  status

**Comments**      None

**Return Codes**    RT\_ERR\_OK  
                  RT\_ERR\_FAILED  
                  RT\_ERR\_INPUT

ok  
failed  
invalid input parameter

---

## 13.10. rtk\_ptp\_refTimeEnable\_get

`rtk_api_ret_t rtk_ptp_refTimeEnable_get(rtk_enable_t *pEnable)`

Get the enable state of reference time of the specified device.

Defined in: ptp.h

**Parameters**      *\*pEnable*  
                  status

**Comments**      None

**Return Codes**    RT\_ERR\_OK  
                  RT\_ERR\_FAILED  
                  RT\_ERR\_UNIT\_ID  
                  RT\_ERR\_NOT\_INIT  
                  RT\_ERR\_NULL\_POINTER  
                  Applicable:  
                  8390,8380

ok  
failed  
invalid unit id  
The module is not initial  
input parameter may be null pointer

---

### 13.11. rtk\_ptp\_portEnable\_set

`rtk_api_ret_t rtk_ptp_portEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set PTP status of the specified port.

Defined in: ptp.h

**Parameters**

*port*  
port id

*enable*  
status

**Comments**

None

**Return Codes**

RT\_ERR\_OK  
RT\_ERR\_FAILED  
RT\_ERR\_PORT  
RT\_ERR\_INPUT

ok  
failed  
invalid port id  
invalid input parameter

---

### 13.12. rtk\_ptp\_portEnable\_get

`rtk_api_ret_t rtk_ptp_portEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get PTP status of the specified port.

Defined in: ptp.h

**Parameters**

*port*  
port id  
*\*pEnable*  
status

**Comments**

None

**Return Codes**

RT\_ERR\_OK  
RT\_ERR\_FAILED  
RT\_ERR\_PORT  
RT\_ERR\_NULL\_POINTER

ok  
failed  
invalid port id  
input parameter may be null pointer

---

---

### 13.13. rtk\_ptp\_portTimestamp\_get

```
rtk_api_ret_t rtk_ptp_portTimestamp_get(rtk_port_t port,  
rtk_ptp_msgType_t type, rtk_ptp_info_t *pInfo)
```

Get PTP timestamp according to the PTP identifier on the dedicated port from the specified device.

Defined in: ptp.h

**Parameters**

*port*  
unit id  
*type*  
port id  
*\*pInfo*  
PTP message type

**Comments**

None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_NULL_POINTER	input parameter may be null pointer
Applicable: 8390,8380	

---

### 13.14. rtk\_ptp\_intControl\_set

```
rtk_api_ret_t rtk_ptp_intControl_set(rtk_ptp_intType_t type, rtk_enable_t  
enable)
```

Set PTP interrupt trigger status configuration.

Defined in: ptp.h

**Parameters**

*type*  
Interrupt type.  
*enable*  
Interrupt status.

**Comments**

The API can set PTP interrupt status configuration. The interrupt trigger status is shown in the following: PTP\_INT\_TYPE\_TX\_SYNC = 0, PTP\_INT\_TYPE\_TX\_DELAY\_REQ, PTP\_INT\_TYPE\_TX\_PDELAY\_REQ, PTP\_INT\_TYPE\_TX\_PDELAY\_RESP, PTP\_INT\_TYPE\_RX\_SYNC, PTP\_INT\_TYPE\_RX\_DELAY\_REQ, PTP\_INT\_TYPE\_RX\_PDELAY\_REQ, PTP\_INT\_TYPE\_RX\_PDELAY\_RESP, PTP\_INT\_TYPE\_ALL,

Return Codes	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.
	RT_ERR_ENABLE	Invalid enable input.

### 13.15. rtk\_ptp\_intControl\_get

```
rtk_api_ret_t rtk_ptp_intControl_get(rtk_ptp_intType_t type, rtk_enable_t  
*pEnable)
```

Get PTP interrupt trigger status configuration.

Defined in: ptp.h

Parameters	type	Interrupt type.
	*pEnable	Interrupt status.

Comments	The API can get interrupt status configuration. The interrupt trigger status is shown in the following: PTP_INT_TYPE_TX_SYNC = 0, PTP_INT_TYPE_TX_DELAY_REQ, PTP_INT_TYPE_TX_PDELAY_REQ, PTP_INT_TYPE_TX_PDELAY_RESP, PTP_INT_TYPE_RX_SYNC, PTP_INT_TYPE_RX_DELAY_REQ, PTP_INT_TYPE_RX_PDELAY_REQ, PTP_INT_TYPE_RX_PDELAY_RESP,
----------	---

Return Codes	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

---

### 13.16. rtk\_ptp\_intStatus\_get

`rtk_api_ret_t rtk_ptp_intStatus_get(rtk_ptp_intStatus_t *pStatusMask)`

Get PTP port interrupt trigger status.

Defined in: ptp.h

**Parameters**

`*pStatusMask`  
physical port

**Comments**

The API can get interrupt trigger status when interrupt happened. The interrupt trigger status is shown in the following:

- PORT 0 INT (value[0] (Bit0))
- PORT 1 INT (value[0] (Bit1))
- PORT 2 INT (value[0] (Bit2))
- PORT 3 INT (value[0] (Bit3))
- PORT 4 INT (value[0] (Bit4))

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

### 13.17. rtk\_ptp\_portIntStatus\_set

`rtk_api_ret_t rtk_ptp_portIntStatus_set(rtk_port_t port, rtk_ptp_intStatus_t statusMask)`

Set PTP port interrupt trigger status to clean.

Defined in: ptp.h

**Parameters**

`port`  
physical port

`statusMask`  
Interrupt status bit mask.

**Comments**

The API can clean interrupt trigger status when interrupt happened. The interrupt trigger status is shown in the following:

- PTP\_INT\_TYPE\_TX\_SYNC (value[0] (Bit0))
- PTP\_INT\_TYPE\_TX\_DELAY\_REQ (value[0] (Bit1))
- PTP\_INT\_TYPE\_TX\_PDELAY\_REQ (value[0] (Bit2))

- PTP\_INT\_TYPE\_TX\_PDELAY\_RESP (value[0] (Bit3))
- PTP\_INT\_TYPE\_RX\_SYNC (value[0] (Bit4))
- PTP\_INT\_TYPE\_RX\_DELAY\_REQ (value[0] (Bit5))
- PTP\_INT\_TYPE\_RX\_PDELAY\_REQ (value[0] (Bit6))
- PTP\_INT\_TYPE\_RX\_PDELAY\_RESP (value[0] (Bit7)) The status will be cleared after execute this API.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

### 13.18. rtk\_ptp\_portIntStatus\_get

`rtk_api_ret_t rtk_ptp_portIntStatus_get(rtk_port_t port, rtk_ptp_intStatus_t *pStatusMask)`

Get PTP port interrupt trigger status.

Defined in: ptp.h

#### Parameters

*port*  
physical port

*\*pStatusMask*  
Interrupt status bit mask.

#### Comments

The API can get interrupt trigger status when interrupt happened. The interrupt trigger status is shown in the following:

- PTP\_INT\_TYPE\_TX\_SYNC (value[0] (Bit0))
- PTP\_INT\_TYPE\_TX\_DELAY\_REQ (value[0] (Bit1))
- PTP\_INT\_TYPE\_TX\_PDELAY\_REQ (value[0] (Bit2))
- PTP\_INT\_TYPE\_TX\_PDELAY\_RESP (value[0] (Bit3))
- PTP\_INT\_TYPE\_RX\_SYNC (value[0] (Bit4))
- PTP\_INT\_TYPE\_RX\_DELAY\_REQ (value[0] (Bit5))
- PTP\_INT\_TYPE\_RX\_PDELAY\_REQ (value[0] (Bit6))
- PTP\_INT\_TYPE\_RX\_PDELAY\_RESP (value[0] (Bit7))

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

---

### 13.19. rtk\_ptp\_portTrap\_set

`rtk_api_ret_t rtk_ptp_portTrap_set(rtk_port_t port, rtk_enable_t enable)`

Set PTP packet trap of the specified port.

Defined in: ptp.h

**Parameters**

*port*

port id

*enable*

status

**Comments**

None

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_PORT`

invalid port id

`RT_ERR_INPUT`

invalid input parameter

---

### 13.20. rtk\_ptp\_portTrap\_get

`rtk_api_ret_t rtk_ptp_portTrap_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get PTP packet trap of the specified port.

Defined in: ptp.h

**Parameters**

*port*

port id

*\*pEnable*

status

**Comments**

None

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_PORT`

invalid port id

`RT_ERR_NULL_POINTER`

input parameter may be null pointer

---

## 14. Module qos.h - Unmanaged switch high-level API

Filename: qos.h

**Description** The file includes QoS module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

qos.h - Unmanaged switch high-level API  
rtk\_qos\_init  
rtk\_qos\_priSel\_set  
rtk\_qos\_priSel\_get  
rtk\_qos\_1pPriRemap\_set  
rtk\_qos\_1pPriRemap\_get  
rtk\_qos\_1pRemarkSrcSel\_set  
rtk\_qos\_1pRemarkSrcSel\_get  
rtk\_qos\_dscpPriRemap\_set  
rtk\_qos\_dscpPriRemap\_get  
rtk\_qos\_portPri\_set  
rtk\_qos\_portPri\_get  
rtk\_qos\_queueNum\_set  
rtk\_qos\_queueNum\_get  
rtk\_qos\_priMap\_set  
rtk\_qos\_priMap\_get  
rtk\_qos\_schedulingQueue\_set  
rtk\_qos\_schedulingQueue\_get  
rtk\_qos\_1pRemarkEnable\_set  
rtk\_qos\_1pRemarkEnable\_get  
rtk\_qos\_1pRemark\_set  
rtk\_qos\_1pRemark\_get  
rtk\_qos\_dscpRemarkEnable\_set  
rtk\_qos\_dscpRemarkEnable\_get  
rtk\_qos\_dscpRemark\_set  
rtk\_qos\_dscpRemark\_get  
rtk\_qos\_dscpRemarkSrcSel\_set  
rtk\_qos\_dscpRemarkSrcSel\_get  
rtk\_qos\_dscpRemark2Dscp\_set  
rtk\_qos\_dscpRemark2Dscp\_get  
rtk\_qos\_portPriSelIndex\_set

---

rtk\_qos\_portPriSelIndex\_get  
rtk\_qos\_schedulingType\_set  
rtk\_qos\_schedulingType\_get

---

## 14.1. rtk\_qos\_init

`rtk_api_ret_t rtk_qos_init(rtк_queue_num_t queueNum)`

Configure Qos default settings with queue number assignment to each port.

Defined in: qos.h

**Parameters**

*queueNum*  
Queue number of each port.

**Comments**

This API will initialize related Qos setting with queue number assignment. The queue number is from 1 to 8.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_QUEUE_NUM</code>	Invalid queue number.
<code>RT_ERR_INPUT</code>	Invalid input parameters.

## 14.2. rtk\_qos\_priSel\_set

`rtk_api_ret_t rtk_qos_priSel_set(rtк_qos_priDecTbl_t index,  
rtк_priority_select_t *pPriDec)`

Configure the priority order among different priority mechanism.

Defined in: qos.h

**Parameters**

*index*  
Priority decision table index (0~1)

*\*pPriDec*

Priority assign for port, dscp, 802.1p, cvlan, svlan, acl based priority decision.

**Comments**

ASIC will follow user priority setting of mechanisms to select mapped queue priority for receiving frame. If two priority mechanisms are the same, the ASIC will chose the highest priority from mechanisms to assign queue priority to

receiving frame. The priority sources are:

- PRIDEC\_PORT
- PRIDEC\_ACL
- PRIDEC\_DSCP
- PRIDEC\_1Q
- PRIDEC\_1AD
- PRIDEC\_CVLAN
- PRIDEC\_DA
- PRIDEC\_SA

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_QOS_SEL_PRI_SOURCE	Invalid priority decision source parameter.

### 14.3.rtk\_qos\_priSel\_get

```
rtk_api_ret_t rtk_qos_priSel_get(rtk_qos_priDecTbl_t index,  
                                rtk_priority_select_t *pPriDec)
```

Get the priority order configuration among different priority mechanism.

Defined in: qos.h

*index*

Priority decision table index (0~1)

*\*pPriDec*

Priority assign for port, dscp, 802.1p, cvlan, svlan, acl based priority decision .

#### Comments

ASIC will follow user priority setting of mechanisms to select mapped queue priority for receiving frame. If two priority mechanisms are the same, the ASIC will chose the highest priority from mechanisms to assign queue priority to receiving frame. The priority sources are:

- PRIDEC\_PORT,
- PRIDEC\_ACL,
- PRIDEC\_DSCP,
- PRIDEC\_1Q,
- PRIDEC\_1AD,
- PRIDEC\_CVLAN,
- PRIDEC\_DA,
- PRIDEC\_SA,

<b>Return Codes</b>	RT_ERR_OK	ok
---------------------	-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 14.4. rtk\_qos\_1pPriRemap\_set

`rtk_api_ret_t rtk_qos_1pPriRemap_set(rtk_pri_t dot1p_pri, rtk_pri_t int_pri)`

Configure 1Q priorities mapping to internal absolute priority.

Defined in: qos.h

**Parameters**

*dot1p\_pri*  
802.1p priority value.  
*int\_pri*  
internal priority value.

**Comments**

Priority of 802.1Q assignment for internal asic priority, and it is used for queue usage and packet scheduling.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_VLAN_PRIORITY	Invalid 1p priority.
RT_ERR_QOS_INT_PRIORITY	Invalid priority.

## 14.5. rtk\_qos\_1pPriRemap\_get

`rtk_api_ret_t rtk_qos_1pPriRemap_get(rtk_pri_t dot1p_pri, rtk_pri_t *pInt_pri)`

Get 1Q priorities mapping to internal absolute priority.

Defined in: qos.h

**Parameters**

*dot1p\_pri*  
802.1p priority value .  
*\*pInt\_pri*  
internal priority value.

**Comments**

Priority of 802.1Q assignment for internal asic priority, and it is used for queue usage and packet scheduling.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_VLAN_PRIORITY	Invalid priority.
	RT_ERR_QOS_INT_PRIORITY	Invalid priority.

## 14.6.rtk\_qos\_1pRemarkSrcSel\_set

`rtk_api_ret_t rtk_qos_1pRemarkSrcSel_set(rtk_qos_1pRmkSrc_t type)`

Set remarking source of 802.1p remarking.

Defined in: qos.h

<b>Parameters</b>	<i>type</i>
	remarking source

**Comments** The API can configure 802.1p remark functionality to map original 802.1p value or internal priority to TX DSCP value.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_NOT_INIT	The module is not initial
	RT_ERR_PORT_ID	invalid port id
	RT_ERR_INPUT	invalid input parameter
	RT_ERR_QOS_INT_PRIORITY	

## 14.7.rtk\_qos\_1pRemarkSrcSel\_get

`rtk_api_ret_t rtk_qos_1pRemarkSrcSel_get(rtk_qos_1pRmkSrc_t *pType)`

Get remarking source of 802.1p remarking.

Defined in: qos.h

<b>Parameters</b>	* <i>pType</i>
	remarking source

<b>Comments</b>	None												
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_NOT_INIT</td> <td>The module is not initial</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>invalid port id</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>invalid input parameter</td> </tr> <tr> <td>RT_ERR_NULL_POINTER</td> <td>input parameter may be null pointer</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_NOT_INIT	The module is not initial	RT_ERR_PORT_ID	invalid port id	RT_ERR_INPUT	invalid input parameter	RT_ERR_NULL_POINTER	input parameter may be null pointer
RT_ERR_OK	ok												
RT_ERR_FAILED	failed												
RT_ERR_NOT_INIT	The module is not initial												
RT_ERR_PORT_ID	invalid port id												
RT_ERR_INPUT	invalid input parameter												
RT_ERR_NULL_POINTER	input parameter may be null pointer												

## **14.8. rtk\_qos\_dscpPriRemap\_set**

**rtk\_api\_ret\_t rtk\_qos\_dscpPriRemap\_set(rtk\_dscp\_t dscp, rtk\_pri\_t int\_pri)**

Map dscp value to internal priority.

Defined in: qos.h

*dscp*  
Dscp value of receiving frame

*int\_pri*  
internal priority value.

**Comments** The Differentiated Service Code Point is a selector for router's per-hop behaviors. As a selector, there is no implication that a numerically greater DSCP implies a better network service. As can be seen, the DSCP totally overlaps the old precedence field of TOS. So if values of DSCP are carefully chosen then backward compatibility can be achieved.

<b>Return Codes</b>	<b>Description</b>
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_QOS_DSCP_VALUE	Invalid DSCP value.
RT_ERR_QOS_INT_PRIORITY	Invalid priority.

---

## 14.9. rtk\_qos\_dscpPriRemap\_get

`rtk_api_ret_t rtk_qos_dscpPriRemap_get(rtk_dscp_t dscp, rtk_pri_t *pInt_pri)`

Get dscp value to internal priority.

Defined in: qos.h

**Parameters**

*dscp*  
Dscp value of receiving frame  
*\*pInt\_pri*  
internal priority value.

**Comments**

The Differentiated Service Code Point is a selector for router's per-hop behaviors. As a selector, there is no implication that a numerically greater DSCP implies a better network service. As can be seen, the DSCP totally overlaps the old precedence field of TOS. So if values of DSCP are carefully chosen then backward compatibility can be achieved.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_OOS_DSCP_VALUE	Invalid DSCP value.

---

## 14.10. rtk\_qos\_portPri\_set

`rtk_api_ret_t rtk_qos_portPri_set(rtk_port_t port, rtk_pri_t int_pri)`

Configure priority usage to each port.

Defined in: qos.h

**Parameters**

*port*  
Port id.  
*int\_pri*  
internal priority value.

**Comments**

The API can set priority of port assignments for queue usage and packet scheduling.

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	invalid port number.
RT_ERR_QOS_SEL_PORT_PRI	Invalid port priority.
RT_ERR_QOS_INT_PRIORITY	Invalid priority.

---

## 14.11. rtk\_qos\_portPri\_get

`rtk_api_ret_t rtk_qos_portPri_get(rt_k_port_t port, rt_k_pri_t *pInt_pri)`

Get priority usage to each port.

Defined in: qos.h

### Parameters

*port*

Port id.

*\*pInt\_pri*

internal priority value.

### Comments

The API can get priority of port assignments for queue usage and packet scheduling.

### Return Codes

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_INPUT

Invalid input parameters.

---

## 14.12. rtk\_qos\_queueNum\_set

`rtk_api_ret_t rtk_qos_queueNum_set(rt_k_port_t port, rt_k_queue_num_t queue_num)`

Set output queue number for each port.

Defined in: qos.h

### Parameters

*port*

Port id.

<b>queue_num</b>	Mapping queue number (1~8)										
<b>Comments</b>	The API can set the output queue number of the specified port. The queue number is from 1 to 8.										
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> <tr> <td>RT_ERR_QUEUE_NUM</td><td>Invalid queue number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_QUEUE_NUM	Invalid queue number.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_PORT_ID	Invalid port number.										
RT_ERR_QUEUE_NUM	Invalid queue number.										

---

### 14.13. rtk\_qos\_queueNum\_get

```
rtk_api_ret_t rtk_qos_queueNum_get(rtk_port_t port, rtk_queue_num_t
*pQueue_num)
```

Get output queue number.

Defined in: qos.h

<b>Parameters</b>	<i>port</i>
	Port id.
	<i>*pQueue_num</i>

Mapping queue number

<b>Comments</b>	The API will return the output queue number of the specified port. The queue number is from 1 to 8.
-----------------	---

<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_PORT_ID	Invalid port number.								

---

### 14.14. rtk\_qos\_priMap\_set

```
rtk_api_ret_t rtk_qos_priMap_set(rtk_queue_num_t queue_num,
rtk_qos_pri2queue_t *pPri2qid)
```

Set output queue number for each port.

---

	Defined in: qos.h																
<b>Parameters</b>	<p><i>queue_num</i> Queue number usage.</p> <p><i>*pPri2qid</i> Priority mapping to queue ID.</p>																
<b>Comments</b>	ASIC supports priority mapping to queue with different queue number from 1 to 8. For different queue numbers usage, ASIC supports different internal available queue IDs.																
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> <tr> <td>RT_ERR_QUEUE_NUM</td> <td>Invalid queue number.</td> </tr> <tr> <td>RT_ERR_QUEUE_ID</td> <td>Invalid queue id.</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> <tr> <td>RT_ERR_QOS_INT_PRORITY</td> <td>Invalid priority.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_QUEUE_NUM	Invalid queue number.	RT_ERR_QUEUE_ID	Invalid queue id.	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_QOS_INT_PRORITY	Invalid priority.
RT_ERR_OK	ok																
RT_ERR_FAILED	failed																
RT_ERR_SMI	SMI access error																
RT_ERR_INPUT	Invalid input parameters.																
RT_ERR_QUEUE_NUM	Invalid queue number.																
RT_ERR_QUEUE_ID	Invalid queue id.																
RT_ERR_PORT_ID	Invalid port number.																
RT_ERR_QOS_INT_PRORITY	Invalid priority.																

---

## 14.15. rtk\_qos\_priMap\_get

```
rtk_api_ret_t rtk_qos_priMap_get(rtk_queue_num_t queue_num,
                                  rtk_qos_pri2queue_t *pPri2qid)
```

Get priority to queue ID mapping table parameters.

Defined in: qos.h

	Defined in: qos.h										
<b>Parameters</b>	<p><i>queue_num</i> Queue number usage.</p> <p><i>*pPri2qid</i> Priority mapping to queue ID.</p>										
<b>Comments</b>	The API can return the mapping queue id of the specified priority and queue number. The queue number is from 1 to 8.										
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> <tr> <td>RT_ERR_QUEUE_NUM</td> <td>Invalid queue number.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_QUEUE_NUM	Invalid queue number.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_INPUT	Invalid input parameters.										
RT_ERR_QUEUE_NUM	Invalid queue number.										

