RTL838x
LAYER 2 PLUS MANAGED 28*10/100/1000M -PORT SWITCH CONTROLLER

RTL833x
LAYER 2 PLUS MANAGED 28*10/100M -PORT SWITCH CONTROLLER

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

Developer Guide

Rev. 1.3
3 Dec 2013

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

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

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

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

USING THIS DOCUMENT
This document is intended for use by the system engineer when integrating with Realtek switch products. 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	Release Date	Summary
1.0	2012/10/30	Initial Release
1.1	2012/12/18	Modify vlan module FID_MSTI entry number and delete register RNG_CHK_VID_CTRL in chapter 2.8.1
1.2	2013/7/24	1. add mirror function initial api description in chapter 10.1 2. add vlan translation example in chapter 2.8.2 3. add description that only the first 6 blocks of acl support AND2 operation in chapter 4.4 4. CPU port must select not WFQ but WRR scheduling for PPS mode in chapter 15.3 5. add meter descriprion in acl module 6. Add VLAN profile set API setting example code 7. Modify TRK_SEP_TRAFFIC_CTRL.SEP_TRAFFIC description 8. add chapter14.3 Internal Prioirty for Packets Fowarded to CPU 9. add physical/logical port description in chapter 9 trunk
1.3	2013/12/3	1. modify 20.1 RTCT, 20.2 Green Ethernet cable length inaccuracy 2. add Green Ethernet API example

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

TABLE OF CONTENTS

Realtek CONFIDENTIAL CAMEO Communications, Inc

Realtek CONFIDENTIAL CAMEO Communications, Inc

Realtek CONFIDENTIAL AMEC Communications, Inc.

LIST OF TABLES

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

LIST OF FIGURES

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

1 Introduction

This document helps programmer to develop with RTL838x/RTL833x family devices. RTL838x is Gigabit Ethernet switch, whose member consists of **RTL8382M**, **RTL8380M**. RTL833x is Fast Ethernet switch, **RTL8332M** and **RTL8330M** are this family members.

The document will introduce following moules of the device family in later sections.

- ***Virtual Local Area Network***
- ***Layer 2 switch***
- ***Access Control List***
- ***Unicast Routing***
- ***Spanning Tree***
- ***Traffic Isolation***
- ***Reserved Multicast Address (RMA)***
- ***Trunk***
- ***Mirror***
- ***Ingress Bandwidth Control***
- ***Storm Control***
- ***Attack Prevention***
- ***Priority Decision***
- ***Egress Shaping***
- ***Egress Remarking***
- ***NIC (Network Interface Controller)***
- ***CPU Tag***
- ***802.1X***
- ***Cable Diagnostic***

Realtek CONFIDENTIAL CAMEO Communications, Inc.

2 Virtual Local Area Network

VLANs are used to segment the network into smaller broadcast domain or segments. The primary reason to segment the network is to relieve network congestion and increase bandwidth. The device supports up to 4096 (0-4095) VLANs which are used by inner and outer VLAN. Each VLAN can specify the member ports, untag ports for VLAN filtering and VLAN Tag manipulation. It supports flexible VLAN tag manipulation for different VLAN translation and Q-in-Q applications, such as, 1:1 VLAN translation and N:1 VLAN translation. In addition to support port-based VLAN and 1Q/1ad VLAN, the device also supports protocol-based VLAN by ACL and flow-based VLAN.

2.1 VLAN Functional Blocks

Packet goes over the VLAN functional blocks in below sequence.