---

## 14.16. rtk\_qos\_schedulingQueue\_set

`rtk_api_ret_t rtk_qos_schedulingQueue_set(rtk_port_t port,  
 rtk_qos_queue_weights_t *pQweights)`

Set weight and type of queues in dedicated port.

Defined in: qos.h

**Parameters**

*port*

Port id.

*\*pQweights*

The array of weights for WRR/WFQ queue (0 for STRICT\_PRIORITY queue).

**Comments**

The API can set weight and type, strict priority or weight fair queue (WFQ) for dedicated port for using queues. If queue id is not included in queue usage, then its type and weight setting in dummy for setting. There are priorities as queue id in strict queues. It means strict queue id 5 carrying higher priority than strict queue id 4. The WFQ queue weight is from 1 to 128, and weight 0 is for strict priority queue type.

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_SMI`

SMI access error

`RT_ERR_PORT_ID`

Invalid port number.

`RT_ERR_QOS_QUEUE_WEIGHT`

Invalid queue weight.

---

## 14.17. rtk\_qos\_schedulingQueue\_get

`rtk_api_ret_t rtk_qos_schedulingQueue_get(rtk_port_t port,  
 rtk_qos_queue_weights_t *pQweights)`

Get weight and type of queues in dedicated port.

Defined in: qos.h

**Parameters**

*port*

Port id.

---

<i>*pQweights</i>	The array of weights for WRR/WFQ queue (0 for STRICT_PRIORITY queue).										
<b>Comments</b>	The API can get weight and type, strict priority or weight fair queue (WFQ) for dedicated port for using queues. The WFQ queue weight is from 1 to 128, and weight 0 is for strict priority queue type.										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_INPUT	Invalid input parameters.										
RT_ERR_PORT_ID	Invalid port number.										

---

## 14.18. rtk\_qos\_1pRemarkEnable\_set

<code>rtk_api_ret_t rtk_qos_1pRemarkEnable_set(rtк_port_t port, rtк_enable_t enable)</code>											
Set 1p Remarking state											
Defined in: qos.h											
<i>port</i>	Port id.										
<i>enable</i>	State of per										
<b>Comments</b>	The API can enable or disable 802.1p remarking ability for whole system. The status of 802.1p remark:										
	- DISABLED										
	- ENABLED										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> <tr> <td>RT_ERR_ENABLE</td><td>Invalid enable parameter.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_ENABLE	Invalid enable parameter.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_PORT_ID	Invalid port number.										
RT_ERR_ENABLE	Invalid enable parameter.										

---

## 14.19. rtk\_qos\_1pRemarkEnable\_get

`rtk_api_ret_t rtk_qos_1pRemarkEnable_get(rtк_port_t port, rtк_enable_t *pEnable)`

Get 802.1p remarking ability.

Defined in: qos.h

**Parameters**

*port*  
Port id.

*\*pEnable*  
Status of 802.1p remark.

**Comments**

The API can get 802.1p remarking ability. The status of 802.1p remark:

- DISABLED
- ENABLED

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.

---

## 14.20. rtk\_qos\_1pRemark\_set

`rtk_api_ret_t rtk_qos_1pRemark_set(rtк_pri_t int_pri, rtк_pri_t dot1p_pri)`

Set 802.1p remarking parameter.

Defined in: qos.h

**Parameters**

*int\_pri*  
Internal priority value.

*dot1p\_pri*  
802.1p priority value.

**Comments**

The API can set 802.1p parameters source priority and new priority.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_VLAN_PRIORITY</code>	Invalid 1p priority.

---

RT_ERR_QOS_INT_PRIORITY	Invalid priority.
-------------------------	-------------------

---

## 14.21. rtk\_qos\_1pRemark\_get

`rtk_api_ret_t rtk_qos_1pRemark_get(rtk_pri_t int_pri, rtk_pri_t *pDot1p_pri)`

Get 802.1p remarking parameter.

Defined in: qos.h

**Parameters**

*int\_pri*  
Internal priority value.  
*\*pDot1p\_pri*  
802.1p priority value.

**Comments**

The API can get 802.1p remarking parameters. It would return new priority of ingress priority.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_QOS_INT_PRIORITY	invalid priority

## 14.22. rtk\_qos\_dscpRemarkEnable\_set

`rtk_api_ret_t rtk_qos_dscpRemarkEnable_set(rtk_port_t port, rtk_enable_t enable)`

Set DSCP remarking ability.

Defined in: qos.h

**Parameters**

*port*  
Port id.  
*enable*  
status of DSCP remark.

**Comments**

The API can enable or disable DSCP remarking ability for whole system. The status of DSCP remark:

	- DISABLED - ENABLED	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_QOS_INT_PRIORITY RT_ERR_ENABLE	ok failed SMI access error Invalid port number. Invalid priority. Invalid enable parameter.

---

### 14.23. rtk\_qos\_dscpRemarkEnable\_get

`rtk_api_ret_t rtk_qos_dscpRemarkEnable_get(rtk_port_t port, rtk_enable_t *pEnable)`

Get DSCP remarking ability.

Defined in: qos.h

<b>Parameters</b>	<i>port</i> Port id. <i>*pEnable</i> status of DSCP remarking.
-------------------	---

<b>Comments</b>	The API can get DSCP remarking ability. The status of DSCP remark: - DISABLED - ENABLED
-----------------	---

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID	ok failed SMI access error Invalid port number.
---------------------	--	--

---

### 14.24. rtk\_qos\_dscpRemark\_set

`rtk_api_ret_t rtk_qos_dscpRemark_set(rtk_pri_t int_pri, rtk_dscp_t dscp)`

Set DSCP remarking parameter.

Defined in: qos.h

---

<b>Parameters</b>	<i>int_pri</i> Internal priority value.										
	<i>dscp</i> DSCP value.										
<b>Comments</b>	The API can set DSCP value and mapping priority.										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_QOS_INT_PRIORITY</td><td>Invalid priority.</td></tr> <tr> <td>RT_ERR_QOS_DSCP_VALUE</td><td>Invalid DSCP value.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_QOS_INT_PRIORITY	Invalid priority.	RT_ERR_QOS_DSCP_VALUE	Invalid DSCP value.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_QOS_INT_PRIORITY	Invalid priority.										
RT_ERR_QOS_DSCP_VALUE	Invalid DSCP value.										

---

## 14.25. rtk\_qos\_dscpRemark\_get

`rtk_api_ret_t rtk_qos_dscpRemark_get(rtk_pri_t int_pri, rtk_dscp_t *pDscp)`

Get DSCP remarking parameter.

Defined in: qos.h

<b>Parameters</b>	<i>int_pri</i> Internal priority value.								
	<i>*pDscp</i> DSCP value.								
<b>Comments</b>	The API can get DSCP parameters. It would return DSCP value for mapping priority.								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_QOS_INT_PRIORITY</td><td>Invalid priority.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_QOS_INT_PRIORITY	Invalid priority.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_QOS_INT_PRIORITY	Invalid priority.								

---

## 14.26. rtk\_qos\_dscpRemarkSrcSel\_set

`rtk_api_ret_t rtk_qos_dscpRemarkSrcSel_set(rtk_qos_dscpRmkSrc_t type)`

Set remarking source of DSCP remarking.

Defined in: qos.h

<b>Parameters</b>	<i>type</i> remarking source
<b>Comments</b>	The API can configure DSCP remark functionality to map original DSCP value or internal priority to TX DSCP value.
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_PORT_ID RT_ERR_INPUT RT_ERR_ENABLE
	ok failed The module is not initial invalid port id invalid input parameter

---

#### 14.27. rtk\_qos\_dscpRemarkSrcSel\_get

```
rtk_api_ret_t rtk_qos_dscpRemarkSrcSel_get(rtk_qos_dscpRmkSrc_t  
*pType)
```

Get remarking source of DSCP remarking.

Defined in: qos.h

<b>Parameters</b>	<i>*pType</i> remarking source
<b>Comments</b>	None
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_PORT_ID RT_ERR_INPUT RT_ERR_NULL_POINTER RT_ERR_PORT_ID
	ok failed The module is not initial invalid port id invalid input parameter input parameter may be null pointer

---

---

## 14.28. rtk\_qos\_dscpRemark2Dscp\_set

`rtk_api_ret_t rtk_qos_dscpRemark2Dscp_set(rtk_dscp_t dscp, rtk_dscp_t rmkDscp)`

Set DSCP to remarked DSCP mapping.

Defined in: qos.h

**Parameters**

*dscp*  
    DSCP value  
*rmkDscp*  
    remarked DSCP value

**Comments**

*dscp* parameter can be DSCP value or internal priority according to configuration of API `dal_apollomp_qos_dscpRemarkSrcSel_set()`, because DSCP remark functionality can map original DSCP value or internal priority to TX DSCP value.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_QOS_DSCP_VALUE</code>	Invalid dscp value

---

## 14.29. rtk\_qos\_dscpRemark2Dscp\_get

`rtk_api_ret_t rtk_qos_dscpRemark2Dscp_get(rtk_dscp_t dscp, rtk_dscp_t *pDscp)`

Get DSCP to remarked DSCP mapping.

Defined in: qos.h

**Parameters**

*dscp*  
    DSCP value  
*\*pDscp*  
    remarked DSCP value

**Comments**

None.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_QOS_DSCP_VALUE</code>	Invalid dscp value
<code>RT_ERR_NULL_POINTER</code>	NULL pointer

---

## 14.30. rtk\_qos\_portPriSelIndex\_set

`rtk_api_ret_t rtk_qos_portPriSelIndex_set(rtk_port_t port,  
rtk_qos_priDecTbl_t index)`

Configure priority decision index to each port.

Defined in: qos.h

**Parameters**

*port*  
Port id.

*index*  
priority decision index.

**Comments**

The API can set priority of port assignments for queue usage and packet scheduling.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_PORT_ID</code>	Invalid port number.
<code>RT_ERR_ENTRY_INDEX</code>	Invalid entry index.

---

## 14.31. rtk\_qos\_portPriSelIndex\_get

`rtk_api_ret_t rtk_qos_portPriSelIndex_get(rtk_port_t port,  
rtk_qos_priDecTbl_t *pIndex)`

Get priority decision index from each port.

Defined in: qos.h

**Parameters**

*port*  
Port id.  
*\*pIndex*  
priority decision index.

**Comments**

The API can get priority of port assignments for queue usage and packet scheduling.

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.

---

### 14.32. rtk\_qos\_schedulingType\_set

`rtk_api_ret_t rtk_qos_schedulingType_set(rtk_qos_scheduling_type_t queueType)`

Configure type of scheduling.

Defined in: qos.h

<b>Parameters</b>	<i>queueType</i>
	Scheduling type.

**Comments** The API can set type of scheduling.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_QOS_SCHE_TYPE	Invalid QoS scheduling type.
	RT_ERR_INPUT	Invalid input parameters.

---

### 14.33. rtk\_qos\_schedulingType\_get

`rtk_api_ret_t rtk_qos_schedulingType_get(rtk_qos_scheduling_type_t *pQueueType)`

Configure type of scheduling.

Defined in: qos.h

<b>Parameters</b>	<i>pQueueType</i>
	Scheduling type.

**Comments** The API can set type of scheduling.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed

RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Null pointer.
RT_ERR_INPUT	Invalid input parameters.

---

## 15. Module rate.h - Unmanaged switch high-level API

Filename: rate.h

**Description** The file includes rate module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

rate.h - Unmanaged switch high-level API  
 rtk\_rate\_shareMeter\_set  
 rtk\_rate\_shareMeter\_get  
 rtk\_rate\_shareMeterBucket\_set  
 rtk\_rate\_shareMeterBucket\_get  
 rtk\_rate\_igrBandwidthCtrlRate\_set  
 rtk\_rate\_igrBandwidthCtrlRate\_get  
 rtk\_rate\_egrBandwidthCtrlRate\_set  
 rtk\_rate\_egrBandwidthCtrlRate\_get  
 rtk\_rate\_egrQueueBwCtrlEnable\_set  
 rtk\_rate\_egrQueueBwCtrlEnable\_get  
 rtk\_rate\_egrQueueBwCtrlRate\_set  
 rtk\_rate\_egrQueueBwCtrlRate\_get

---

### 15.1. rtk\_rate\_shareMeter\_set

`rtk_api_ret_t rtk_rate_shareMeter_set(rtk_meter_id_t index,  
 rtk_meter_type_t type, rtk_rate_t rate, rtk_enable_t ifg_include)`

Set meter configuration

---

	Defined in: rate.h												
<b>Parameters</b>	<p><i>index</i> shared meter index</p> <p><i>type</i> shared meter type</p> <p><i>rate</i> rate of share meter</p> <p><i>ifg_include</i> include IFG or not, ENABLE:include DISABLE:exclude</p>												
<b>Comments</b>	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 if type is METER_TYPE_KBPS and the granularity of rate is 8 kbps. The rate unit is packets per second and the range is 1 ~ 0x1FFF if type is METER_TYPE_PPS. The ifg_include parameter is used for rate calculation with/without inter-frame-gap and preamble.												
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_FILTER_METER_ID</td> <td>Invalid meter</td> </tr> <tr> <td>RT_ERR_RATE</td> <td>Invalid rate</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_FILTER_METER_ID	Invalid meter	RT_ERR_RATE	Invalid rate	RT_ERR_INPUT	Invalid input parameters
RT_ERR_OK	ok												
RT_ERR_FAILED	failed												
RT_ERR_SMI	SMI access error												
RT_ERR_FILTER_METER_ID	Invalid meter												
RT_ERR_RATE	Invalid rate												
RT_ERR_INPUT	Invalid input parameters												

---

## 15.2. rtk\_rate\_shareMeter\_get

```
rtk_api_ret_t rtk_rate_shareMeter_get(rtk_meter_id_t index,  
rtk_meter_type_t *pType, rtk_rate_t *pRate, rtk_enable_t *pIfg_include)
```

Get meter configuration

Defined in: rate.h

<b>Parameters</b>	<p><i>index</i> shared meter index</p> <p>*<i>pType</i> Meter Type</p> <p>*<i>pRate</i> pointer of rate of share meter</p> <p>*<i>pIfg_include</i> include IFG or not, ENABLE:include DISABLE:exclude</p>
-------------------	---

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_FILTER_METER_ID	ok failed SMI access error Invalid meter
---------------------	--	---

---

### 15.3. rtk\_rate\_shareMeterBucket\_set

```
rtk_api_ret_t rtk_rate_shareMeterBucket_set(rtk_meter_id_t index,  
                                         rtk_uint32 bucket_size)
```

Set meter Bucket Size

Defined in: rate.h

**Parameters**

<i>index</i>	shared meter index
<i>bucket_size</i>	Bucket Size

**Comments**

The API can set shared meter bucket size.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	Error Input
RT_ERR_SMI	SMI access error
RT_ERR_FILTER_METER_ID	Invalid meter

---

### 15.4. rtk\_rate\_shareMeterBucket\_get

```
rtk_api_ret_t rtk_rate_shareMeterBucket_get(rtk_meter_id_t index,  
                                         rtk_uint32 *pBucket_size)
```

Get meter Bucket Size

Defined in: rate.h

**Parameters**

<i>index</i>	shared meter index
--------------	--------------------

---

<i>*pBucket_size</i>	Bucket Size								
<b>Comments</b>	The API can get shared meter bucket size.								
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_FILTER_METER_ID</td><td>Invalid meter</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_FILTER_METER_ID	Invalid meter
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_FILTER_METER_ID	Invalid meter								

---

## 15.5. rtk\_rate\_igrBandwidthCtrlRate\_set

`rtk_api_ret_t rtk_rate_igrBandwidthCtrlRate_set(rtк_port_t port, rtк_rate_t rate, rtк_enable_t ifg_include, rtк_enable_t fc_enable)`

Set port ingress bandwidth control

Defined in: rate.h

<b>Parameters</b>	<i>port</i> Port id
	<i>rate</i> Rate of share meter
	<i>ifg_include</i> include IFG or not, ENABLE:include DISABLE:exclude
	<i>fc_enable</i> enable flow control or not, ENABLE:use flow control DISABLE:drop

**Comments**  
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.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_ENABLE RT_ERR_INBW_RATE	ok failed SMI access error Invalid port number. Invalid IFG parameter. Invalid ingress rate parameter.
---------------------	---	---

## 15.6.rtk\_rate\_igrBandwidthCtrlRate\_get

```
rtk_api_ret_t rtk_rate_igrBandwidthCtrlRate_get(rtk_port_t port,  
rtk_rate_t *pRate, rtk_enable_t *pIfg_include, rtk_enable_t *pFc_enable)
```

Get port ingress bandwidth control

Defined in: rate.h

### Parameters

<i>port</i>	Port id
<i>*pRate</i>	Rate of share meter
<i>*pIfg_include</i>	Rate's calculation including IFG, ENABLE:include DISABLE:exclude
<i>*pFc_enable</i>	enable flow control or not, ENABLE:use flow control DISABLE:drop

### Comments

The rate unit is 1 kbps and the range is from 8k to 1048568k. The granularity of rate is 8 kbps. The ifg\_include parameter is used for rate calculation with/without inter-frame-gap and preamble.

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number
RT_ERR_INPUT	Invalid input parameters.

## 15.7.rtk\_rate\_egrBandwidthCtrlRate\_set

```
rtk_api_ret_t rtk_rate_egrBandwidthCtrlRate_set(rtk_port_t port,  
rtk_rate_t rate, rtk_enable_t ifg_includ)
```

Set port egress bandwidth control

Defined in: rate.h

### Parameters

<i>port</i>	Port id
<i>rate</i>	Rate of egress bandwidth

---

	<i>ifg_includ</i>	include IFG or not, ENABLE:include DISABLE:exclude
<b>Comments</b>	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.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_INPUT RT_ERR_QOS_EBW_RATE	ok failed SMI access error Invalid port number. Invalid input parameters. Invalid egress bandwidth/rate

---

## 15.8. rtk\_rate\_egressBandwidthCtrlRate\_get

`rtk_api_ret_t rtk_rate_egressBandwidthCtrlRate_get(rtк_port_t port,  
                  rtк_rate_t *pRate, rtк_enable_t *pIfg_include)`

Get port egress bandwidth control

Defined in: rate.h

<b>Parameters</b>	<i>port</i> Port id <i>*pRate</i> Rate of egress bandwidth <i>*pIfg_include</i> Rate's calculation including IFG, ENABLE:include DISABLE:exclude
<b>Comments</b>	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.
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_INPUT

---

## 15.9. rtk\_rate\_egrQueueBwCtrlEnable\_set

`rtk_api_ret_t rtk_rate_egrQueueBwCtrlEnable_set(rtк_port_t port,  
rtк_qid_t queue, rtк_enable_t enable)`

Set enable status of egress bandwidth control on specified queue.

Defined in: rate.h

**Parameters**

*port*  
port id  
*queue*  
queue id  
*enable*  
enable status of egress queue bandwidth control

**Comments**

None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_QUEUE_ID	invalid queue id
RT_ERR_INPUT	invalid input parameter

---

## 15.10. rtk\_rate\_egrQueueBwCtrlEnable\_get

`rtk_api_ret_t rtk_rate_egrQueueBwCtrlEnable_get(rtк_port_t port,  
rtк_qid_t queue, rtк_enable_t *pEnable)`

Get rate of egress bandwidth control on specified queue.

Defined in: rate.h

**Parameters**

*port*  
port id  
*queue*  
queue id  
*\*pEnable*  
shared meter index

**Comments**

None.

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_PORT_ID	invalid port id
	RT_ERR_QUEUE_ID	invalid queue id
	RT_ERR_FILTER_METER_ID	Invalid meter id

---

## 15.11. rtk\_rate\_egressQueueBwCtrlRate\_set

`rtk_api_ret_t rtk_rate_egressQueueBwCtrlRate_set(rt_k_port_t port, rt_k_qid_t queue, rt_k_meter_id_t index)`

Set rate of egress bandwidth control on specified queue.

Defined in: rate.h

<b>Parameters</b>	<i>port</i>	port id
	<i>queue</i>	queue id
	<i>index</i>	shared meter index

**Comments** The actual rate control is set in shared meters. The unit of granularity is 8Kbps.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_PORT_ID	invalid port id
	RT_ERR_QUEUE_ID	invalid queue id
	RT_ERR_FILTER_METER_ID	Invalid meter id

---

## 15.12. rtk\_rate\_egressQueueBwCtrlRate\_get

`rtk_api_ret_t rtk_rate_egressQueueBwCtrlRate_get(rt_k_port_t port, rt_k_qid_t queue, rt_k_meter_id_t *pIndex)`

Get rate of egress bandwidth control on specified queue.

Defined in: rate.h

<b>Parameters</b>	<i>port</i> port id <i>queue</i> queue id <i>*pIndex</i> shared meter index
<b>Comments</b>	The actual rate control is set in shared meters. The unit of granularity is 8Kbps.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_PORT_ID invalid port id RT_ERR_QUEUE_ID invalid queue id RT_ERR_FILTER_METER_ID Invalid meter id

---

## 16. Module rldp.h - Declaration of RLDP and RLPP API

### Description

Filename: rldp.h

The file have include the following module and sub-modules 1) RLDP and RLPP configuration and status

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

### List of Symbols

Here is a list of all functions and variables in this module

rldp.h - Declaration of RLDP and RLPP API  
 rtk\_api\_ret\_t  
 rtk\_rldp\_config\_set  
 rtk\_rldp\_config\_get  
 rtk\_rldp\_portConfig\_set  
 rtk\_rldp\_portConfig\_get  
 rtk\_rldp\_status\_get  
 rtk\_rldp\_portStatus\_get  
 rtk\_rldp\_portStatus\_set  
 rtk\_rldp\_portLoopPair\_get

---

## 16.1. rtk\_rldp\_config\_set

`rtk_api_ret_t rtk_rldp_config_set(rtk_rldp_config_t *pConfig)`

Set RLDP module configuration

Defined in: rldp.h

**Parameters**      `*pConfig`  
                      configuration structure of RLDP

**Comments**      None

**Return Codes**     `RT_ERR_OK`      ok  
                      `RT_ERR_FAILED`    failed  
                      `RT_ERR_INPUT`  
                      `RT_ERR_NULL_POINTER`

---

## 16.2. rtk\_rldp\_config\_get

`rtk_api_ret_t rtk_rldp_config_get(rtk_rldp_config_t *pConfig)`

Get RLDP module configuration

Defined in: rldp.h

**Parameters**      `*pConfig`  
                      configuration structure of RLDP

**Comments**      None

**Return Codes**     `RT_ERR_OK`      ok  
                      `RT_ERR_FAILED`    failed  
                      `RT_ERR_INPUT`  
                      `RT_ERR_NULL_POINTER`

---

## 16.3.rtk\_rldp\_portConfig\_set

`rtk_api_ret_t rtk_rldp_portConfig_set(rtk_port_t port,  
rtk_rldp_portConfig_t *pPortConfig)`

Set per port RLDP module configuration

Defined in: rldp.h

**Parameters**

*port*  
port number to be configured  
*\*pPortConfig*  
per port configuration structure of RLDP

**Comments**

None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	
RT_ERR_NULL_POINTER	

---

## 16.4.rtk\_rldp\_portConfig\_get

`rtk_api_ret_t rtk_rldp_portConfig_get(rtk_port_t port,  
rtk_rldp_portConfig_t *pPortConfig)`

Get per port RLDP module configuration

Defined in: rldp.h

**Parameters**

*port*  
port number to be get  
*\*pPortConfig*  
per port configuration structure of RLDP

**Comments**

None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	
RT_ERR_NULL_POINTER	

---

## 16.5. rtk\_rldp\_status\_get

`rtk_api_ret_t rtk_rldp_status_get(rtk_rldp_status_t *pStatus)`

Get RLDP module status

Defined in: rldp.h

**Parameters**      *\*pStatus*  
                          status structure of RLDP

**Comments**      None

**Return Codes**      RT\_ERR\_OK                            ok  
                          RT\_ERR\_FAILED                 failed  
                          RT\_ERR\_NULL\_POINTER

---

## 16.6. rtk\_rldp\_portStatus\_get

`rtk_api_ret_t rtk_rldp_portStatus_get(rtk_port_t port,  
  rtk_rldp_portStatus_t *pPortStatus)`

Get RLDP module status

Defined in: rldp.h

**Parameters**      *port*  
                          port number to be get  
                          *\*pPortStatus*  
                          per port status structure of RLDP

**Comments**      None

**Return Codes**      RT\_ERR\_OK                            ok  
                          RT\_ERR\_FAILED                 failed  
                          RT\_ERR\_INPUT  
                          RT\_ERR\_NULL\_POINTER

---

## 16.7.rtk\_rldp\_portStatus\_set

`rtk_api_ret_t rtk_rldp_portStatus_set(rtk_port_t port, rtk_rldp_portStatus_t *pPortStatus)`

Clear RLDP module status

Defined in: rldp.h

**Parameters**

*port*  
port number to be clear

*\*pPortStatus*  
per port status structure of RLDP

**Comments**

Clear operation effect loop\_enter and loop\_leave only, other field in the structure are don't care

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	
RT_ERR_NULL_POINTER	

---

## 16.8.rtk\_rldp\_portLoopPair\_get

`rtk_api_ret_t rtk_rldp_portLoopPair_get(rtk_port_t port, rtk_portmask_t *pPortmask)`

Get RLDP port loop pairs

Defined in: rldp.h

**Parameters**

*port*  
port number to be get  
*\*pPortmask*  
per port related loop ports

**Comments**

None

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	
RT_ERR_NULL_POINTER	

---

## 17. Module rtk\_switch.h - Definition function prototype of RTK switch API.

Filename: rtk\_switch.h

**Description** Function prototype definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

rtk\_switch.h - Definition function prototype of RTK switch API.  
rtk\_switch\_probe  
rtk\_switch\_initialState\_set  
rtk\_switch\_initialState\_get  
rtk\_switch\_logicalPortCheck  
rtk\_switch\_isUtpPort  
rtk\_switch\_isExtPort  
rtk\_switch\_isHsgPort  
rtk\_switch\_isComboPort  
rtk\_switch\_ComboPort\_get  
rtk\_switch\_port\_L2P\_get  
rtk\_switch\_port\_P2L\_get  
rtk\_switch\_isPortMaskValid  
rtk\_switch\_isPortMaskUtp  
rtk\_switch\_isPortMaskExt  
rtk\_switch\_portmask\_L2P\_get  
rtk\_switch\_portmask\_P2L\_get  
rtk\_switch\_phyPortMask\_get  
rtk\_switch\_logPortMask\_get  
rtk\_switch\_init  
rtk\_switch\_portMaxPkLen\_set  
rtk\_switch\_portMaxPktLen\_get  
rtk\_switch\_maxPktLenCfg\_set  
rtk\_switch\_maxPktLenCfg\_get  
rtk\_switch\_greenEthernet\_set  
rtk\_switch\_greenEthernet\_get  
rtk\_switch\_maxLogicalPort\_get

---

## 17.1.rtk\_switch\_probe

`rtk_api_ret_t rtk_switch_probe(switch_chip_t *pSwitchChip)`

Probe switch

Defined in: rtk\_switch.h

**Parameters** `*pSwitchChip`

**Comments**

**Return Codes**      `RT_ERR_OK`                        ok  
                        `RT_ERR_FAILED`               failed

---

## 17.2.rtk\_switch\_initialState\_set

`rtk_api_ret_t rtk_switch_initialState_set(init_state_t state)`

Set initial status

Defined in: rtk\_switch.h

**Parameters**      `state`  
                        Initial state;

**Comments**

**Return Codes**      `RT_ERR_OK`                        ok  
                        `RT_ERR_FAILED`               failed

---

## 17.3.rtk\_switch\_initialState\_get

`init_state_t rtk_switch_initialState_get( void)`

Get initial status

Defined in: rtk\_switch.h

**Parameters**

---

*void*

**Comments**

<b>Return Codes</b>	INIT_COMPLETED INIT_NOT_COMPLETED	Initialized Uninitialized
---------------------	--------------------------------------	------------------------------

---

## 17.4. rtk\_switch\_logicalPortCheck

**rtk\_api\_ret\_t rtk\_switch\_logicalPortCheck(rtk\_port\_t logicalPort)**

Check logical port ID.

Defined in: rtk\_switch.h

**Parameters**

*logicalPort*  
logical port ID

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT	ok failed Not Initialize
---------------------	---	--------------------------------

---

## 17.5. rtk\_switch\_isUtpPort

**rtk\_api\_ret\_t rtk\_switch\_isUtpPort(rtk\_port\_t logicalPort)**

Check is logical port a UTP port

Defined in: rtk\_switch.h

**Parameters**

*logicalPort*  
logical port ID

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT	ok failed Not Initialize
---------------------	---	--------------------------------

---

## 17.6.rtk\_switch\_isExtPort

`rtk_api_ret_t rtk_switch_isExtPort(rtk_port_t logicalPort)`

Check is logical port a Extension port

Defined in: rtk\_switch.h

**Parameters**

*logicalPort*

logical port ID

**Comments**

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_NOT\_INIT

Not Initialize

---

## 17.7.rtk\_switch\_isHsgPort

`rtk_api_ret_t rtk_switch_isHsgPort(rtk_port_t logicalPort)`

Check is logical port a HSG port

Defined in: rtk\_switch.h

**Parameters**

*logicalPort*

logical port ID

**Comments**

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_NOT\_INIT

Not Initialize

---

## 17.8.rtk\_switch\_isComboPort

`rtk_api_ret_t rtk_switch_isComboPort(rtk_port_t logicalPort)`

Check is logical port a Combo port

---

	Defined in: rtk_switch.h
<b>Parameters</b>	<i>logicalPort</i> logical port ID
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT
	ok failed Not Initialize

---

## 17.9. rtk\_switch\_ComboPort\_get

`rtk_uint32 rtk_switch_ComboPort_get( void)`

Get Combo port ID

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>void</i>
<b>Comments</b>	
<b>Return Codes</b>	

<b>Parameters</b>	<i>PortIDofcomboport</i>
	Port ID is a combo port

## 17.10. rtk\_switch\_port\_L2P\_get

`rtk_uint32 rtk_switch_port_L2P_get(rtk_port_t logicalPort)`

Get physical port ID

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>logicalPort</i> logical port ID
<b>Comments</b>	
<b>Return Codes</b>	

<b>Parameters</b>	<i>PhysicalportID</i>
	Port ID is a combo port

---

## 17.11. rtk\_switch\_port\_P2L\_get

`rtk_port_t rtk_switch_port_P2L_get(rtк_uint32 physicalPort)`

Get logical port ID

Defined in: rtk\_switch.h

**Parameters**

*physicalPort*

physical port ID

**Comments**

**Return Codes**

logicalportID

Port ID is a combo port

---

## 17.12. rtk\_switch\_isPortMaskValid

`rtк_api_ret_t rtk_switch_isPortMaskValid(rtк_portmask_t *pPmask)`

Check portmask is valid or not

Defined in: rtk\_switch.h

**Parameters**

*\*pPmask*

logical port mask

**Comments**

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_NOT\_INIT

Not Initialize

RT\_ERR\_NULL\_POINTER

Null pointer

---

## 17.13. rtk\_switch\_isPortMaskUtp

`rtк_api_ret_t rtk_switch_isPortMaskUtp(rtк_portmask_t *pPmask)`

Check all ports in portmask are only UTP port

Defined in: rtk\_switch.h

---

<b>Parameters</b>	<i>*pPmask</i> logical port mask
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_NULL_POINTER
	ok failed Not Initialize Null pointer

---

## 17.14. rtk\_switch\_isPortMaskExt

`rtk_api_ret_t rtk_switch_isPortMaskExt(rtk_portmask_t *pPmask)`

Check all ports in portmask are only EXT port

Defined in: `rtk_switch.h`

<b>Parameters</b>	<i>*pPmask</i> logical port mask
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_NULL_POINTER
	ok failed Not Initialize Null pointer

---

## 17.15. rtk\_switch\_portmask\_L2P\_get

`rtk_api_ret_t rtk_switch_portmask_L2P_get(rtk_portmask_t *pLogicalPmask, rtk_uint32 *pPhysicalPortmask)`

Get physical portmask from logical portmask

Defined in: `rtk_switch.h`

<b>Parameters</b>	<i>*pLogicalPmask</i> logical port mask
	<i>*pPhysicalPortmask</i> physical port mask

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_NOT_INIT RT_ERR_NULL_POINTER RT_ERR_PORT_MASK	ok Not Initialize Null pointer Error port mask
---------------------	---	---

---

## 17.16. rtk\_switch\_portmask\_P2L\_get

`rtk_api_ret_t rtk_switch_portmask_P2L_get(rtk_uint32 physicalPortmask,  
rtk_portmask_t *pLogicalPmask)`

Get logical portmask from physical portmask

Defined in: rtk\_switch.h

**Parameters**

*physicalPortmask*  
physical port mask  
*\*pLogicalPmask*  
logical port mask

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_NOT_INIT RT_ERR_NULL_POINTER RT_ERR_PORT_MASK	ok Not Initialize Null pointer Error port mask
---------------------	---	---

---

## 17.17. rtk\_switch\_phyPortMask\_get

`rtk_uint32 rtk_switch_phyPortMask_get( void)`

Get physical portmask

Defined in: rtk\_switch.h

**Parameters**

*void*

**Comments****Return Codes**

---

0x00	Not Initialize
Othervalue	Physical port mask

---

## 17.18. rtk\_switch\_logPortMask\_get

`rtk_api_ret_t rtk_switch_logPortMask_get(rtк_portmask_t *pPortmask)`

Get Logical portmask

Defined in: rtk\_switch.h

**Parameters**

`*pPortmask`  
physical port mask

**Comments**

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_NOT_INIT</code>	Not Initialize
<code>RT_ERR_NULL_POINTER</code>	Null pointer

---

## 17.19. rtk\_switch\_init

`rtk_api_ret_t rtk_switch_init( void)`

Set chip to default configuration environment

Defined in: rtk\_switch.h

**Parameters**

`void`

**Comments**

The API can set chip registers to default configuration for different release chip model.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 17.20. rtk\_switch\_portMaxPktLen\_set

```
rtk_api_ret_t rtk_switch_portMaxPktLen_set(rtk_port_t port,  
                                         rtk_switch_maxPktLen_linkSpeed_t speed, rtk_uint32 cfgId)
```

Set Max packet length

Defined in: rtk\_switch.h

**Parameters**

*port*  
Port ID

*speed*  
Speed

*cfgId*  
Configuration ID

**Comments**

The API can set chip registers to default configuration for different release chip model.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Error Input

---

## 17.21. rtk\_switch\_portMaxPktLen\_get

```
rtk_api_ret_t rtk_switch_portMaxPktLen_get(rtk_port_t port,  
                                         rtk_switch_maxPktLen_linkSpeed_t *speed, rtk_uint32 *pCfgId)
```

Get Max packet length

Defined in: rtk\_switch.h

**Parameters**

*port*  
Port ID

*speed*  
Speed

*\*pCfgId*  
Configuration ID

**Comments**

The API can set chip registers to default configuration for different release chip model.

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

---

## 17.22. rtk\_switch\_maxPktLenCfg\_set

`rtk_api_ret_t rtk_switch_maxPktLenCfg_set(rtk_uint32 cfgId, rtk_uint32 pktLen)`

Set Max packet length configuration

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>cfgId</i>	Configuration ID
	<i>pktLen</i>	Max packet length

**Comments**  
The API can set chip registers to default configuration for different release chip model.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

---

## 17.23. rtk\_switch\_maxPktLenCfg\_get

`rtk_api_ret_t rtk_switch_maxPktLenCfg_get(rtk_uint32 cfgId, rtk_uint32 *pPktLen)`

Get Max packet length configuration

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>cfgId</i>	Configuration ID
	<i>*pPktLen</i>	Max packet length

<b>Comments</b>	The API can set chip registers to default configuration for different release chip model.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

---

## 17.24. rtk\_switch\_greenEthernet\_set

`rtk_api_ret_t rtk_switch_greenEthernet_set(rtк_enable_t enable)`

Set all Ports Green Ethernet state.

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>enable</i>	Green Ethernet state.
-------------------	---------------	-----------------------

<b>Comments</b>	This API can set all Ports Green Ethernet state. The configuration is as following: - DISABLE - ENABLE	
-----------------	--	--

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_ENABLE	Invalid enable input.

---

## 17.25. rtk\_switch\_greenEthernet\_get

`rtk_api_ret_t rtk_switch_greenEthernet_get(rtк_enable_t *pEnable)`

Get all Ports Green Ethernet state.

Defined in: rtk\_switch.h

<b>Parameters</b>	<i>*pEnable</i>	Green Ethernet state.
-------------------	-----------------	-----------------------

<b>Comments</b>	This API can get Green Ethernet state.	
-----------------	--	--

<b>Return Codes</b>	RT_ERR_OK	ok
---------------------	-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 17.26. rtk\_switch\_maxLogicalPort\_get

`rtk_port_t rtk_switch_maxLogicalPort_get( void)`

Get Max logical port ID

Defined in: `rtk_switch.h`

**Parameters** `void`

**Comments** This API can get max logical port

**Return Codes** `Maxlogicalport` `OK`

---

## 18. Module stat.h - Unmanaged switch high-level API

Filename: `stat.h`

**Description** The file includes MIB module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

**List of Symbols**

Here is a list of all functions and variables in this module

stat.h - Unmanaged switch high-level API  
`rtk_stat_global_reset`  
`rtk_stat_port_reset`  
`rtk_stat_queueManage_reset`  
`rtk_stat_global_get`  
`rtk_stat_global_getAll`  
`rtk_stat_port_get`  
`rtk_stat_port_getAll`  
`rtk_stat_logging_counterCfg_set`  
`rtk_stat_logging_counterCfg_get`

`rtk_stat_logging_counter_reset`  
`rtk_stat_logging_counter_get`  
`rtk_stat_lengthMode_set`  
`rtk_stat_lengthMode_get`

---

## 18.1.rtk\_stat\_global\_reset

`rtk_api_ret_t rtk_stat_global_reset( void)`

Reset global MIB counter.

Defined in: stat.h

**Parameters**

*void*

**Comments**

Reset MIB counter of ports. API will use global reset while port mask is all-ports.

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_SMI`

SMI access error

---

## 18.2.rtk\_stat\_port\_reset

`rtk_api_ret_t rtk_stat_port_reset(rtk_port_t port)`

Reset per port MIB counter by port.

Defined in: stat.h

**Parameters**

*port*

port id.

**Comments**

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_SMI`

SMI access error

---

---

### 18.3. rtk\_stat\_queueManage\_reset

`rtk_api_ret_t rtk_stat_queueManage_reset( void)`

Reset queue manage MIB counter.

Defined in: stat.h

**Parameters** `void`

**Comments**

<b>Return Codes</b>	<code>RT_ERR_OK</code>	ok
	<code>RT_ERR_FAILED</code>	failed
	<code>RT_ERR_SMI</code>	SMI access error

---

### 18.4. rtk\_stat\_global\_get

`rtk_api_ret_t rtk_stat_global_get(rtк_stat_global_type_t cntr_idx,  
rtк_stat_counter_t *pCntr)`

Get global MIB counter

Defined in: stat.h

**Parameters** `cntr_idx`  
global counter index.

`*pCntr`  
global counter value.

**Comments** Get global MIB counter by index definition.

<b>Return Codes</b>	<code>RT_ERR_OK</code>	ok
	<code>RT_ERR_FAILED</code>	failed
	<code>RT_ERR_SMI</code>	SMI access error
	<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 18.5.rtk\_stat\_global\_getAll

`rtk_api_ret_t rtk_stat_global_getAll(rtk_stat_global_cntr_t *pGlobal_cntrs)`

Get all global MIB counter

Defined in: stat.h

**Parameters** `*pGlobal_cntrs`  
global counter structure.

**Comments** Get all global MIB counter by index definition.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameters.

---

## 18.6.rtk\_stat\_port\_get

`rtk_api_ret_t rtk_stat_port_get(rtk_port_t port, rtk_stat_port_type_t cntr_idx, rtk_stat_counter_t *pCntr)`

Get per port MIB counter by index

Defined in: stat.h

**Parameters** `port`  
port id.  
`cntr_idx`  
port counter index.  
`*pCntr`  
MIB retrieved counter.

**Comments** Get per port MIB counter by index definition.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

## 18.7.rtk\_stat\_port\_getAll

```
rtk_api_ret_t rtk_stat_port_getAll(rtk_port_t port, rtk_stat_port_cntr_t  
*pPort_cntrs)
```

Get all counters of one specified port in the specified device.

Defined in: stat.h

**Parameters**

*port*  
port id.  
*\*pPort\_cntrs*  
buffer pointer of counter value.

**Comments**

Get all MIB counters of one port.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 18.8.rtk\_stat\_logging\_counterCfg\_set

```
rtk_api_ret_t rtk_stat_logging_counterCfg_set(rtk_uint32 idx,  
rtk_logging_counter_mode_t mode, rtk_logging_counter_type_t type)
```

Set the type and mode of Logging Counter

Defined in: stat.h

**Parameters**

*idx*  
The index of Logging Counter. Should be even number only.(0,2,4,6,8.....30)  
*mode*  
32 bits or 64 bits mode  
*type*  
Packet counter or byte counter

**Comments**

Set the type and mode of Logging Counter.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_OUT_OF_RANGE	Out of range.
RT_ERR_FAILED	failed

RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 18.9. rtk\_stat\_logging\_counterCfg\_get

`rtk_api_ret_t rtk_stat_logging_counterCfg_get(rtk_uint32 idx,  
rtk_logging_counter_mode_t *pMode, rtk_logging_counter_type_t *pType)`

Get the type and mode of Logging Counter

Defined in: stat.h

**Parameters**

*idx*  
The index of Logging Counter. Should be even number only.(0,2,4,6,8.....30)

*\*pMode*  
32 bits or 64 bits mode

*\*pType*  
Packet counter or byte counter

**Comments**

Get the type and mode of Logging Counter.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_OUT_OF_RANGE	Out of range.
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	NULL Pointer
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

## 18.10. rtk\_stat\_logging\_counter\_reset

`rtk_api_ret_t rtk_stat_logging_counter_reset(rtk_uint32 idx)`

Reset Logging Counter

Defined in: stat.h

**Parameters**

*idx*  
The index of Logging Counter. (0~31)

**Comments**

Reset Logging Counter.

---

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_OUT_OF_RANGE	Out of range.
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error

---

## 18.11. rtk\_stat\_logging\_counter\_get

`rtk_api_ret_t rtk_stat_logging_counter_get(rtk_uint32 idx, rtk_uint32 *pCnt)`

Get Logging Counter

Defined in: stat.h

**Parameters**

*idx*  
The index of Logging Counter. (0~31)

*\*pCnt*  
Logging counter value

**Comments**

Get Logging Counter.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_OUT_OF_RANGE	Out of range.
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 18.12. rtk\_stat\_lengthMode\_set

`rtk_api_ret_t rtk_stat_lengthMode_set(rtk_stat_lengthMode_t txMode,  
rtk_stat_lengthMode_t rxMode)`

Set Legnth mode.