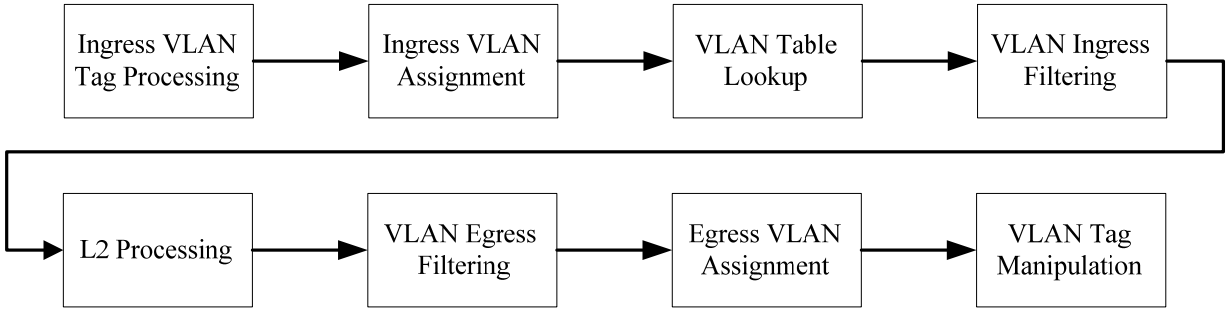


Figure 1: VLAN Functional Blocks Flow

The device first parses the VLAN tag of the packet. The packet is then examined by Accept Frame Type configurations (VLAN_PORT_ACCEPT_FRAME_TYPE register). Ingress VLAN assignment module determines the forwarding VLAN ID which is used for VLAN table lookup. After obtaining the VLAN member ports, the packet is examined by VLAN ingress and egress filtering modules. Before the packet is transmitted, egress VLAN assignment module determines the VLAN ID that is going to be encapsulated in VLAN tag. The egress VLAN ID can be the VLAN ID the device received or a new one assigned by other modules, such as ACL. VLAN tag manipulation module finally determines the VLAN tag status of the packet.

2.2 Ingress VLAN Tag Processing

The device supports up to three layer VLAN tags, namely outer (S-Tag), inner (C-Tag) and extra tag to cover different VLAN applications. In traditional VLAN application, only one VLAN tag is used and usually configures inner tag to be C-Tag (0x8100). In Q-in-Q application, two VLAN tags are used and usually configure inner tag to be C-Tag (0x8100) and outer tag to be S-Tag (0x88A8). Three VLAN tags are used for supporting the Q-in-Q application which may receive S+C tagged packets from downlink port and need to insert a third tag to the packet before transmitting it to the uplink port. In this case, the device should configure the inner tag as S-Tag and extra tag as C-Tag and the device can then insert the outer tag (third tag) to the packet according to the ingress port or the S-VID.

The TPID of three VLAN tags are configurable. The device supports four global outer TPIDs configured by VLAN_TAG_TPID_CTRL.OTPID fields, four global inner TPIDs configured by VLAN_TAG_TPID_CTRL.ITPID fields and one global extra TPID configured by VLAN_ETAG_TPID_CTRL.ETPID filed. Each ingress port can specify which TPIDs are used for parsing the packets through VLAN_PORT_OTAG_TPID_CMP_MSK, VLAN_PORT_ITAG_TPID_CMP_MSK, VLAN_PORT_ETAG_TPID_CMP registers. The VLAN tag processing flow is illustrated below:

Realtek CONFIDENTIAL CAMEO Communications, Inc.