Defined in: stat.h

**Parameters**

*txMode*  
The length counting mode

*rxMode*  
The length counting mode

**Comments**

<b>Return Codes</b>	RT_ERR_OK RT_ERR_INPUT RT_ERR_FAILED RT_ERR_SMI	ok Out of range. failed SMI access error
---------------------	--	---

---

### 18.13. rtk\_stat\_lengthMode\_get

`rtk_api_ret_t rtk_stat_lengthMode_get(rtk_stat_lengthMode_t *pTxMode,  
rtk_stat_lengthMode_t *pRxMode)`

Get Legnth mode.

Defined in: stat.h

<b>Parameters</b>	* <i>pTxMode</i> The length counting mode  * <i>pRxMode</i> The length counting mode
-------------------	--

#### Comments

<b>Return Codes</b>	RT_ERR_OK RT_ERR_INPUT RT_ERR_FAILED RT_ERR_SMI	ok Out of range. failed SMI access error
---------------------	--	---

---

## 19. Module storm.h - Unmanaged switch high-level API

Filename: storm.h

**Description** The file includes Storm module high-layer API defination

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

storm.h - Unmanaged switch high-level API

---

```
rtk_rate_stormControlMeterIdx_set  
rtk_rate_stormControlMeterIdx_get  
rtk_rate_stormControlPortEnable_set  
rtk_rate_stormControlPortEnable_get  
rtk_storm_bypass_set  
rtk_storm_bypass_get  
rtk_rate_stormControlExtPortmask_set  
rtk_rate_stormControlExtPortmask_get  
rtk_rate_stormControlExtEnable_set  
rtk_rate_stormControlExtEnable_get  
rtk_rate_stormControlExtMeterIdx_set  
rtk_rate_stormControlExtMeterIdx_get
```

---

## 19.1. rtk\_rate\_stormControlMeterIdx\_set

`rtk_api_ret_t rtk_rate_stormControlMeterIdx_set(rtк_port_t port,  
rtк_rate_storm_group_t stormType, rtк_uint32 index)`

Set the storm control meter index.

Defined in: storm.h

### Parameters

*port*  
port id  
*stormType*  
storm group type  
*index*  
storm control meter index.

### Comments

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_FILTER_METER_ID	Invalid meter

## 19.2. rtk\_rate\_stormControlMeterIdx\_get

`rtk_api_ret_t rtk_rate_stormControlMeterIdx_get(rtк_port_t port,  
rtк_rate_storm_group_t stormType, rtк_uint32 *pIndex)`

Get the storm control meter index.

Defined in: storm.h

**Parameters**

*port*  
port id  
*stormType*  
storm group type  
*\*pIndex*  
storm control meter index.

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_FILTER_METER_ID	Invalid meter

### 19.3.rtk\_rate\_stormControlPortEnable\_set

`rtk_api_ret_t rtk_rate_stormControlPortEnable_set(rtк_port_t port,  
rtк_rate_storm_group_t stormType, rtк_enable_t enable)`

Set enable status of storm control on specified port.

Defined in: storm.h

**Parameters**

*port*  
port id  
*stormType*  
storm group type  
*enable*  
enable status of storm control

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NOT_INIT	The module is not initial
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter

---

---

## 19.4. rtk\_rate\_stormControlPortEnable\_get

```
rtk_api_ret_t rtk_rate_stormControlPortEnable_get(rtk_port_t port,  
rtk_rate_storm_group_t stormType, rtk_enable_t *pEnable)
```

Set enable status of storm control on specified port.

Defined in: storm.h

**Parameters**

*port*  
port id  
*stormType*  
storm group type  
*\*pEnable*  
enable status of storm control

**Comments****Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NOT_INIT	The module is not initial
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter

---

## 19.5. rtk\_storm\_bypass\_set

```
rtk_api_ret_t rtk_storm_bypass_set(rtk_storm_bypass_t type, rtk_enable_t  
enable)
```

Set bypass storm filter control configuration.

Defined in: storm.h

**Parameters**

*type*  
Bypass storm filter control type.  
*enable*  
Bypass status.

**Comments**

This API can set per-port bypass storm filter control frame type including RMA and igmp. The bypass frame type is as following:  
- BYPASS\_BRG\_GROUP,  
- BYPASS\_FD\_PAUSE,

- BYPASS\_SP\_MCAST,
- BYPASS\_1X\_PAE,
- BYPASS\_UNDEF\_BRG\_04,
- BYPASS\_UNDEF\_BRG\_05,
- BYPASS\_UNDEF\_BRG\_06,
- BYPASS\_UNDEF\_BRG\_07,
- BYPASS\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- BYPASS\_UNDEF\_BRG\_09,
- BYPASS\_UNDEF\_BRG\_0A,
- BYPASS\_UNDEF\_BRG\_0B,
- BYPASS\_UNDEF\_BRG\_0C,
- BYPASS\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- BYPASS\_8021AB,
- BYPASS\_UNDEF\_BRG\_0F,
- BYPASS\_BRG\_MNGEMENT,
- BYPASS\_UNDEFINED\_11,
- BYPASS\_UNDEFINED\_12,
- BYPASS\_UNDEFINED\_13,
- BYPASS\_UNDEFINED\_14,
- BYPASS\_UNDEFINED\_15,
- BYPASS\_UNDEFINED\_16,
- BYPASS\_UNDEFINED\_17,
- BYPASS\_UNDEFINED\_18,
- BYPASS\_UNDEFINED\_19,
- BYPASS\_UNDEFINED\_1A,
- BYPASS\_UNDEFINED\_1B,
- BYPASS\_UNDEFINED\_1C,
- BYPASS\_UNDEFINED\_1D,
- BYPASS\_UNDEFINED\_1E,
- BYPASS\_UNDEFINED\_1F,
- BYPASS\_GMRP,
- BYPASS\_GVRP,
- BYPASS\_UNDEF\_GARP\_22,
- BYPASS\_UNDEF\_GARP\_23,
- BYPASS\_UNDEF\_GARP\_24,
- BYPASS\_UNDEF\_GARP\_25,
- BYPASS\_UNDEF\_GARP\_26,
- BYPASS\_UNDEF\_GARP\_27,
- BYPASS\_UNDEF\_GARP\_28,
- BYPASS\_UNDEF\_GARP\_29,
- BYPASS\_UNDEF\_GARP\_2A,
- BYPASS\_UNDEF\_GARP\_2B,
- BYPASS\_UNDEF\_GARP\_2C,
- BYPASS\_UNDEF\_GARP\_2D,
- BYPASS\_UNDEF\_GARP\_2E,

- 
- BYPASS\_UNDEF\_GARP\_2F,
  - BYPASS\_IGMP.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.
	RT_ERR_ENABLE	Invalid IFG parameter

---

## 19.6. rtk\_storm\_bypass\_get

```
rtk_api_ret_t rtk_storm_bypass_get(rtk_storm_bypass_t type, rtk_enable_t  
*pEnable)
```

Get bypass storm filter control configuration.

Defined in: storm.h

### Parameters

*type*  
Bypass storm filter control type.

*\*pEnable*  
Bypass status.

### Comments

This API can get per-port bypass storm filter control frame type including RMA and igmp. The bypass frame type is as following:

- BYPASS\_BRG\_GROUP,
- BYPASS\_FD\_PAUSE,
- BYPASS\_SP\_MCAST,
- BYPASS\_1X\_PAE,
- BYPASS\_UNDEF\_BRG\_04,
- BYPASS\_UNDEF\_BRG\_05,
- BYPASS\_UNDEF\_BRG\_06,
- BYPASS\_UNDEF\_BRG\_07,
- BYPASS\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- BYPASS\_UNDEF\_BRG\_09,
- BYPASS\_UNDEF\_BRG\_0A,
- BYPASS\_UNDEF\_BRG\_0B,
- BYPASS\_UNDEF\_BRG\_0C,
- BYPASS\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- BYPASS\_8021AB,
- BYPASS\_UNDEF\_BRG\_0F,
- BYPASS\_BRG\_MNGEMENT,
- BYPASS\_UNDEFINED\_11,

- BYPASS\_UNDEFINED\_12,  
- BYPASS\_UNDEFINED\_13,  
- BYPASS\_UNDEFINED\_14,  
- BYPASS\_UNDEFINED\_15,  
- BYPASS\_UNDEFINED\_16,  
- BYPASS\_UNDEFINED\_17,  
- BYPASS\_UNDEFINED\_18,  
- BYPASS\_UNDEFINED\_19,  
- BYPASS\_UNDEFINED\_1A,  
- BYPASS\_UNDEFINED\_1B,  
- BYPASS\_UNDEFINED\_1C,  
- BYPASS\_UNDEFINED\_1D,  
- BYPASS\_UNDEFINED\_1E,  
- BYPASS\_UNDEFINED\_1F,  
- BYPASS\_CMRP,  
- BYPASS\_GVRP,  
- BYPASS\_UNDEF\_GARP\_22,  
- BYPASS\_UNDEF\_GARP\_23,  
- BYPASS\_UNDEF\_GARP\_24,  
- BYPASS\_UNDEF\_GARP\_25,  
- BYPASS\_UNDEF\_GARP\_26,  
- BYPASS\_UNDEF\_GARP\_27,  
- BYPASS\_UNDEF\_GARP\_28,  
- BYPASS\_UNDEF\_GARP\_29,  
- BYPASS\_UNDEF\_GARP\_2A,  
- BYPASS\_UNDEF\_GARP\_2B,  
- BYPASS\_UNDEF\_GARP\_2C,  
- BYPASS\_UNDEF\_GARP\_2D,  
- BYPASS\_UNDEF\_GARP\_2E,  
- BYPASS\_UNDEF\_GARP\_2F,  
- BYPASS\_IGMP.

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

## 19.7.rtk\_rate\_stormControlExtPortmask\_set

`rtk_api_ret_t rtk_rate_stormControlExtPortmask_set(rtk_portmask_t  
*pPortmask)`

---

	Set externsion storm control port mask
	Defined in: storm.h
<b>Parameters</b>	* <i>pPortmask</i> port mask
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_INPUT
	ok failed The module is not initial invalid input parameter

---

### 19.8. rtk\_rate\_stormControlExtPortmask\_get

`rtk_api_ret_t rtk_rate_stormControlExtPortmask_get(rtk_portmask_t *pPortmask)`

	Set externsion storm control port mask
	Defined in: storm.h
<b>Parameters</b>	* <i>pPortmask</i> port mask
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_INPUT
	ok failed The module is not initial invalid input parameter

---

### 19.9. rtk\_rate\_stormControlExtEnable\_set

`rtk_api_ret_t rtk_rate_stormControlExtEnable_set(rtk_rate_storm_group_t stormType, rtk_enable_t enable)`

Set externsion storm control state

Defined in: storm.h

<b>Parameters</b>	<i>stormType</i> storm group type <i>enable</i> externsion storm control state
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_NOT_INIT The module is not initial RT_ERR_INPUT invalid input parameter

---

## 19.10. rtk\_rate\_stormControlExtEnable\_get

`rtk_api_ret_t rtk_rate_stormControlExtEnable_get(rtk_rate_storm_group_t  
stormType, rtk_enable_t *pEnable)`

Get externsion storm control state

Defined in: storm.h

<b>Parameters</b>	<i>stormType</i> storm group type <i>*pEnable</i> externsion storm control state
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_NOT_INIT The module is not initial RT_ERR_INPUT invalid input parameter

---

## 19.11. rtk\_rate\_stormControlExtMeterIdx\_set

`rtk_api_ret_t  
rtk_rate_stormControlExtMeterIdx_set(rtk_rate_storm_group_t stormType,  
rtk_uint32 index)`

Set externsion storm control meter index

---

	Defined in: storm.h								
<b>Parameters</b>	<p><i>stormType</i> storm group type</p> <p><i>index</i> externsion storm control state</p>								
<b>Comments</b>									
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_NOT_INIT</td> <td>The module is not initial</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>invalid input parameter</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_NOT_INIT	The module is not initial	RT_ERR_INPUT	invalid input parameter
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_NOT_INIT	The module is not initial								
RT_ERR_INPUT	invalid input parameter								

---

## 19.12. rtk\_rate\_stormControlExtMeterIdx\_get

	<b>rtk_api_ret_t</b>								
	<b>rtk_rate_stormControlExtMeterIdx_get(rtk_rate_storm_group_t <i>stormType</i>, rtk_uint32 *<i>pIndex</i>)</b>								
	Get externsion storm control meter index								
	Defined in: storm.h								
<b>Parameters</b>	<p><i>stormType</i> storm group type</p> <p><i>*pIndex</i> externsion storm control state</p>								
<b>Comments</b>									
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_NOT_INIT</td> <td>The module is not initial</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>invalid input parameter</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_NOT_INIT	The module is not initial	RT_ERR_INPUT	invalid input parameter
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_NOT_INIT	The module is not initial								
RT_ERR_INPUT	invalid input parameter								

---

## 20. Module svlan.h - Unmanaged switch high-level API

Filename: svlan.h

### Description

The file includes SVLAN module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

### List of Symbols

Here is a list of all functions and variables in this module

svlan.h - Unmanaged switch high-level API  
rtk\_svlan\_init  
rtk\_svlan\_servicePort\_add  
rtk\_svlan\_servicePort\_get  
rtk\_svlan\_servicePort\_del  
rtk\_svlan\_tpidEntry\_set  
rtk\_svlan\_tpidEntry\_get  
rtk\_svlan\_priorityRef\_set  
rtk\_svlan\_priorityRef\_get  
rtk\_svlan\_memberPortEntry\_set  
rtk\_svlan\_memberPortEntry\_get  
rtk\_svlan\_memberPortEntry\_adv\_set  
rtk\_svlan\_memberPortEntry\_adv\_get  
rtk\_svlan\_defaultSvlan\_set  
rtk\_svlan\_defaultSvlan\_get  
rtk\_svlan\_c2s\_add  
rtk\_svlan\_c2s\_del  
rtk\_svlan\_c2s\_get  
rtk\_svlan\_untag\_action\_set  
rtk\_svlan\_untag\_action\_get  
rtk\_svlan\_unmatch\_action\_set  
rtk\_svlan\_unmatch\_action\_get  
rtk\_svlan\_dmac\_vidsel\_set  
rtk\_svlan\_dmac\_vidsel\_get  
rtk\_svlan\_ipmc2s\_add  
rtk\_svlan\_ipmc2s\_del  
rtk\_svlan\_ipmc2s\_get  
rtk\_svlan\_l2mc2s\_add  
rtk\_svlan\_l2mc2s\_del  
rtk\_svlan\_l2mc2s\_get  
rtk\_svlan\_sp2c\_add

---

```
rtk_svlan_sp2c_get  
rtk_svlan_sp2c_del  
rtk_svlan_lookupType_set  
rtk_svlan_lookupType_get  
rtk_svlan_trapPri_set  
rtk_svlan_trapPri_get  
rtk_svlan_unassign_action_set  
rtk_svlan_unassign_action_get  
rtk_svlan_checkAndCreateMbr
```

---

## 20.1. rtk\_svlan\_init

**rtk\_api\_ret\_t rtk\_svlan\_init(*void*)**

Initialize SVLAN Configuration

Defined in: svlan.h

**Parameters**

*void*

**Comments**

Ether type of S-tag in 802.1ad is 0x88a8 and there are existed ether type 0x9100 and 0x9200 for Q-in-Q SLAN design. User can set matched ether type as service provider supported protocol.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

## 20.2. rtk\_svlan\_servicePort\_add

**rtk\_api\_ret\_t rtk\_svlan\_servicePort\_add(*rtk\_port\_t port*)**

Add one service port in the specified device

Defined in: svlan.h

**Parameters**

*port*  
Port id.

**Comments**

This API is setting which port is connected to provider switch. All frames receiving from this port must contain accept SVID in S-tag field.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID RT_ERR_INPUT	ok failed SMI access error Invalid port number. Invalid input parameters.
---------------------	--	---

---

### 20.3. rtk\_svlan\_servicePort\_get

`rtk_api_ret_t rtk_svlan_servicePort_get(rtk_portmask_t *pSvlan_portmask)`

Get service ports in the specified device.

Defined in: svlan.h

<b>Parameters</b>	<code>*pSvlan_portmask</code> pointer buffer of svlan ports.
-------------------	---

**Comments**  
This API is setting which port is connected to provider switch. All frames receiving from this port must contain accept SVID in S-tag field.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI	ok failed SMI access error
---------------------	--	----------------------------------

---

### 20.4. rtk\_svlan\_servicePort\_del

`rtk_api_ret_t rtk_svlan_servicePort_del(rtk_port_t port)`

Delete one service port in the specified device

Defined in: svlan.h

<b>Parameters</b>	<code>port</code>
-------------------	-------------------

Port id.

**Comments**  
This API is removing SVLAN service port in the specified device.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_PORT_ID	ok failed SMI access error Invalid port number.
---------------------	--	--

---

## 20.5. rtk\_svlan\_tpidEntry\_set

`rtk_api_ret_t rtk_svlan_tpidEntry_set(rtk_uint32 svlan_tag_id)`

Configure accepted S-VLAN ether type.

Defined in: svlan.h

<b>Parameters</b>	<i>svlan_tag_id</i> Ether type of S
<b>Comments</b>	Ether type of S-tag in 802.1ad is 0x88a8 and there are existed ether type 0x9100 and 0x9200 for Q-in-Q SLAN design. User can set matched ether type as service provider supported protocol.
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.

---

## 20.6. rtk\_svlan\_tpidEntry\_get

`rtk_api_ret_t rtk_svlan_tpidEntry_get(rtk_uint32 *pSvlan_tag_id)`

Get accepted S-VLAN ether type setting.

Defined in: svlan.h

<b>Parameters</b>	<i>*pSvlan_tag_id</i> Ether type of S
<b>Comments</b>	This API is setting which port is connected to provider switch. All frames receiving from this port must contain accept SVID in S-tag field.
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

---

## 20.7.rtk\_svlan\_priorityRef\_set

`rtk_api_ret_t rtk_svlan_priorityRef_set(rtk_svlan_pri_ref_t ref)`

Set S-VLAN upstream priority reference setting.

Defined in: svlan.h

**Parameters**

*ref*  
reference selection parameter.

**Comments**

The API can set the upstream SVLAN tag priority reference source. The related priority sources are as following:

- REF\_INTERNAL\_PRI,
- REF\_CTAG\_PRI,
- REF\_SVLAN\_PRI.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_INPUT</code>	Invalid input parameter.

---

## 20.8.rtk\_svlan\_priorityRef\_get

`rtk_api_ret_t rtk_svlan_priorityRef_get(rtk_svlan_pri_ref_t *pRef)`

Get S-VLAN upstream priority reference setting.

Defined in: svlan.h

**Parameters**

*\*pRef*  
reference selection parameter.

**Comments**

The API can get the upstream SVLAN tag priority reference source. The related priority sources are as following:

- REF\_INTERNAL\_PRI,
- REF\_CTAG\_PRI,
- REF\_SVLAN\_PRI.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

---

## 20.9. rtk\_svlan\_memberPortEntry\_set

```
rtk_api_ret_t rtk_svlan_memberPortEntry_set(rtk_uint32 svid_idx,  
    rtk_svlan_memberCfg_t *psvlan_cfg)
```

Configure system SVLAN member content

Defined in: svlan.h

**Parameters**

<i>svid_idx</i>	SVID id
<i>*psvlan_cfg</i>	SVLAN member configuration

**Comments**

The API can set system 64 accepted s-tag frame format. Only 64 SVID S-tag frame will be accepted to receiving from uplink ports. Other SVID S-tag frame or S-untagged frame will be droped by default setup.

- *rtk\_svlan\_memberCfg\_t->svid* is SVID of SVLAN member configuration.
- *rtk\_svlan\_memberCfg\_t->memberport* is member port mask of SVLAN member configuration.
- *rtk\_svlan\_memberCfg\_t->fid* is filtering database of SVLAN member configuration.
- *rtk\_svlan\_memberCfg\_t->priority* is priority of SVLAN member configuration.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
RT_ERR_PORT_MASK	Invalid portmask.
RT_ERR_SVLAN_TABLE_FULL	SVLAN configuration is full.

---

## 20.10. rtk\_svlan\_memberPortEntry\_get

```
rtk_api_ret_t rtk_svlan_memberPortEntry_get(rtk_uint32 svid_idx,  
    rtk_svlan_memberCfg_t *pSvlan_cfg)
```

Get SVLAN member Configure.

Defined in: svlan.h

**Parameters**

	<i>svid_idx</i>	SVLAN id
	<i>*pSvlan_cfg</i>	SVLAN member configuration
<b>Comments</b>	The API can get system 64 accepted s-tag frame format. Only 64 SVID S-tag frame will be accepted to receiving from uplink ports. Other SVID S-tag frame or S-untagged frame will be dropped.	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_SVLAN_ENTRY_NOT_FOU ND RT_ERR_INPUT	
		ok failed SMI access error specified svlan entry not found. ND Invalid input parameters.

---

## 20.11. rtk\_svlan\_memberPortEntry\_adv\_set

**rtk\_api\_ret\_t rtk\_svlan\_memberPortEntry\_adv\_set(rtk\_uint32 *idx*,  
rtk\_svlan\_memberCfg\_t \**pSvlan\_cfg*)**

Configure system SVLAN member by index

Defined in: svlan.h

*idx*  
Index (0 ~ 63)

*\*pSvlan\_cfg*  
SVLAN member configuration

### Comments

The API can set system 64 accepted s-tag frame format by index.

- rtk\_svlan\_memberCfg\_t->svid is SVID of SVLAN member configuration.
- rtk\_svlan\_memberCfg\_t->memberport is member port mask of SVLAN member configuration.
- rtk\_svlan\_memberCfg\_t->fid is filtering database of SVLAN member configuration.
- rtk\_svlan\_memberCfg\_t->priority is priority of SVLAN member configuration.

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.

---

RT_ERR_PORT_MASK	Invalid portmask.
RT_ERR_SVLAN_TABLE_FULL	SVLAN configuration is full.

---

## 20.12. rtk\_svlan\_memberPortEntry\_adv\_get

`rtk_api_ret_t rtk_svlan_memberPortEntry_adv_get(rtk_uint32 idx,  
rtk_svlan_memberCfg_t *pSvlan_cfg)`

Get SVLAN member Configure by index.

Defined in: svlan.h

### Parameters

<i>idx</i>	
	Index (0 ~ 63)
<i>*pSvlan_cfg</i>	SVLAN member configuration

### Comments

The API can get system 64 accepted s-tag frame format. Only 64 SVID S-tag frame will be accepted to receiving from uplink ports. Other SVID S-tag frame or S-untagged frame will be droped.

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_SVLAN_ENTRY_NOT_FOU ND	specified svlan entry not found.
RT_ERR_INPUT	Invalid input parameters.

## 20.13. rtk\_svlan\_defaultSvlan\_set

`rtk_api_ret_t rtk_svlan_defaultSvlan_set(rtk_port_t port, rtk_vlan_t svid)`

Configure default egress SVLAN.

Defined in: svlan.h

### Parameters

<i>port</i>	
	Source port
<i>svid</i>	SVLAN id

<b>Comments</b>	The API can set port n S-tag format index while receiving frame from port n is transmit through uplink port with s-tag field												
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameter.</td> </tr> <tr> <td>RT_ERR_SVLAN_VID</td> <td>Invalid SVLAN VID parameter.</td> </tr> <tr> <td>RT_ERR_SVLAN_ENTRY_NOT_FOU ND</td> <td>specified svlan entry not found.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameter.	RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.	RT_ERR_SVLAN_ENTRY_NOT_FOU ND	specified svlan entry not found.
RT_ERR_OK	ok												
RT_ERR_FAILED	failed												
RT_ERR_SMI	SMI access error												
RT_ERR_INPUT	Invalid input parameter.												
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.												
RT_ERR_SVLAN_ENTRY_NOT_FOU ND	specified svlan entry not found.												

---

## 20.14. rtk\_svlan\_defaultSvlan\_get

`rtk_api_ret_t rtk_svlan_defaultSvlan_get(rtk_port_t port, rtk_vlan_t *pSvid)`

Get the configure default egress SVLAN.

Defined in: svlan.h

<b>Parameters</b>	<code>port</code> Source port <code>*pSvid</code> SVLAN VID								
<b>Comments</b>	The API can get port n S-tag format index while receiving frame from port n is transmit through uplink port with s-tag field								
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_INPUT</td> <td>Invalid input parameters.</td> </tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Invalid input parameters.								

---

## 20.15. rtk\_svlan\_c2s\_add

`rtk_api_ret_t rtk_svlan_c2s_add(rtk_vlan_t vid, rtk_port_t src_port,  
rtk_vlan_t svvid)`

Configure SVLAN C2S table

Defined in: svlan.h

---

<b>Parameters</b>	<i>vid</i> VLAN ID <i>src_port</i> Ingress Port <i>svid</i> SVLAN VID
<b>Comments</b>	The API can set system C2S configuration. ASIC will check upstream's VID and assign related SVID to matched packet. There are 128 SVLAN C2S configurations.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port ID. RT_ERR_SVLAN_VID Invalid SVLAN VID parameter. RT_ERR_VLAN_VID Invalid VID parameter. RT_ERR_OUT_OF_RANGE input out of range. RT_ERR_INPUT Invalid input parameters.

---

## 20.16. rtk\_svlan\_c2s\_del

`rtk_api_ret_t rtk_svlan_c2s_del(rtk_vlan_t vid, rtk_port_t src_port)`

Delete one C2S entry

Defined in: svlan.h

<b>Parameters</b>	<i>vid</i> VLAN ID <i>src_port</i> Ingress Port
<b>Comments</b>	The API can delete system C2S configuration. There are 128 SVLAN C2S configurations.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_VLAN_VID Invalid VID parameter. RT_ERR_PORT_ID Invalid port ID. RT_ERR_OUT_OF_RANGE input out of range.

---

## 20.17. rtk\_svlan\_c2s\_get

```
rtk_api_ret_t rtk_svlan_c2s_get(rtk_vlan_t vid, rtk_port_t src_port,  
                                rtk_vlan_t *pSvid)
```

Get configure SVLAN C2S table

Defined in: svlan.h

**Parameters**

*vid*  
VLAN ID

*src\_port*  
Ingress Port

*\*pSvid*  
SVLAN ID

**Comments**

The API can get system C2S configuration. There are 128 SVLAN C2S configurations.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port ID.
RT_ERR_OUT_OF_RANGE	input out of range.

---

## 20.18. rtk\_svlan\_untag\_action\_set

```
rtk_api_ret_t rtk_svlan_untag_action_set(rtk_svlan_untag_action_t action,  
                                         rtk_vlan_t svid)
```

Configure Action of downstream Un-Stag packet

Defined in: svlan.h

**Parameters**

*action*  
Action for UnStag

*svid*  
The SVID assigned to UnStag packet

**Comments**

---

The API can configure action of downstream Un-Stag packet. A SVID assigned to the un-stag is also supported by this API. The parameter of svid is only referenced when the action is set to UNTAG\_ASSIGN

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
	RT_ERR_SVLAN_ENTRY_NOT_FOUNDED	specified svlan entry not found.
	RT_ERR_OUT_OF_RANGE	input out of range.
	RT_ERR_INPUT	Invalid input parameters.

---

## 20.19. rtk\_svlan\_untag\_action\_get

`rtk_api_ret_t rtk_svlan_untag_action_get(rtk_svlan_untag_action_t *pAction,  
rtk_vlan_t *pSvid)`

Get Action of downstream Un-Stag packet

Defined in: svlan.h

<b>Parameters</b>	<i>*pAction</i>	Action for UnStag
	<i>*pSvid</i>	The SVID assigned to UnStag packet

**Comments**  
The API can Get action of downstream Un-Stag packet. A SVID assigned to the un-stag is also retrieved by this API. The parameter pSvid is only referenced when the action is UNTAG\_ASSIGN

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
	RT_ERR_SVLAN_ENTRY_NOT_FOUNDED	specified svlan entry not found.
	RT_ERR_OUT_OF_RANGE	input out of range.
	RT_ERR_INPUT	Invalid input parameters.

---

## 20.20. rtk\_svlan\_unmatch\_action\_set

**rtk\_api\_ret\_t rtk\_svlan\_unmatch\_action\_set(rtk\_svlan\_unmatch\_action\_t  
action, rtk\_vlan\_t svid)**

Configure Action of downstream Unmatch packet

Defined in: svlan.h

**Parameters**

*action*  
Action for Unmatch

*svid*  
The SVID assigned to Unmatch packet

**Comments**

The API can configure action of downstream Un-match packet. A SVID assigned to the un-match is also supported by this API. The parameter od svid is only referenced when the action is set to UNMATCH\_ASSIGN

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
RT_ERR_SVLAN_ENTRY_NOT_FOU ND	specified svlan entry not found.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_INPUT	Invalid input parameters.

---

## 20.21. rtk\_svlan\_unmatch\_action\_get

**rtk\_api\_ret\_t rtk\_svlan\_unmatch\_action\_get(rtk\_svlan\_unmatch\_action\_t  
\*pAction, rtk\_vlan\_t \*pSvid)**

Get Action of downstream Unmatch packet

Defined in: svlan.h

**Parameters**

*\*pAction*  
Action for Unmatch

*\*pSvid*  
The SVID assigned to Unmatch packet

**Comments**

---

The API can Get action of downstream Un-match packet. A SVID assigned to the un-match is also retrieved by this API. The parameter pSvid is only referenced when the action is UNMATCH\_ASSIGN

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
	RT_ERR_SVLAN_ENTRY_NOT_FOUNDED	specified svlan entry not found.
	RT_ERR_OUT_OF_RANGE	input out of range.
	RT_ERR_INPUT	Invalid input parameters.

---

## 20.22. rtk\_svlan\_dmac\_vidsel\_set

`rtk_api_ret_t rtk_svlan_dmac_vidsel_set(rtk_port_t port, rtk_enable_t enable)`

Set DMAC CVID selection

Defined in: svlan.h

### Parameters

*port*  
Port  
*enable*  
state of DMAC CVID Selection

### Comments

This API can set DMAC CVID Selection state

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
RT_ERR_SVLAN_ENTRY_NOT_FOUNDED	specified svlan entry not found.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_INPUT	Invalid input parameters.

---

## 20.23. rtk\_svlan\_dmac\_vidsel\_get

**rtk\_api\_ret\_t rtk\_svlan\_dmac\_vidsel\_get(rtk\_port\_t port, rtk\_enable\_t \*pEnable)**

Get DMAC CVID selection

Defined in: svlan.h

**Parameters**

*port*  
Port  
*\*pEnable*  
state of DMAC CVID Selection

**Comments**

This API can get DMAC CVID Selection state

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.
RT_ERR_SVLAN_ENTRY_NOT_FOU ND	specified svlan entry not found.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_INPUT	Invalid input parameters.

---

## 20.24. rtk\_svlan\_ipmc2s\_add

**rtk\_api\_ret\_t rtk\_svlan\_ipmc2s\_add(ipaddr\_t ipmc, ipaddr\_t ipmcMsk,  
rtk\_vlan\_t svid)**

add ip multicast address to SVLAN

Defined in: svlan.h

**Parameters**

*ipmc*  
SVLAN VID  
*ipmcMsk*  
ip multicast address  
*svid*  
ip multicast mask

**Comments**

---

The API can set IP multicast to SVID configuration. If upstream packet is IPv4 multicast packet and DIP is matched MC2S configuration, ASIC will assign egress SVID to the packet. There are 32 SVLAN multicast configurations for IP and L2 multicast.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_SVLAN_VID RT_ERR_SVLAN_ENTRY_NOT_FOU ND RT_ERR_OUT_OF_RANGE RT_ERR_INPUT	ok failed SMI access error Invalid SVLAN VID parameter. specified svlan entry not found. input out of range. Invalid input parameters.
---------------------	---	--

---

## 20.25. rtk\_svlan\_ipmc2s\_del

`rtk_api_ret_t rtk_svlan_ipmc2s_del(ipaddr_t ipmc, ipaddr_t ipmcMsk)`

delete ip multicast address to SVLAN

Defined in: svlan.h

**Parameters**

*ipmc*  
ip multicast address  
*ipmcMsk*  
ip multicast mask

**Comments**

The API can delete IP multicast to SVID configuration. There are 32 SVLAN multicast configurations for IP and L2 multicast.

**Return Codes**

RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_SVLAN_VID RT_ERR_OUT_OF_RANGE	ok failed SMI access error Invalid SVLAN VID parameter. input out of range.
---	---

---

## 20.26. rtk\_svlan\_ipmc2s\_get

```
rtk_api_ret_t rtk_svlan_ipmc2s_get(ipaddr_t ipmc, ipaddr_t ipmcMsk,  
rtk_vlan_t *pSvid)
```

Get ip multicast address to SVLAN

Defined in: svlan.h

**Parameters**

<i>ipmc</i>	ip multicast address
<i>ipmcMsk</i>	ip multicast mask
<i>*pSvid</i>	SVLAN VID

**Comments**

The API can get IP multicast to SVID configuration. There are 32 SVLAN multicast configurations for IP and L2 multicast.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OUT_OF_RANGE	input out of range.

---

## 20.27. rtk\_svlan\_l2mc2s\_add

```
rtk_api_ret_t rtk_svlan_l2mc2s_add(rtk_mac_t mac, rtk_mac_t macMsk,  
rtk_vlan_t svid)
```

Add L2 multicast address to SVLAN

Defined in: svlan.h

**Parameters**

<i>mac</i>	L2 multicast address
<i>macMsk</i>	L2 multicast address mask
<i>svid</i>	SVLAN VID

**Comments**

---

The API can set L2 Multicast to SVID configuration. If upstream packet is L2 multicast packet and DMAC is matched, ASIC will assign egress SVID to the packet. There are 32 SVLAN multicast configurations for IP and L2 multicast.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_SVLAN_VID RT_ERR_SVLAN_ENTRY_NOT_FOUND RT_ERR_OUT_OF_RANGE RT_ERR_INPUT	ok failed SMI access error Invalid SVLAN VID parameter. specified svlan entry not found. input out of range. Invalid input parameters.
---------------------	---	--

---

## 20.28. rtk\_svlan\_l2mc2s\_del

`rtk_api_ret_t rtk_svlan_l2mc2s_del(rtk_mac_t mac, rtk_mac_t macMsk)`

delete L2 multicast address to SVLAN

Defined in: svlan.h

### Parameters

*mac*  
L2 multicast address

*macMsk*  
L2 multicast address mask

### Comments

The API can delete Multicast to SVID configuration. There are 32 SVLAN multicast configurations for IP and L2 multicast.

### Return Codes

RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_SVLAN_VID RT_ERR_OUT_OF_RANGE	ok failed SMI access error Invalid SVLAN VID parameter. input out of range.
---	---

---

## 20.29. rtk\_svlan\_l2mc2s\_get

`rtk_api_ret_t rtk_svlan_l2mc2s_get(rtk_mac_t mac, rtk_mac_t macMsk, rtk_vlan_t *pSvid)`



---

RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_SVLAN_VID	invalid SVLAN VID parameter.
RT_ERR_VLAN_VID	Invalid VID parameter.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_INPUT	Invalid input parameters.

---

### 20.31. rtk\_svlan\_sp2c\_get

`rtk_api_ret_t rtk_svlan_sp2c_get(rtk_vlan_t svid, rtk_port_t dst_port,  
rtk_vlan_t *pCvid)`

Get configure system SP2C content

Defined in: svlan.h

#### Parameters

<i>svid</i>	SVLAN VID
<i>dst_port</i>	Destination port of SVLAN to CVLAN configuration
<i>*pCvid</i>	VLAN ID

#### Comments

The API can get SVID & Destination Port to CVLAN configuration. There are 128 SP2C configurations.

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.

---

### 20.32. rtk\_svlan\_sp2c\_del

`rtk_api_ret_t rtk_svlan_sp2c_del(rtk_vlan_t svid, rtk_port_t dst_port)`

	Delete system SP2C configuration												
	Defined in: svlan.h												
<b>Parameters</b>	<p><i>svid</i> SVLAN VID</p> <p><i>dst_port</i> Destination port of SVLAN to CVLAN configuration</p>												
<b>Comments</b>	The API can delete SVID & Destination Port to CVLAN configuration. There are 128 SP2C configurations.												
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_PORT_ID</td><td>Invalid port number.</td></tr> <tr> <td>RT_ERR_SVLAN_VID</td><td>Invalid SVLAN VID parameter.</td></tr> <tr> <td>RT_ERR_OUT_OF_RANGE</td><td>input out of range.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.	RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_OK	ok												
RT_ERR_FAILED	failed												
RT_ERR_SMI	SMI access error												
RT_ERR_PORT_ID	Invalid port number.												
RT_ERR_SVLAN_VID	Invalid SVLAN VID parameter.												
RT_ERR_OUT_OF_RANGE	input out of range.												

---

### 20.33. rtk\_svlan\_lookupType\_set

`rtk_api_ret_t rtk_svlan_lookupType_set(rtk_svlan_lookupType_t type)`

Set lookup type of SVLAN

Defined in: svlan.h

<b>Parameters</b>	<i>type</i> lookup type				
<b>Comments</b>	none				
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed
RT_ERR_OK	ok				
RT_ERR_FAILED	failed				

---

### 20.34. rtk\_svlan\_lookupType\_get

`rtk_api_ret_t rtk_svlan_lookupType_get(rtk_svlan_lookupType_t *pType)`

Get lookup type of SVLAN

Defined in: svlan.h

---

<b>Parameters</b>	<i>*pType</i> lookup type
<b>Comments</b>	none
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED

---

### 20.35. rtk\_svlan\_trapPri\_set

`rtk_api_ret_t rtk_svlan_trapPri_set(rtk_pri_t priority)`

Set svlan trap priority

Defined in: svlan.h

<b>Parameters</b>	<i>priority</i> priority for trap packets
<b>Comments</b>	None
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_QOS_INT_PRIORITY

---

### 20.36. rtk\_svlan\_trapPri\_get

`rtk_api_ret_t rtk_svlan_trapPri_get(rtk_pri_t *pPriority)`

Get svlan trap priority

Defined in: svlan.h

<b>Parameters</b>	<i>*pPriority</i> priority for trap packets
<b>Comments</b>	None
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NULL_POINTER

---

## 20.37. rtk\_svlan\_unassign\_action\_set

`rtk_api_ret_t rtk_svlan_unassign_action_set(rtk_svlan_unassign_action_t action)`

Configure Action of upstream without svid assign action

Defined in: svlan.h

**Parameters**

*action*

Action for Un

**Comments**

The API can configure action of upstream Un-assign svid packet. If action is not trap to CPU, the port-based SVID sure be assign as system need

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_OUT\_OF\_RANGE

input out of range.

RT\_ERR\_INPUT

Invalid input parameters.

---

## 20.38. rtk\_svlan\_unassign\_action\_get

`rtk_api_ret_t rtk_svlan_unassign_action_get(rtk_svlan_unassign_action_t *pAction)`

Get action of upstream without svid assignment

Defined in: svlan.h

**Parameters**

*\*pAction*

Action for Un

**Comments**

None

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

---

---

## 20.39. rtk\_svlan\_checkAndCreateMbr

`rtk_api_ret_t rtk_svlan_checkAndCreateMbr(rtk_vlan_t vid, rtk_uint32 *pIndex)`

Check and create Member configuration and return index

Defined in: svlan.h

### Parameters

*vid*  
VLAN id.  
*\*pIndex*  
Member configuration index

### Comments

### Return Codes

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_VLAN_VID</code>	Invalid VLAN ID.
<code>RT_ERR_TBL_FULL</code>	Member Configuration table full

---

## 21. Module trap.h - Unmanaged switch high-level API

Filename: trap.h

### Description

The file includes Trap module high-layer API definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

### List of Symbols

Here is a list of all functions and variables in this module

trap.h - Unmanaged switch high-level API  
`rtk_trap_unknownUnicastPktAction_set`  
`rtk_trap_unknownUnicastPktAction_get`  
`rtk_trap_unknownMacPktAction_set`  
`rtk_trap_unknownMacPktAction_get`  
`rtk_trap_unmatchMacPktAction_set`

```
rtk_trap_unmatchMacPktAction_get  
rtk_trap_unmatchMacMoving_set  
rtk_trap_unmatchMacMoving_get  
rtk_trap_unknownMcastPktAction_set  
rtk_trap_unknownMcastPktAction_get  
rtk_trap_lldpEnable_set  
rtk_trap_lldpEnable_get  
rtk_trap_reasonTrapToCpuPriority_set  
rtk_trap_reasonTrapToCpuPriority_get  
rtk_trap_rmaAction_set  
rtk_trap_rmaAction_get  
rtk_trap_rmaKeepFormat_set  
rtk_trap_rmaKeepFormat_get  
rtk_trap_portUnknownMacPktAction_set  
rtk_trap_portUnknownMacPktAction_get  
rtk_trap_portUnmatchMacPktAction_set  
rtk_trap_portUnmatchMacPktAction_get
```

---

## 21.1. rtk\_trap\_unknownUnicastPktAction\_set

`rtk_api_ret_t rtk_trap_unknownUnicastPktAction_set(rtк_port_t port,  
rtk_trap_unicast_action_t ucast_action)`

Set unknown unicast packet action configuration.

Defined in: trap.h

### Parameters

*port*  
ingress port ID for unknown unicast packet

*ucast\_action*  
Unknown unicast action.

### Comments

This API can set unknown unicast packet action configuration. The unknown unicast action is as following.

- UCST\_ACTION\_FORWARD\_PMASK
- UCST\_ACTION\_DROP
- UCST\_ACTION\_TRAP2CPU
- UCST\_ACTION\_FLOODING

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NOT_ALLOWED	Invalid action.

---

RT_ERR_INPUT	Invalid input parameters.
--------------	---------------------------

---

## 21.2. rtk\_trap\_unknownUnicastPktAction\_get

`rtk_api_ret_t rtk_trap_unknownUnicastPktAction_get(rtк_port_t port,  
rtк_trap_icast_action_t *pUcast_action)`

Get unknown unicast packet action configuration.

Defined in: trap.h

### Parameters

*port*  
ingress port ID for unknown unicast packet  
*\*pUcast\_action*  
Unknown unicast action.

### Comments

This API can get unknown unicast packet action configuration. The unknown unicast action is as following:

- UCASACTION\_FORWARD\_PMASK
- UCASACTION\_DROP
- UCASACTION\_TRAP2CPU
- UCASACTION\_FLOODING

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NOT_ALLOWED	Invalid action.
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_NULL_POINTER	Null pointer

---

## 21.3. rtk\_trap\_unknownMacPktAction\_set

`rtk_api_ret_t rtk_trap_unknownMacPktAction_set(rtк_trap_icast_action_t  
uCast_action)`

Set unknown source MAC packet action configuration.

Defined in: trap.h

### Parameters

*uCast\_action*

Unknown source MAC action.

**Comments**

This API can set unknown unicast packet action configuration. The unknown unicast action is as following:

- UCST\_ACTION\_FORWARD\_PMASK
- UCST\_ACTION\_DROP
- UCST\_ACTION\_TRAP2CPU

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NOT_ALLOWED	Invalid action.
RT_ERR_INPUT	Invalid input parameters.

---

## 21.4. rtk\_trap\_unknownMacPktAction\_get

`rtk_api_ret_t rtk_trap_unknownMacPktAction_get(rtk_trap_uCast_action_t *pUcast_action)`

Get unknown source MAC packet action configuration.

Defined in: trap.h

*\*pUcast\_action*

Unknown source MAC action.

**Parameters**

**Comments**

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Null Pointer.
RT_ERR_INPUT	Invalid input parameters.

---

## 21.5. rtk\_trap\_unmatchMacPktAction\_set

`rtk_api_ret_t rtk_trap_unmatchMacPktAction_set(rtk_trap_uCast_action_t uCast_action)`

---

Set unmatch source MAC packet action configuration.

Defined in: trap.h

**Parameters**

*ucast\_action*

Unknown source MAC action.

**Comments**

This API can set unknown unicast packet action configuration. The unknown unicast action is as following:

- UCST\_ACTION\_FORWARD\_PMASK
- UCST\_ACTION\_DROP
- UCST\_ACTION\_TRAP2CPU

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_NOT\_ALLOWED

Invalid action.

RT\_ERR\_INPUT

Invalid input parameters.

---

## 21.6. rtk\_trap\_unmatchMacPktAction\_get

`rtk_api_ret_t rtk_trap_unmatchMacPktAction_get(rtk_trap_uctact_action_t *pUcast_action)`

Get unmatch source MAC packet action configuration.

Defined in: trap.h

**Parameters**

*\*pUcast\_action*

Unknown source MAC action.

**Comments**

This API can set unknown unicast packet action configuration. The unknown unicast action is as following:

- UCST\_ACTION\_FORWARD\_PMASK
- UCST\_ACTION\_DROP
- UCST\_ACTION\_TRAP2CPU

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_NOT\_ALLOWED

Invalid action.

RT\_ERR\_INPUT

Invalid input parameters.

## 21.7.rtk\_trap\_unmatchMacMoving\_set

`rtk_api_ret_t rtk_trap_unmatchMacMoving_set(rtk_port_t port,  
rtk_enable_t enable)`

Set unmatch source MAC packet moving state.

Defined in: trap.h

### Parameters

*port*  
Port ID.

*enable*  
ENABLED: allow SA moving, DISABLE: don't allow SA moving.

### Comments

This API can set unknown unicast packet action configuration. The unknown unicast action is as following:

- UCST\_ACTION\_FORWARD\_PMASK
- UCST\_ACTION\_DROP
- UCST\_ACTION\_TRAP2CPU

### Return Codes

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error
<code>RT_ERR_NOT_ALLOWED</code>	Invalid action.
<code>RT_ERR_INPUT</code>	Invalid input parameters.

## 21.8.rtk\_trap\_unmatchMacMoving\_get

`rtk_api_ret_t rtk_trap_unmatchMacMoving_get(rtk_port_t port,  
rtk_enable_t *pEnable)`

Set unmatch source MAC packet moving state.

Defined in: trap.h

### Parameters

*port*  
Port ID.

*\*pEnable*  
ENABLED: allow SA moving, DISABLE: don't allow SA moving.

### Comments

This API can set unknown unicast packet action configuration. The unknown unicast action is as following:

- 
- UCAST\_ACTION\_FORWARD\_PMASK
  - UCAST\_ACTION\_DROP
  - UCAST\_ACTION\_TRAP2CPU

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_NOT_ALLOWED	Invalid action.
	RT_ERR_INPUT	Invalid input parameters.

---

## 21.9. rtk\_trap\_unknownMcastPktAction\_set

`rtk_api_ret_t rtk_trap_unknownMcastPktAction_set(rtk_port_t port,  
rtk_mcast_type_t type, rtk_trap_mcast_action_t mcast_action)`

Set behavior of unknown multicast

Defined in: trap.h

<b>Parameters</b>	<i>port</i> Port id.
	<i>type</i> unknown multicast packet type.
	<i>mcast_action</i> unknown multicast action.

**Comments**  
When receives an unknown multicast packet, switch may trap, drop or flood this packet

(1) The unknown multicast packet type is as following:

- MCAST\_L2
- MCAST\_IPV4
- MCAST\_IPV6

(2) The unknown multicast action is as following:

- MCAST\_ACTION\_FORWARD
- MCAST\_ACTION\_DROP
- MCAST\_ACTION\_TRAP2CPU

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.
	RT_ERR_NOT_ALLOWED	Invalid action.

RT\_ERR\_INPUT

Invalid input parameters.

## 21.10. rtk\_trap\_unknownMcastPktAction\_get

`rtk_api_ret_t rtk_trap_unknownMcastPktAction_get(rtк_port_t port,  
rtк_mcast_type_t type, rtк_trap_mcast_action_t *pMcast_action)`

Get behavior of unknown multicast

Defined in: trap.h

### Parameters

*port*  
unknown multicast packet type.  
*type*  
unknown multicast action.  
*\*pMcast\_action*  
unknown multicast action.

### Comments

When receives an unknown multicast packet, switch may trap, drop or flood this packet

(1) The unknown multicast packet type is as following:

- MCAST\_L2
- MCAST\_IPV4
- MCAST\_IPV6

(2) The unknown multicast action is as following:

- MCAST\_ACTION\_FORWARD
- MCAST\_ACTION\_DROP
- MCAST\_ACTION\_TRAP2CPU

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_NOT_ALLOWED	Invalid operation.
RT_ERR_INPUT	Invalid input parameters.

## 21.11. rtk\_trap\_lldpEnable\_set

`rtk_api_ret_t rtk_trap_lldpEnable_set(rtк_enable_t enabled)`

---

	Set LLDP enable.	
	Defined in: trap.h	
<b>Parameters</b>	<i>enabled</i>	
	LLDP enable, 0: follow RMA, 1: use LLDP action.	
<b>Comments</b>	DMAC	Assignment
	- 01:80:c2:00:00:0e ethertype = 0x88CC	LLDP
	- 01:80:c2:00:00:03 ethertype = 0x88CC	
	- 01:80:c2:00:00:00 ethertype = 0x88CC	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_NOT_ALLOWED	Invalid action.
	RT_ERR_INPUT	Invalid input parameters.

---

## 21.12. rtk\_trap\_lldpEnable\_get

`rtk_api_ret_t rtk_trap_lldpEnable_get(rtk_enable_t *pEnabled)`

	Get LLDP status.	
	Defined in: trap.h	
<b>Parameters</b>	<i>*pEnabled</i>	
	LLDP enable, 0: follow RMA, 1: use LLDP action.	
<b>Comments</b>	LLDP is as following definition.	Assignment
	- DMAC	
	- 01:80:c2:00:00:0e ethertype = 0x88CC	LLDP
	- 01:80:c2:00:00:03 ethertype = 0x88CC	
	- 01:80:c2:00:00:00 ethertype = 0x88CC	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.

---

## 21.13. rtk\_trap\_reasonTrapToCpuPriority\_set

`rtk_api_ret_t`

`rtk_trap_reasonTrapToCpuPriority_set(rtk_trap_reason_type_t type,  
rtk_pri_t priority)`

Set priority value of a packet that trapped to CPU port according to specific reason.

Defined in: trap.h

**Parameters**

*type*

reason that trap to CPU port.

*priority*

internal priority that is going to be set for specific trap reason.

**Comments**

Currently the trap reason that supported are listed as follows:

- TRAP\_REASON\_RMA
- TRAP\_REASON\_OAM
- TRAP\_REASON\_1XUNAUTH
- TRAP\_REASON\_VLANSTACK
- TRAP\_REASON\_UNKNOWNMC

**Return Codes**

`RT_ERR_OK`

ok

`RT_ERR_FAILED`

failed

`RT_ERR_NOT_INIT`

The module is not initial

`RT_ERR_INPUT`

Invalid input parameter

---

## 21.14. rtk\_trap\_reasonTrapToCpuPriority\_get

`rtk_api_ret_t`

`rtk_trap_reasonTrapToCpuPriority_get(rtk_trap_reason_type_t type,  
rtk_pri_t *pPriority)`

Get priority value of a packet that trapped to CPU port according to specific reason.

Defined in: trap.h

**Parameters**

*type*

reason that trap to CPU port.

---

	<i>*pPriority</i>	configured internal priority for such reason.
<b>Comments</b>	Currently the trap reason that supported are listed as follows:	
	- TRAP_REASON_RMA	
	- TRAP_REASON_OAM	
	- TRAP_REASON_1XUNAUTH	
	- TRAP_REASON_VLANSTACK	
	- TRAP_REASON_UNKNOWNMNC	
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_NOT_INIT RT_ERR_INPUT RT_ERR_NULL_POINTER	ok failed The module is not initial Invalid input parameter NULL pointer

---

## 21.15. rtk\_trap\_rmaAction\_set

`rtk_api_ret_t rtk_trap_rmaAction_set(rtk_trap_type_t type,  
rtk_trap_rma_action_t rma_action)`

Set Reserved multicast address action configuration.

Defined in: trap.h

<b>Parameters</b>	<i>type</i> rma type. <i>rma_action</i> RMA action.
-------------------	--

**Comments** There are 48 types of Reserved Multicast Address frame for application usage.

(1)They are as following definition.

- TRAP\_BRG\_GROUP,
- TRAP\_FD\_PAUSE,
- TRAP\_SP\_MCAST,
- TRAP\_1X\_PAE,
- TRAP\_UNDEF\_BRG\_04,
- TRAP\_UNDEF\_BRG\_05,
- TRAP\_UNDEF\_BRG\_06,
- TRAP\_UNDEF\_BRG\_07,
- TRAP\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- TRAP\_UNDEF\_BRG\_09,
- TRAP\_UNDEF\_BRG\_0A,
- TRAP\_UNDEF\_BRG\_0B,

- TRAP\_UNDEF\_BRG\_0C,
  - TRAP\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
  - TRAP\_8021AB,
  - TRAP\_UNDEF\_BRG\_0F,
  - TRAP\_BRG\_MNGEMENT,
  - TRAP\_UNDEFINED\_11,
  - TRAP\_UNDEFINED\_12,
  - TRAP\_UNDEFINED\_13,
  - TRAP\_UNDEFINED\_14,
  - TRAP\_UNDEFINED\_15,
  - TRAP\_UNDEFINED\_16,
  - TRAP\_UNDEFINED\_17,
  - TRAP\_UNDEFINED\_18,
  - TRAP\_UNDEFINED\_19,
  - TRAP\_UNDEFINED\_1A,
  - TRAP\_UNDEFINED\_1B,
  - TRAP\_UNDEFINED\_1C,
  - TRAP\_UNDEFINED\_1D,
  - TRAP\_UNDEFINED\_1E,
  - TRAP\_UNDEFINED\_1F,
  - TRAP\_GMRP,
  - TRAP\_GVRP,
  - TRAP\_UNDEF\_GARP\_22,
  - TRAP\_UNDEF\_GARP\_23,
  - TRAP\_UNDEF\_GARP\_24,
  - TRAP\_UNDEF\_GARP\_25,
  - TRAP\_UNDEF\_GARP\_26,
  - TRAP\_UNDEF\_GARP\_27,
  - TRAP\_UNDEF\_GARP\_28,
  - TRAP\_UNDEF\_GARP\_29,
  - TRAP\_UNDEF\_GARP\_2A,
  - TRAP\_UNDEF\_GARP\_2B,
  - TRAP\_UNDEF\_GARP\_2C,
  - TRAP\_UNDEF\_GARP\_2D,
  - TRAP\_UNDEF\_GARP\_2E,
  - TRAP\_UNDEF\_GARP\_2F,
  - TRAP\_CDP.
  - TRAP\_CSSTP.
  - TRAP\_LLDP.
- (2) The RMA action is as following:
- RMA\_ACTION\_FORWARD
  - RMA\_ACTION\_TRAP2CPU
  - RMA\_ACTION\_DROP
  - RMA\_ACTION\_FORWARD\_EXCLUDE\_CPU

<b>Return Codes</b>	RT_ERR_OK	ok
---------------------	-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	invalid input parameters.
RT_ERR_ENABLE	Invalid IFG parameter

---

## 21.16. rtk\_trap\_rmaAction\_get

`rtk_api_ret_t rtk_trap_rmaAction_get(rtk_trap_type_t type,  
rtk_trap_rma_action_t *pRma_action)`

Get Reserved multicast address action configuration.

Defined in: trap.h

### Parameters

*type*  
rma type.  
*\*pRma\_action*  
RMA action.

### Comments

There are 48 types of Reserved Multicast Address frame for application usage.

(1)They are as following definition,

- TRAP\_BRG\_GROUP,
- TRAP\_FD\_PAUSE,
- TRAP\_SP\_MCAST,
- TRAP\_1X\_PAE,
- TRAP\_UNDEF\_BRG\_04,
- TRAP\_UNDEF\_BRG\_05,
- TRAP\_UNDEF\_BRG\_06,
- TRAP\_UNDEF\_BRG\_07,
- TRAP\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- TRAP\_UNDEF\_BRG\_09,
- TRAP\_UNDEF\_BRG\_0A,
- TRAP\_UNDEF\_BRG\_0B,
- TRAP\_UNDEF\_BRG\_0C,
- TRAP\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- TRAP\_8021AB,
- TRAP\_UNDEF\_BRG\_0F,
- TRAP\_BRG\_MNGEMENT,
- TRAP\_UNDEFINED\_11,
- TRAP\_UNDEFINED\_12,
- TRAP\_UNDEFINED\_13,
- TRAP\_UNDEFINED\_14,
- TRAP\_UNDEFINED\_15,

- TRAP\_UNDEFINED\_16,
- TRAP\_UNDEFINED\_17,
- TRAP\_UNDEFINED\_18,
- TRAP\_UNDEFINED\_19,
- TRAP\_UNDEFINED\_1A,
- TRAP\_UNDEFINED\_1B,
- TRAP\_UNDEFINED\_1C,
- TRAP\_UNDEFINED\_1D,
- TRAP\_UNDEFINED\_1E,
- TRAP\_UNDEFINED\_1F,
- TRAP\_GMRP,
- TRAP\_GVRP,
- TRAP\_UNDEF\_GARP\_22,
- TRAP\_UNDEF\_GARP\_23,
- TRAP\_UNDEF\_GARP\_24,
- TRAP\_UNDEF\_GARP\_25,
- TRAP\_UNDEF\_GARP\_26,
- TRAP\_UNDEF\_GARP\_27,
- TRAP\_UNDEF\_GARP\_28,
- TRAP\_UNDEF\_GARP\_29,
- TRAP\_UNDEF\_GARP\_2A,
- TRAP\_UNDEF\_GARP\_2B,
- TRAP\_UNDEF\_GARP\_2C,
- TRAP\_UNDEF\_GARP\_2D,
- TRAP\_UNDEF\_GARP\_2E,
- TRAP\_UNDEF\_GARP\_2F,
- TRAP\_CDP.
- TRAP\_CSSTP.
- TRAP\_LLDP.

(2) The RMA action is as following:

- RMA\_ACTION\_FORWARD
- RMA\_ACTION\_TRAP2CPU
- RMA\_ACTION\_DROP
- RMA\_ACTION\_FORWARD\_EXCLUDE\_CPU

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

---

---

## 21.17. rtk\_trap\_rmaKeepFormat\_set

`rtk_api_ret_t rtk_trap_rmaKeepFormat_set(rtk_trap_type_t type,  
rtk_enable_t enable)`

Set Reserved multicast address keep format configuration.

Defined in: trap.h

### Parameters

*type*

rma type.

*enable*

enable keep format.

### Comments

There are 48 types of Reserved Multicast Address frame for application usage.  
They are as following definition.

- TRAP\_BRG\_GROUP,
- TRAP\_FD\_PAUSE,
- TRAP\_SP\_MCAST,
- TRAP\_1X\_PAE,
- TRAP\_UNDEF\_BRG\_04,
- TRAP\_UNDEF\_BRG\_05,
- TRAP\_UNDEF\_BRG\_06,
- TRAP\_UNDEF\_BRG\_07,
- TRAP\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- TRAP\_UNDEF\_BRG\_09,
- TRAP\_UNDEF\_BRG\_0A,
- TRAP\_UNDEF\_BRG\_0B,
- TRAP\_UNDEF\_BRG\_0C,
- TRAP\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- TRAP\_8021AB,
- TRAP\_UNDEF\_BRG\_0F,
- TRAP\_BRG\_MNGEMENT,
- TRAP\_UNDEFINED\_11,
- TRAP\_UNDEFINED\_12,
- TRAP\_UNDEFINED\_13,
- TRAP\_UNDEFINED\_14,
- TRAP\_UNDEFINED\_15,
- TRAP\_UNDEFINED\_16,
- TRAP\_UNDEFINED\_17,
- TRAP\_UNDEFINED\_18,
- TRAP\_UNDEFINED\_19,
- TRAP\_UNDEFINED\_1A,
- TRAP\_UNDEFINED\_1B,
- TRAP\_UNDEFINED\_1C,

- TRAP\_UNDEFINED\_1D,  
- TRAP\_UNDEFINED\_1E,  
- TRAP\_UNDEFINED\_1F,  
- TRAP\_GMRP,  
- TRAP\_GVRP,  
- TRAP\_UNDEF\_GARP\_22,  
- TRAP\_UNDEF\_GARP\_23,  
- TRAP\_UNDEF\_GARP\_24,  
- TRAP\_UNDEF\_GARP\_25,  
- TRAP\_UNDEF\_GARP\_26,  
- TRAP\_UNDEF\_GARP\_27,  
- TRAP\_UNDEF\_GARP\_28,  
- TRAP\_UNDEF\_GARP\_29,  
- TRAP\_UNDEF\_GARP\_2A,  
- TRAP\_UNDEF\_GARP\_2B,  
- TRAP\_UNDEF\_GARP\_2C,  
- TRAP\_UNDEF\_GARP\_2D,  
- TRAP\_UNDEF\_GARP\_2E,  
- TRAP\_UNDEF\_GARP\_2F,  
- TRAP\_CDP.  
- TRAP\_CSSTP.  
- TRAP\_LLDP.

Return Codes	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_ENABLE	Invalid IFG parameter

## 21.18. rtk\_trap\_rmaKeepFormat\_get

**rtk\_api\_ret\_t rtk\_trap\_rmaKeepFormat\_get(rtk\_trap\_type\_t type,  
rtk\_enable\_t \*pEnable)**

Get Reserved multicast address action configuration.

Defined in: trap.h

Parameters	
<i>type</i>	rma type.
<i>*pEnable</i>	keep format status.

---

**Comments**

There are 48 types of Reserved Multicast Address frame for application usage.  
They are as following definition.

- TRAP\_BRG\_GROUP,
- TRAP\_FD\_PAUSE,
- TRAP\_SP\_MCAST,
- TRAP\_1X\_PAE,
- TRAP\_UNDEF\_BRG\_04,
- TRAP\_UNDEF\_BRG\_05,
- TRAP\_UNDEF\_BRG\_06,
- TRAP\_UNDEF\_BRG\_07,
- TRAP\_PROVIDER\_BRIDGE\_GROUP\_ADDRESS,
- TRAP\_UNDEF\_BRG\_09,
- TRAP\_UNDEF\_BRG\_0A,
- TRAP\_UNDEF\_BRG\_0B,
- TRAP\_UNDEF\_BRG\_0C,
- TRAP\_PROVIDER\_BRIDGE\_GVRP\_ADDRESS,
- TRAP\_8021AB,
- TRAP\_UNDEF\_BRG\_0F,
- TRAP\_BRG\_MNGEMENT,
- TRAP\_UNDEFINED\_11,
- TRAP\_UNDEFINED\_12,
- TRAP\_UNDEFINED\_13,
- TRAP\_UNDEFINED\_14,
- TRAP\_UNDEFINED\_15,
- TRAP\_UNDEFINED\_16,
- TRAP\_UNDEFINED\_17,
- TRAP\_UNDEFINED\_18,
- TRAP\_UNDEFINED\_19,
- TRAP\_UNDEFINED\_1A,
- TRAP\_UNDEFINED\_1B,
- TRAP\_UNDEFINED\_1C,
- TRAP\_UNDEFINED\_1D,
- TRAP\_UNDEFINED\_1E,
- TRAP\_UNDEFINED\_1F,
- TRAP\_GMRP,
- TRAP\_GVRP,
- TRAP\_UNDEF\_GARP\_22,
- TRAP\_UNDEF\_GARP\_23,
- TRAP\_UNDEF\_GARP\_24,
- TRAP\_UNDEF\_GARP\_25,
- TRAP\_UNDEF\_GARP\_26,
- TRAP\_UNDEF\_GARP\_27,
- TRAP\_UNDEF\_GARP\_28,
- TRAP\_UNDEF\_GARP\_29,
- TRAP\_UNDEF\_GARP\_2A,

- TRAP\_UNDEF\_GARP\_2B,
- TRAP\_UNDEF\_GARP\_2C,
- TRAP\_UNDEF\_GARP\_2D,
- TRAP\_UNDEF\_GARP\_2E,
- TRAP\_UNDEF\_GARP\_2F,
- TRAP\_CDP.
- TRAP\_CSSTP.
- TRAP\_LLDP.

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 21.19. rtk\_trap\_portUnknownMacPktAction\_set

`rtk_api_ret_t rtk_trap_portUnknownMacPktAction_set(rtk_port_t port,  
rtk_trap_unicast_action_t ucast_action)`

Set unknown source MAC packet action configuration.

Defined in: trap.h

*port*  
Port ID.

*ucast\_action*  
Unknown source MAC action

**Comments**  
This API can set unknown unicast packet action configuration

The unknown unicast action is as following:

- UCAST\_ACTION\_FORWARD\_PMASK
- UCAST\_ACTION\_DROP
- UCAST\_ACTION\_TRAP2CPU

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 21.20. rtk\_trap\_portUnknownMacPktAction\_get

`rtk_api_ret_t rtk_trap_portUnknownMacPktAction_get(rtk_port_t port,  
rtk_trap_icast_action_t *pUcast_action)`

Get unknown source MAC packet action configuration.

Defined in: trap.h

**Parameters**

*port*

Port ID.

*pUcast\_action*

Unknown source MAC action

**Comments**

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_INPUT

Invalid input parameters.

---

## 21.21. rtk\_trap\_portUnmatchMacPktAction\_set

`rtk_api_ret_t rtk_trap_portUnmatchMacPktAction_set(rtk_port_t port,  
rtk_trap_icast_action_t ucast_action)`

Set unmatch source MAC packet action configuration.

Defined in: trap.h

**Parameters**

*port*

Port ID.

*ucast\_action*

Unmatch source MAC action

**Comments**

This API can set unmatch unicast packet action configuration

The unmatch unicast action is as following:

- UCST\_ACTION\_FORWARD\_PMASK

- UCAST\_ACTION\_DROP
- UCAST\_ACTION\_TRAP2CPU

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 21.22. rtk\_trap\_portUnmatchMacPktAction\_get

```
rtk_api_ret_t rtk_trap_portUnmatchMacPktAction_get(rtk_port_t port,  
rtk_trap_icast_action_t *pUcast_action)
```

Get unmatch source MAC packet action configuration.

Defined in: trap.h

<b>Parameters</b>	<i>port</i> Port ID. <i>pUcast_action</i> Unmatch source MAC action
-------------------	--

### Comments

<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_SMI RT_ERR_INPUT	ok failed SMI access error Invalid input parameters.
---------------------	--	---

---

## 22. Module trunk.h - Unmanaged switch high-level API

Filename: trunk.h

---

**Description** The file includes Trunk module high-layer TRUNK definition  
Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

#### List of Symbols

Here is a list of all functions and variables in this module

trunk.h - Unmanaged switch high-level API  
rtk\_trunk\_port\_set  
rtk\_trunk\_port\_get  
rtk\_trunk\_distributionAlgorithm\_set  
rtk\_trunk\_distributionAlgorithm\_get  
rtk\_trunk\_queueEmptyStatus\_get  
rtk\_trunk\_trafficSeparate\_set  
rtk\_trunk\_trafficSeparate\_get  
rtk\_trunk\_mode\_set  
rtk\_trunk\_mode\_get  
rtk\_trunk\_trafficPause\_set  
rtk\_trunk\_trafficPause\_get  
rtk\_trunk\_hashMappingTable\_set  
rtk\_trunk\_hashMappingTable\_get  
rtk\_trunk\_portQueueEmpty\_get

---

## 22.1. rtk\_trunk\_port\_set

**rtk\_api\_ret\_t rtk\_trunk\_port\_set(rtк\_trunk\_group\_t *trk\_gid*,  
                                  rtк\_portmask\_t \**pTrunk\_member\_portmask*)**

Set trunking group available port mask

Defined in: trunk.h

**Parameters**

*trk\_gid*  
    trunk group id  
*\*pTrunk\_member\_portmask*  
    Logic trunking member port mask

**Comments**

The API can set port trunking group port mask. Each port trunking group has max 4 ports. If enabled port mask has less than 2 ports available setting, then this trunking group function is disabled.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error

RT_ERR_LA_TRUNK_ID	Invalid trunking group
RT_ERR_PORT_MASK	Invalid portmask.

## 22.2. rtk\_trunk\_port\_get

**rtk\_api\_ret\_t rtk\_trunk\_port\_get(rtk\_trunk\_group\_t *trk\_gid*,  
rtk\_portmask\_t \**pTrunk\_member\_portmask*)**

Get trunking group available port mask

Defined in: trunk.h

**Parameters**

*trk\_gid*  
trunk group id  
*\*pTrunk\_member\_portmask*  
Logic trunking member port mask

**Comments**

The API can get 2 port trunking group.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_LA_TRUNK_ID	Invalid trunking group

## 22.3. rtk\_trunk\_distributionAlgorithm\_set

**rtk\_api\_ret\_t rtk\_trunk\_distributionAlgorithm\_set(rtk\_trunk\_group\_t  
*trk\_gid*, rtk\_uint32 *algo\_bitmask*)**

Set port trunking hash select sources

Defined in: trunk.h

**Parameters**

*trk\_gid*  
trunk group id  
*algo\_bitmask*  
Bitmask of the distribution algorithm

**Comments**

The API can set port trunking hash algorithm sources. 7 bits mask for link aggregation group0 hash parameter selection {DIP, SIP, DMAC, SMAC, SPA} - 0b0000001: SPA

- 
- 0b0000010: SMAC
  - 0b0000100: DMAC
  - 0b0001000: SIP
  - 0b0010000: DIP
  - 0b0100000: TCP/UDP Source Port
  - 0b1000000: TCP/UDP Destination Port Example:
  - 0b0000011: SMAC & SPA
  - Note that it could be an arbitrary combination or independent set

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_LA_TRUNK_ID	Invalid trunking group
	RT_ERR_LA_HASHMASK	Hash algorithm selection error.
	RT_ERR_PORT_MASK	Invalid portmask.

---

## 22.4. rtk\_trunk\_distributionAlgorithm\_get

`rtk_api_ret_t rtk_trunk_distributionAlgorithm_get(rtk_trunk_group_t  
trk_gid, rtk_uint32 *pAlgo_bitmask)`

Get port trunking hash select sources

Defined in: trunk.h

<b>Parameters</b>	<i>trk_gid</i> trunk group id
	<i>*pAlgo_bitmask</i> Bitmask of the distribution algorithm

**Comments** The API can get port trunking hash algorithm sources.

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_LA_TRUNK_ID	Invalid trunking group

---

## 22.5.rtk\_trunk\_queueEmptyStatus\_get

`rtk_api_ret_t rtk_trunk_queueEmptyStatus_get(rtk_portmask_t *pPortmask)`

Get current output queue if empty status

Defined in: trunk.h

**Parameters**

*\*pPortmask*

trunk group id

**Comments**

The API can get queues are empty port mask

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

---

## 22.6.rtk\_trunk\_trafficSeparate\_set

`rtk_api_ret_t rtk_trunk_trafficSeparate_set(rtk_trunk_group_t trk_gid, rtk_trunk_separateType_t separateType)`

Set the traffic separation setting of a trunk group from the specified device.

Defined in: trunk.h

**Parameters**

*trk\_gid*

trunk group id

*separateType*

traffic separation setting

**Comments**

SEPARATE\_NONE: disable traffic separation SEPARATE\_FLOOD: trunk MSB link up port is dedicated to TX flooding (L2 lookup miss) traffic

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_UNIT\_ID

invalid unit id

RT\_ERR\_LA\_TRUNK\_ID

invalid trunk ID

RT\_ERR\_LA\_HASHMASK

invalid hash mask

---

---

## 22.7. rtk\_trunk\_trafficSeparate\_get

`rtk_api_ret_t rtk_trunk_trafficSeparate_get(rtk_trunk_group_t trk_gid,  
rtk_trunk_separateType_t *pSeparateType)`

Get the traffic separation setting of a trunk group from the specified device.

Defined in: `trunk.h`

**Parameters**

*trk\_gid*  
trunk group id  
*\*pSeparateType*  
pointer separated traffic type

**Comments**

SEPARATE\_NONE: disable traffic separation SEPARATE\_FLOOD: trunk MSB link up port is dedicated to TX flooding (L2 lookup miss) traffic

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_UNIT_ID</code>	invalid unit id
<code>RT_ERR_LA_TRUNK_ID</code>	invalid trunk ID
<code>RT_ERR_NULL_POINTER</code>	input parameter may be null pointer

---

## 22.8. rtk\_trunk\_mode\_set

`rtk_api_ret_t rtk_trunk_mode_set(rtk_trunk_mode_t mode)`

Set the trunk mode to the specified device.

Defined in: `trunk.h`

**Parameters**

*mode*  
trunk mode

**Comments**

The enum of the trunk mode as following

- `TRUNK_MODE_NORMAL`
- `TRUNK_MODE_DUMB`

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_INPUT</code>	invalid input parameter

## 22.9. rtk\_trunk\_mode\_get

```
rtk_api_ret_t rtk_trunk_mode_get(rtk_trunk_mode_t *pMode)
```

Get the trunk mode from the specified device.

Defined in: trunk.h

<b>Parameters</b>	<i>*pMode</i> pointer buffer of trunk mode
<b>Comments</b>	The enum of the trunk mode as following - TRUNK_MODE_NORMAL - TRUNK_MODE_DUMB
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_NULL_POINTER input parameter may be null pointer

## 22.10. rtk trunk trafficPause set

Set the traffic pause setting of a trunk group.

Defined in: trunk.h

<b>Parameters</b>	<i>trk_gid</i> trunk group id <i>enable</i> traffic pause state
<b>Comments</b>	None.
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_LA_TRUNK_ID invalid trunk ID

---

---

## 22.11. rtk\_trunk\_trafficPause\_get

`rtk_api_ret_t rtk_trunk_trafficPause_get(rtk_trunk_group_t trk_gid,  
rtk_enable_t *pEnable)`

Get the traffic pause setting of a trunk group.

Defined in: `trunk.h`

**Parameters**

*trk\_gid*  
trunk group id  
*\*pEnable*  
pointer of traffic pause state.

**Comments**

None.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_LA_TRUNK_ID</code>	invalid trunk ID
<code>RT_ERR_NULL_POINTER</code>	input parameter may be null pointer

---

## 22.12. rtk\_trunk\_hashMappingTable\_set

`rtk_api_ret_t rtk_trunk_hashMappingTable_set(rtk_trunk_group_t trk_gid,  
rtk_trunk_hashVal2Port_t *pHash2Port_array)`

Set hash value to port array in the trunk group id from the specified device.

Defined in: `trunk.h`

**Parameters**

*trk\_gid*  
trunk group id  
*\*pHash2Port\_array*  
ports associate with the hash value

**Comments**

None.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_UNIT_ID</code>	invalid unit id
<code>RT_ERR_LA_TRUNK_ID</code>	invalid trunk ID
<code>RT_ERR_NULL_POINTER</code>	input parameter may be null pointer

RT_ERR_LA_TRUNK_NOT_EXIST	the trunk doesn't exist
RT_ERR_LA_NOT_MEMBER_PORT	the port is not a member port of the trunk
RT_ERR_LA_CPUPORT	CPU port can not be aggregated port

---

## 22.13. rtk\_trunk\_hashMappingTable\_get

`rtk_api_ret_t rtk_trunk_hashMappingTable_get(rtk_trunk_group_t trk_gid,  
rtk_trunk_hashVal2Port_t *pHash2Port_array)`

Get hash value to port array in the trunk group id from the specified device.

Defined in: trunk.h

**Parameters**

*trk\_gid*  
trunk group id  
*\*pHash2Port\_array*  
pointer buffer of ports associate with the hash value

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_UNIT_ID	invalid unit id
RT_ERR_LA_TRUNK_ID	invalid trunk ID
RT_ERR_NULL_POINTER	input parameter may be null pointer

---

## 22.14. rtk\_trunk\_portQueueEmpty\_get

`rtk_api_ret_t rtk_trunk_portQueueEmpty_get(rtk_portmask_t  
*pEmpty_portmask)`

Get the port mask which all queues are empty.

Defined in: trunk.h

**Parameters**

*\*pEmpty\_portmask*  
pointer empty port mask

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	input parameter may be null pointer

---

## 23. Module `vlan.h` - Unmanaged switch high-level API

Filename: `vlan.h`

**Description** The file includes Trap module high-layer VLAN definition

Copyright © 2013 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

`vlan.h` - Unmanaged switch high-level API  
`rtk_vlan_init`  
`rtk_vlan_set`  
`rtk_vlan_get`  
`rtk_vlan_egressFilterEnable_set`  
`rtk_vlan_egressFilterEnable_get`  
`rtk_vlan_mbrCfg_set`  
`rtk_vlan_mbrCfg_get`  
`rtk_vlan_portPvid_set`  
`rtk_vlan_portPvid_get`  
`rtk_vlan_portIgrFilterEnable_set`  
`rtk_vlan_portIgrFilterEnable_get`  
`rtk_vlan_portAcceptFrameType_set`  
`rtk_vlan_portAcceptFrameType_get`  
`rtk_vlan_tagMode_set`  
`rtk_vlan_tagMode_get`  
`rtk_vlan_transparent_set`  
`rtk_vlan_transparent_get`  
`rtk_vlan_keep_set`  
`rtk_vlan_keep_get`  
`rtk_vlan_stg_set`  
`rtk_vlan_stg_get`  
`rtk_vlan_protoAndPortBasedVlan_add`  
`rtk_vlan_protoAndPortBasedVlan_get`  
`rtk_vlan_protoAndPortBasedVlan_del`  
`rtk_vlan_protoAndPortBasedVlan_delAll`

```
rtk_vlan_portFid_set  
rtk_vlan_portFid_get  
rtk_vlan_UntagDscpPriorityEnable_set  
rtk_vlan_UntagDscpPriorityEnable_get  
rtk_stp_mstpState_set  
rtk_stp_mstpState_get  
rtk_vlan_checkAndCreateMbr  
rtk_vlan_reservedVidAction_set  
rtk_vlan_reservedVidAction_get  
rtk_vlan_realKeepRemarkEnable_set  
rtk_vlan_realKeepRemarkEnable_get  
rtk_vlan_reset
```

---

### 23.1. rtk\_vlan\_init

`rtk_api_ret_t rtk_vlan_init( void )`

Initialize VLAN.

Defined in: `vlan.h`

**Parameters**

`void`

**Comments**

VLAN is disabled by default. User has to call this API to enable VLAN before using it. And It will set a default VLAN(vid 1) including all ports and set all ports PVID to the default VLAN.

**Return Codes**

<code>RT_ERR_OK</code>	ok
<code>RT_ERR_FAILED</code>	failed
<code>RT_ERR_SMI</code>	SMI access error

---

### 23.2. rtk\_vlan\_set

`rtk_api_ret_t rtk_vlan_set(rtk_vlan_t vid, rtk_vlan_cfg_t *pVlanCfg)`

Set a VLAN entry.

Defined in: `vlan.h`

**Parameters**

---

<i>vid</i>	VLAN ID to configure.
<i>*pVlanCfg</i>	VLAN Configuration
<b>Comments</b>	
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_L2_FID	Invalid FID.
RT_ERR_VLAN_PORT_MBR_EXIST	Invalid member port mask.
RT_ERR_VLAN_VID	Invalid VID parameter.

---

### 23.3. rtk\_vlan\_get

`rtk_api_ret_t rtk_vlan_get(rtk_vlan_t vid, rtk_vlan_cfg_t *pVlanCfg)`

<i>vid</i>	Get a VLAN entry.
Defined in: vlan.h	
<b>Parameters</b>	
<i>vid</i>	VLAN ID to configure.
<i>*pVlanCfg</i>	VLAN Configuration
<b>Comments</b>	
<b>Return Codes</b>	
RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_VLAN_VID	Invalid VID parameter.

---

### 23.4. rtk\_vlan\_egressFilterEnable\_set

`rtk_api_ret_t rtk_vlan_egressFilterEnable_set(rtk_enable_t egrFilter)`

	Set VLAN egress filter.
	Defined in: vlan.h
<b>Parameters</b>	<i>egrFilter</i> Egress filtering
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_ENABLE Invalid input parameters.

---

### 23.5.rtk\_vlan\_egressFilterEnable\_get

`rtk_api_ret_t rtk_vlan_egressFilterEnable_get(rtk_enable_t *pEgrFilter)`

	Get VLAN egress filter.
	Defined in: vlan.h
<b>Parameters</b>	<i>*pEgrFilter</i> Egress filtering
<b>Comments</b>	
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_NULL_POINTER NULL Pointer.

---

### 23.6.rtk\_vlan\_mbrCfg\_set

`rtk_api_ret_t rtk_vlan_mbrCfg_set(rtk_uint32 idx, rtk_vlan_mbrcfg_t *pMbrcfg)`

	Set a VLAN Member Configuration entry by index.
	Defined in: vlan.h
<b>Parameters</b>	

---

<i>idx</i>	Index of VLAN Member Configuration.										
<i>*pMbrcfg</i>	VLAN member Configuration.										
<b>Comments</b>	Set a VLAN Member Configuration entry by index.										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> <tr> <td>RT_ERR_VLAN_VID</td><td>Invalid VID parameter.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_VLAN_VID	Invalid VID parameter.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_INPUT	Invalid input parameters.										
RT_ERR_VLAN_VID	Invalid VID parameter.										

---

### 23.7. rtk\_vlan\_mbrCfg\_get

`rtk_api_ret_t rtk_vlan_mbrCfg_get(rtk_uint32 idx, rtk_vlan_mbrcfg_t *pMbrcfg)`

Get a VLAN Member Configuration entry by index.

Defined in: vlan.h

<i>idx</i>	Index of VLAN Member Configuration.										
<i>*pMbrcfg</i>	VLAN member Configuration.										
<b>Comments</b>	Get a VLAN Member Configuration entry by index.										
<b>Return Codes</b>	<table> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Invalid input parameters.</td></tr> <tr> <td>RT_ERR_VLAN_VID</td><td>Invalid VID parameter.</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Invalid input parameters.	RT_ERR_VLAN_VID	Invalid VID parameter.
RT_ERR_OK	ok										
RT_ERR_FAILED	failed										
RT_ERR_SMI	SMI access error										
RT_ERR_INPUT	Invalid input parameters.										
RT_ERR_VLAN_VID	Invalid VID parameter.										

---

### 23.8. rtk\_vlan\_portPvid\_set

`rtk_api_ret_t rtk_vlan_portPvid_set(rtk_port_t port, rtk_vlan_t pvid, rtk_pri_t priority)`

	Set port to specified VLAN ID(PVID).															
	Defined in: vlan.h															
<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>pvid</i> Specified VLAN ID.</p> <p><i>priority</i> 802.1p priority for the PVID.</p>															
<b>Comments</b>	The API is used for Port-based VLAN. The untagged frame received from the port will be classified to the specified VLAN and assigned to the specified priority.															
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> <tr> <td>RT_ERR_SMI</td> <td>SMI access error</td> </tr> <tr> <td>RT_ERR_PORT_ID</td> <td>Invalid port number.</td> </tr> <tr> <td>RT_ERR_VLAN_PRIORITY</td> <td>Invalid priority.</td> </tr> <tr> <td>RT_ERR_VLAN_ENTRY_NOT_FOUND</td> <td>VLAN entry not found.</td> </tr> <tr> <td>RT_ERR_VLAN_VID</td> <td>Invalid VID parameter.</td> </tr> </table>		RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_PORT_ID	Invalid port number.	RT_ERR_VLAN_PRIORITY	Invalid priority.	RT_ERR_VLAN_ENTRY_NOT_FOUND	VLAN entry not found.	RT_ERR_VLAN_VID	Invalid VID parameter.
RT_ERR_OK	ok															
RT_ERR_FAILED	failed															
RT_ERR_SMI	SMI access error															
RT_ERR_PORT_ID	Invalid port number.															
RT_ERR_VLAN_PRIORITY	Invalid priority.															
RT_ERR_VLAN_ENTRY_NOT_FOUND	VLAN entry not found.															
RT_ERR_VLAN_VID	Invalid VID parameter.															

### 23.9. rtk\_vlan\_portPvid\_get

```
rtk_api_ret_t rtk_vlan_portPvid_get(rtk_port_t port, rtk_vlan_t *pPvid,
                                     rtk_pri_t *pPriority)
```

Get VLAN ID(PVID) on specified port.

	Defined in: vlan.h					
<b>Parameters</b>	<p><i>port</i> Port id.</p> <p><i>*pPvid</i> Specified VLAN ID.</p> <p><i>*pPriority</i> 802.1p priority for the PVID.</p>					
<b>Comments</b>	The API can get the PVID and 802.1p priority for the PVID of Port-based VLAN.					
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td> <td>ok</td> </tr> <tr> <td>RT_ERR_FAILED</td> <td>failed</td> </tr> </table>		RT_ERR_OK	ok	RT_ERR_FAILED	failed
RT_ERR_OK	ok					
RT_ERR_FAILED	failed					

---

RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	invalid port number.

---

### 23.10. rtk\_vlan\_portIgrFilterEnable\_set

`rtk_api_ret_t rtk_vlan_portIgrFilterEnable_set(rtk_port_t port, rtk_enable_t igr_filter)`

Set VLAN ingress for each port.

Defined in: vlan.h

**Parameters**

*port*  
Port id.

*igr\_filter*  
VLAN ingress function enable status.

**Comments**

The status of VLAN ingress filter is as following:

- DISABLED
- ENABLED While VLAN function is enabled, ASIC will decide VLAN ID for each received frame and get belonged member ports from VLAN table. If received port is not belonged to VLAN member ports, ASIC will drop received frame if VLAN ingress function is enabled.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number
RT_ERR_ENABLE	Invalid enable input

---

### 23.11. rtk\_vlan\_portIgrFilterEnable\_get

`rtk_api_ret_t rtk_vlan_portIgrFilterEnable_get(rtk_port_t port, rtk_enable_t *pIgr_filter)`

Get VLAN Ingress Filter

Defined in: vlan.h

<b>Parameters</b>	<i>port</i> Port id.  <i>*pIgr_filter</i> VLAN ingress function enable status.
<b>Comments</b>	The API can Get the VLAN ingress filter status. The status of vlan ingress filter is as following: - DISABLED - ENABLED
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_INPUT Invalid input parameters. RT_ERR_PORT_ID Invalid port number.

---

### 23.12. rtk\_vlan\_portAcceptFrameType\_set

`rtk_api_ret_t rtk_vlan_portAcceptFrameType_set(rtk_port_t port,  
rtk_vlan_acceptFrameType_t accept_frame_type)`

Set VLAN accept\_frame\_type

Defined in: vlan.h

<b>Parameters</b>	<i>port</i> Port id.  <i>accept_frame_type</i> accept frame type
<b>Comments</b>	The API is used for checking 802.1Q tagged frames. The accept frame type as following: - ACCEPT_FRAME_TYPE_ALL - ACCEPT_FRAME_TYPE_TAG_ONLY - ACCEPT_FRAME_TYPE_UNTAG_ONLY
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port number. RT_ERR_VLAN_ACCEPT_FRAME_TYPE_INVALID Invalid frame type.

---

---

### 23.13. rtk\_vlan\_portAcceptFrameType\_get

`rtk_api_ret_t rtk_vlan_portAcceptFrameType_get(rtk_port_t port,  
rtk_vlan_acceptFrameType_t *pAccept_frame_type)`

Get VLAN accept\_frame\_type

Defined in: vlan.h

**Parameters**

*port*

Port id.

*\*pAccept\_frame\_type*

accept frame type

**Comments**

The API can Get the VLAN ingress filter. The accept frame type as following:

- ACCEPT\_FRAME\_TYPE\_ALL
- ACCEPT\_FRAME\_TYPE\_TAG\_ONLY
- ACCEPT\_FRAME\_TYPE\_UNTAG\_ONLY

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_INPUT

Invalid input parameters.

RT\_ERR\_PORT\_ID

Invalid port number.

---

### 23.14. rtk\_vlan\_tagMode\_set

`rtk_api_ret_t rtk_vlan_tagMode_set(rtk_port_t port, rtk_vlan_tagMode_t  
tag_mode)`

Set CVLAN egress tag mode

Defined in: vlan.h

**Parameters**

*port*

Port id.

*tag\_mode*

The egress tag mode.

**Comments**

The API can set Egress tag mode. There are 4 mode for egress tag:

- VLAN\_TAG\_MODE\_ORIGINAL,
- VLAN\_TAG\_MODE\_KEEP\_FORMAT,

- VLAN\_TAG\_MODE\_PRI.
- VLAN\_TAG\_MODE\_REAL\_KEEP\_FORMAT,

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_PORT_ID	Invalid port number.
	RT_ERR_INPUT	Invalid input parameter.
	RT_ERR_ENABLE	Invalid enable input.

### 23.15. rtk\_vlan\_tagMode\_get

```
rtk_api_ret_t rtk_vlan_tagMode_get(rtk_port_t port, rtk_vlan_tagMode_t
*pTag_mode)
```

Get CVLAN egress tag mode

Defined in: wlan.h

<b>Parameters</b>	<i>port</i>	Port id.
	<i>*pTag_mode</i>	The egress tag mode.

<b>Comments</b>	The API can get Egress tag mode. There are 4 mode for egress tag:	
	-	VLAN_TAG_MODE_ORIGINAL,
	-	VLAN_TAG_MODE_KEEP_FORMAT,
	-	VLAN_TAG_MODE_PRI.
	-	VLAN_TAG_MODE_REAL_KEEP_FORMAT,

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.
	RT_ERR_PORT_ID	Invalid port number.

---

---

## 23.16. rtk\_vlan\_transparent\_set

`rtk_api_ret_t rtk_vlan_transparent_set(rtk_port_t egr_port, rtk_portmask_t *pIgr_pmask)`

Set VLAN transparent mode

Defined in: vlan.h

**Parameters**

*egr\_port*  
Egress Port id.  
*\*pIgr\_pmask*  
Ingress Port Mask.

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port number.

---

## 23.17. rtk\_vlan\_transparent\_get

`rtk_api_ret_t rtk_vlan_transparent_get(rtk_port_t egr_port, rtk_portmask_t *pIgr_pmask)`

Get VLAN transparent mode

Defined in: vlan.h

**Parameters**

*egr\_port*  
Egress Port id.  
*\*pIgr\_pmask*  
Ingress Port Mask

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.

RT_ERR_PORT_ID	Invalid port number.
----------------	----------------------

### 23.18. rtk\_vlan\_keep\_set

**rtk\_api\_ret\_t rtk\_vlan\_keep\_set(rtk\_port\_t egr\_port, rtk\_portmask\_t \*pIgr\_pmask)**

Set VLAN egress keep mode

Defined in: vlan.h

**Parameters**

*egr\_port*  
Egress Port id.

*\*pIgr\_pmask*  
Ingress Port Mask.

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	Invalid input parameters.
RT_ERR_PORT_ID	Invalid port number.

### 23.19. rtk\_vlan\_keep\_get

**rtk\_api\_ret\_t rtk\_vlan\_keep\_get(rtk\_port\_t egr\_port, rtk\_portmask\_t \*pIgr\_pmask)**

Get VLAN egress keep mode

Defined in: vlan.h

**Parameters**

*egr\_port*  
Egress Port id.

*\*pIgr\_pmask*  
Ingress Port Mask

**Comments**

None.

**Return Codes**

RT_ERR_OK	ok
-----------	----

---

RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_INPUT	invalid input parameters.
RT_ERR_PORT_ID	Invalid port number.

---

## 23.20. rtk\_vlan\_stg\_set

**rtk\_api\_ret\_t rtk\_vlan\_stg\_set(rtк\_vlan\_t vid, rtк\_stp\_msti\_id\_t stg)**

Set spanning tree group instance of the vlan to the specified device

Defined in: vlan.h

**Parameters**

*vid*  
Specified VLAN ID.

*stg*  
spanning tree group instance.

**Comments**

The API can set spanning tree group instance of the vlan to the specified device.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_MSTI	Invalid msti parameter
RT_ERR_INPUT	Invalid input parameter.
RT_ERR_VLAN_VID	Invalid VID parameter.

## 23.21. rtk\_vlan\_stg\_get

**rtk\_api\_ret\_t rtk\_vlan\_stg\_get(rtк\_vlan\_t vid, rtк\_stp\_msti\_id\_t \*pStg)**

Get spanning tree group instance of the vlan to the specified device

Defined in: vlan.h

**Parameters**

*vid*  
Specified VLAN ID.

*\*pStg*  
spanning tree group instance.

<b>Comments</b>	The API can get spanning tree group instance of the vlan to the specified device.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.
	RT_ERR_VLAN_VID	Invalid VID parameter.

---

## 23.22. rtk\_vlan\_protoAndPortBasedVlan\_add

**rtk\_api\_ret\_t rtk\_vlan\_protoAndPortBasedVlan\_add(rtk\_port\_t port,  
rtk\_vlan\_protoAndPortInfo\_t \*pInfo)**

Add the protocol-and-port-based vlan to the specified port of device.

Defined in: vlan.h

### Parameters

*port*  
Port id.

*\*pInfo*  
Protocol and port based VLAN configuration information.

### Comments

The incoming packet which match the protocol-and-port-based vlan will use the configure vid for ingress pipeline The frame type is shown in the following:

- FRAME\_TYPE\_ETHERNET
- FRAME\_TYPE\_RFC1042
- FRAME\_TYPE\_LLCOOTHER

### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_VLAN_VID	Invalid VID parameter.
RT_ERR_VLAN_PRIORITY	Invalid priority.
RT_ERR_TBL_FULL	Table is full.
RT_ERR_OUT_OF_RANGE	input out of range.

---

---

### 23.23. rtk\_vlan\_protoAndPortBasedVlan\_get

```
rtk_api_ret_t rtk_vlan_protoAndPortBasedVlan_get(rtk_port_t port,  
rtk_vlan_proto_type_t proto_type, rtk_vlan_protoVlan_frameType_t  
frame_type, rtk_vlan_protoAndPortInfo_t *pInfo)
```

Get the protocol-and-port-based vlan to the specified port of device.

Defined in: vlan.h

**Parameters**

*port*  
Port id.  
*proto\_type*  
protocol  
*frame\_type*  
protocol  
*\*pInfo*

Protocol and port based VLAN configuration information.

**Comments**

The incoming packet which match the protocol-and-port-based vlan will use the configure vid for ingress pipeline The frame type is shown in the following:

- FRAME\_TYPE\_ETHERNET
- FRAME\_TYPE\_RFC1042
- FRAME\_TYPE\_LLCOOTHER

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_OUT_OF_RANGE	input out of range.
RT_ERR_TBL_FULL	Table is full.

---

### 23.24. rtk\_vlan\_protoAndPortBasedVlan\_del

```
rtk_api_ret_t rtk_vlan_protoAndPortBasedVlan_del(rtk_port_t port,  
rtk_vlan_proto_type_t proto_type, rtk_vlan_protoVlan_frameType_t  
frame_type)
```

Delete the protocol-and port-based vlan from the specified port of device.

Defined in: vlan.h

<b>Parameters</b>	<i>port</i> Port id. <i>proto_type</i> protocol <i>frame_type</i> protocol
<b>Comments</b>	The incoming packet which match the protocol-and-port-based vlan will use the configure vid for ingress pipeline The frame type is shown in the following: - FRAME_TYPE_ETHERNET - FRAME_TYPE_RFC1042 - FRAME_TYPE_LLCOOTHER
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port number. RT_ERR_OUT_OF_RANGE input out of range. RT_ERR_TBL_FULL Table is full.

---

### 23.25. rtk\_vlan\_protoAndPortBasedVlan\_delAll

`rtk_api_ret_t rtk_vlan_protoAndPortBasedVlan_delAll(rtк_port_t port)`

Delete all protocol-and-port-based vlans from the specified port of device.

Defined in: `vlan.h`

<b>Parameters</b>	<i>port</i> Port id.
<b>Comments</b>	The incoming packet which match the protocol-and-port-based vlan will use the configure vid for ingress pipeline Delete all flow table protocol-and-port-based vlan entries.
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed RT_ERR_SMI SMI access error RT_ERR_PORT_ID Invalid port number. RT_ERR_OUT_OF_RANGE input out of range.

---

---

## 23.26. rtk\_vlan\_portFid\_set

`rtk_api_ret_t rtk_vlan_portFid_set(rtk_port_t port, rtk_enable_t enable,  
rtk_fid_t fid)`

Set port-based filtering database

Defined in: vlan.h

**Parameters**

*port*

Port id.

*enable*

enable port

*fid*

Specified filtering database.

**Comments**

The API can set port-based filtering database. If the function is enabled, all input packets will be assigned to the port-based fid regardless vlan tag.

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_L2\_FID

Invalid fid.

RT\_ERR\_INPUT

Invalid input parameter.

RT\_ERR\_PORT\_ID

Invalid port ID.

---

## 23.27. rtk\_vlan\_portFid\_get

`rtk_api_ret_t rtk_vlan_portFid_get(rtk_port_t port, rtk_enable_t *pEnable,  
rtk_fid_t *pFid)`

Get port-based filtering database

Defined in: vlan.h

**Parameters**

*port*

Port id.

*\*pEnable*

enable port

*\*pFid*

Specified filtering database.

<b>Comments</b>	The API can get port-based filtering database status. If the function is enabled, all input packets will be assigned to the port-based fid regardless vlan tag.	
<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Invalid input parameters.
	RT_ERR_PORT_ID	Invalid port ID.

---

### 23.28. rtk\_vlan\_UntagDscpPriorityEnable\_set

`rtk_api_ret_t rtk_vlan_UntagDscpPriorityEnable_set(rtk_enable_t enable)`

Set Untag DSCP priority assign

Defined in: `vlan.h`

<b>Parameters</b>	<i>enable</i>	
		state of Untag DSCP priority assign

#### Comments

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_ENABLE	Invalid input parameters.

---

### 23.29. rtk\_vlan\_UntagDscpPriorityEnable\_get

`rtk_api_ret_t rtk_vlan_UntagDscpPriorityEnable_get(rtk_enable_t *pEnable)`

Get Untag DSCP priority assign

Defined in: `vlan.h`

<b>Parameters</b>	<i>*pEnable</i>	
		state of Untag DSCP priority assign

#### Comments

#### Return Codes

---

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_NULL_POINTER	Null pointer

---

### 23.30. rtk\_stp\_mstpState\_set

`rtk_api_ret_t rtk_stp_mstpState_set(rtk_stp_msti_id_t msti, rtk_port_t port, rtk_stp_state_t stp_state)`

Configure spanning tree state per each port.

Defined in: vlan.h

#### Parameters

*msti*  
Port id  
*port*  
Multiple spanning tree instance.  
*stp\_state*  
Spanning tree state for msti

#### Comments

System supports per-port multiple spanning tree state for each msti. There are four states supported by ASIC.

- STP\_STATE\_DISABLED
- STP\_STATE\_BLOCKING
- STP\_STATE\_LEARNING
- STP\_STATE\_FORWARDING

#### Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_SMI	SMI access error
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_MSTI	Invalid msti parameter.
RT_ERR_MSTP_STATE	Invalid STP state.

---

### 23.31. rtk\_stp\_mstpState\_get

```
rtk_api_ret_t rtk_stp_mstpState_get(rtk_stp_msti_id_t msti, rtk_port_t port,  
rtk_stp_state_t *pStp_state)
```

Get spanning tree state per each port.

Defined in: vlan.h

**Parameters**

*msti*

Port id.

*port*

Multiple spanning tree instance.

*\*pStp\_state*

Spanning tree state for msti

**Comments**

System supports per-port multiple spanning tree state for each msti. There are four states supported by ASIC.

- STP\_STATE\_DISABLED
- STP\_STATE\_BLOCKING
- STP\_STATE\_LEARNING
- STP\_STATE\_FORWARDING

**Return Codes**

RT\_ERR\_OK

ok

RT\_ERR\_FAILED

failed

RT\_ERR\_SMI

SMI access error

RT\_ERR\_PORT\_ID

Invalid port number.

RT\_ERR\_MSTI

Invalid msti parameter.

---

### 23.32. rtk\_vlan\_checkAndCreateMbr

```
rtk_api_ret_t rtk_vlan_checkAndCreateMbr(rtk_vlan_t vid, rtk_uint32  
*pIndex)
```

Check and create Member configuration and return index

Defined in: vlan.h

**Parameters**

*vid*

VLAN id.

*\*pIndex*  
Member configuration index

## Comments

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_VLAN_VID	Invalid VLAN ID.
	RT_ERR_VLAN_ENTRY_NOT_FOUND	VLAN not found
	RT_ERR_TBL_FULL	Member Configuration table full

### 23.33. rtk\_vlan\_reservedVidAction set

```
rtk_api_ret_t rtk_vlan_reservedVidAction_set(rtк_vlan_resVidAction_t  
action_vid0, rtк_vlan_resVidAction_t action_vid4095)
```

Set Action of VLAN ID = 0 & 4095 tagged packet

Defined in: `vlan.h`

*action\_vid0*

## *action\_vid4095*

## Comments

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

### **23.34. rtk vlan reservedVidAction get**

```
rtk_api_ret_t rtk_vlan_reservedVidAction_get(rtk_vlan_resVidAction_t  
*pAction_vid0, rtk_vlan_resVidAction_t *pAction_vid4095)
```

Get Action of VLAN ID = 0 & 4095 tagged packet

	Defined in: vlan.h								
<b>Parameters</b>	<p><i>*pAction_vid0</i> Action for VID 0.</p> <p><i>*pAction_vid4095</i> Action for VID 4095.</p>								
<b>Comments</b>									
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_NULL_POINTER</td><td>NULL Pointer</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_NULL_POINTER	NULL Pointer
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_NULL_POINTER	NULL Pointer								

---

### 23.35. rtk\_vlan\_realKeepRemarkEnable\_set

`rtk_api_ret_t rtk_vlan_realKeepRemarkEnable_set(rtk_enable_t enabled)`

Set Real keep 1p remarking feature

Defined in: vlan.h
<i>enabled</i> State of 1p remarking at real keep packet

<b>Parameters</b>									
<b>Comments</b>									
<b>Return Codes</b>	<table border="0"> <tr> <td>RT_ERR_OK</td><td>ok</td></tr> <tr> <td>RT_ERR_FAILED</td><td>failed</td></tr> <tr> <td>RT_ERR_SMI</td><td>SMI access error</td></tr> <tr> <td>RT_ERR_INPUT</td><td>Error Input</td></tr> </table>	RT_ERR_OK	ok	RT_ERR_FAILED	failed	RT_ERR_SMI	SMI access error	RT_ERR_INPUT	Error Input
RT_ERR_OK	ok								
RT_ERR_FAILED	failed								
RT_ERR_SMI	SMI access error								
RT_ERR_INPUT	Error Input								

---

### 23.36. rtk\_vlan\_realKeepRemarkEnable\_get

`rtk_api_ret_t rtk_vlan_realKeepRemarkEnable_get(rtk_enable_t *pEnabled)`

Get Real keep 1p remarking feature

Defined in: vlan.h
<i>*pEnabled</i> State of 1p remarking at real keep packet

---

**Comments**

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

---

**23.37. rtk\_vlan\_reset**

`rtk_api_ret_t rtk_vlan_reset( void)`

Reset VLAN

Defined in: vlan.h

**Parameters**

`void`

**Comments**

<b>Return Codes</b>	RT_ERR_OK	ok
	RT_ERR_FAILED	failed
	RT_ERR_SMI	SMI access error
	RT_ERR_INPUT	Error Input

---

**24. Module i2c.h - Unmanaged switch high-level API**

Filename: i2c.h

**Description**

The file includes IGMP module high-layer API definition

Copyright © 2009 Realtek™ Semiconductor Corp. All rights reserved.

List of Symbols

Here is a list of all functions and variables in this module

i2c.h - Unmanaged switch high-level API  
rtk\_i2c\_data\_read

rtk\_i2c\_data\_write  
rtk\_i2c\_init  
rtk\_i2c\_mode\_set  
rtk\_i2c\_mode\_get  
rtk\_i2c\_gpioPinGroup\_set  
rtk\_i2c\_gpioPinGroup\_get

---

## 24.1. rtk\_i2c\_data\_read

**rtk\_api\_ret\_t rtk\_i2c\_data\_read(rtk\_uint8 deviceAddr, rtk\_uint32 slaveRegAddr, rtk\_uint32 \*pRegData)**

read i2c slave device register.

Defined in: i2c.h

### Parameters

*deviceAddr*  
access Slave device address  
*slaveRegAddr*  
access Slave register address  
*\*pRegData*  
read data

### Comments

The API can access i2c slave and read i2c slave device register.

### Return Codes

RT_ERR_OK	ok
RT_ERR_NULL_POINTER	input parameter is null pointer

## 24.2. rtk\_i2c\_data\_write

**rtk\_api\_ret\_t rtk\_i2c\_data\_write(rtk\_uint8 deviceAddr, rtk\_uint32 slaveRegAddr, rtk\_uint32 regData)**

write data to i2c slave device register

Defined in: i2c.h

### Parameters

*deviceAddr*  
access Slave device address  
*slaveRegAddr*  
access Slave register address

	<i>regData</i> data to set	
<b>Comments</b>	The API can access i2c slave and setting i2c slave device register.	
<b>Return Codes</b>	RT_ERR_OK	ok

### 24.3. rtk\_i2c\_init

**rtk\_api\_ret\_t rtk\_i2c\_init( void)**

I2C smart function initialization.

Defined in: i2c.h

<b>Parameters</b>	<i>void</i>
<b>Comments</b>	This API is used to initialize EEE status. need used GPIO clock
<b>Return Codes</b>	RT_ERR_OK ok RT_ERR_FAILED failed

#### 24.4. rtk\_i2c\_mode\_set

**rtk\_api\_ret\_t rtk\_i2c\_mode\_set(rtk\_I2C\_16bit\_mode\_t i2cmode)**

Set I2C data byte-order.

Defined in: i2c.h

<b>Parameters</b>	<i>i2cmode</i> byte
<b>Comments</b>	This API can set I2c traffic's byte-order .
<b>Return Codes</b>	RT_ERR_OK RT_ERR_FAILED RT_ERR_INPUT

---

## 24.5.rtk\_i2c\_mode\_get

`rtk_api_ret_t rtk_i2c_mode_get(rtk_I2C_16bit_mode_t *)`

Get i2c traffic byte-order setting.

Defined in: i2c.h

**Parameters**

\*  
    i2c byte

**Comments**

The API can get i2c traffic byte-order setting.

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	input parameter is null pointer

---

## 24.6.rtk\_i2c\_gpioPinGroup\_set

`rtk_api_ret_t rtk_i2c_gpioPinGroup_set(rtk_I2C_gpio_pin_t pins_group)`

Set i2c SDA & SCL used GPIO pins group.

Defined in: i2c.h

**Parameters**

*pins\_group*  
    GPIO pins group

**Comments**

The API can set i2c used gpio pins group. There are three group pins could be used

**Return Codes**

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	Invalid input parameter.

---

## 24.7.rtk\_i2c\_gpioPinGroup\_get

`rtk_api_ret_t rtk_i2c_gpioPinGroup_get(rtk_I2C_gpio_pin_t *)`

Get i2c SDA & SCL used GPIO pins group.

---

Defined in: i2c.h

**Parameters**

\*  
GPIO pins group

**Comments**

The API can get i2c used gpio pins group. There are three group pins could be used

**Return Codes**

RT\_ERR\_OK  
RT\_ERR\_NULL\_POINTER

ok  
input parameter is null pointer

---