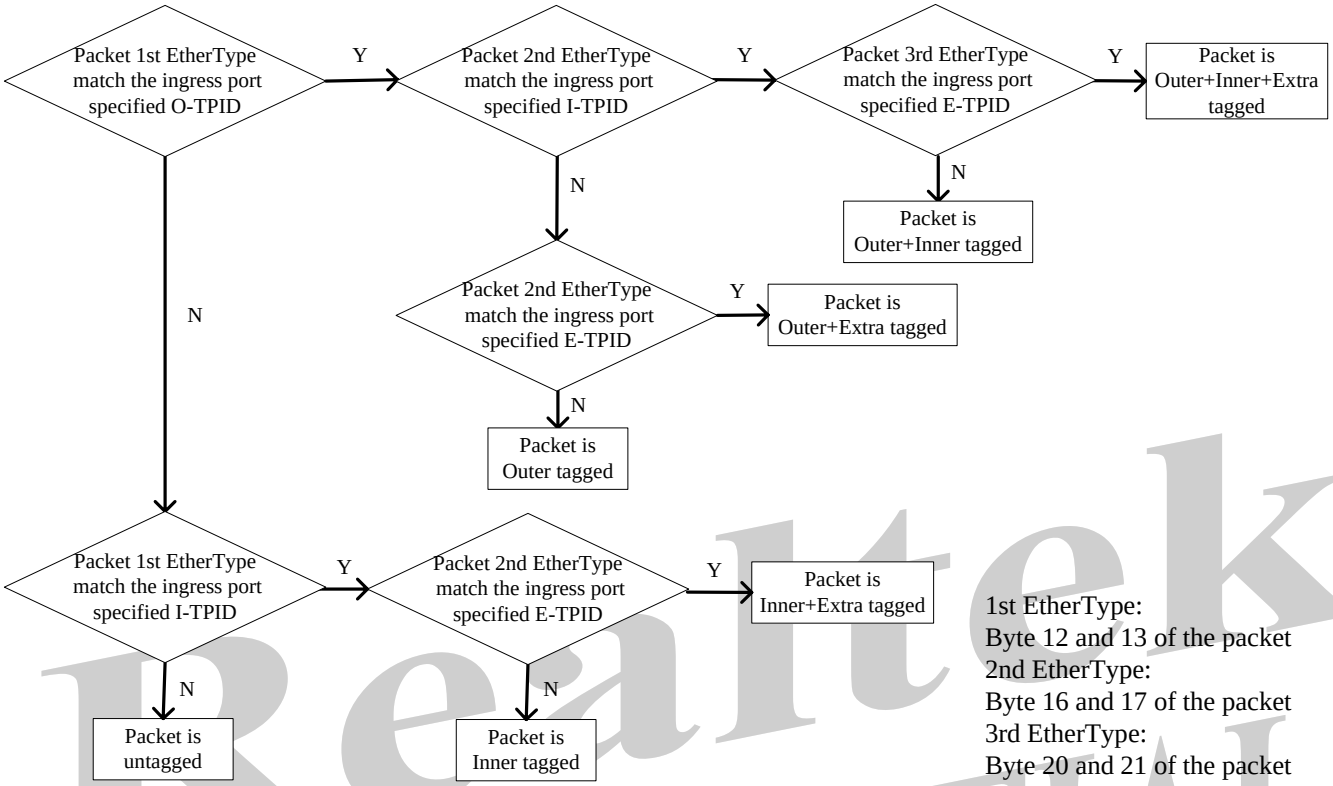


Figure 2: Ingress VLAN Tag Processing Flow

Extra tag is not used inside the device but recognized by the parser. Extra tag can only be parsed for the following tag combinations: Outer+Inner+Extra, Outer+Extra, Inner+Extra.

API REFERENCE

```
rtk_vlan_innerTpidEntry_get(uint32 unit, uint32 tpid_idx, uint32 *pTpid);
rtk_vlan_innerTpidEntry_set(uint32 unit, uint32 tpid_idx, uint32 tpid);
rtk_vlan_outerTpidEntry_get(uint32 unit, uint32 tpid_idx, uint32 *pTpid);
rtk_vlan_outerTpidEntry_set(uint32 unit, uint32 tpid_idx, uint32 tpid);
rtk_vlan_extraTpidEntry_get(uint32 unit, uint32 tpid_idx, uint32 *pTpid);
rtk_vlan_extraTpidEntry_set(uint32 unit, uint32 tpid_idx, uint32 tpid);

rtk_vlan_portIgrInnerTpid_get(uint32 unit, rtk_port_t port, uint32 *pTpid_idx_mask);
rtk_vlan_portIgrInnerTpid_set(uint32 unit, rtk_port_t port, uint32 tpid_idx_mask);
rtk_vlan_portIgrOuterTpid_get(uint32 unit, rtk_port_t port, uint32 *pTpid_idx_mask);
rtk_vlan_portIgrOuterTpid_set(uint32 unit, rtk_port_t port, uint32 tpid_idx_mask);
rtk_vlan_portIgrExtraTagEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_vlan_portIgrExtraTagEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
```

2.3 Ingress VLAN Assignment

The device assigns two ingress VLAN IDs (inner and outer VLAN ID) to each packet received, and per ingress port has a VLAN_PORT_FWD register to dictate which VLAN ID is used as forwarding VLAN ID to lookup VLAN table. In Q-in-Q application, outer VLAN ID is used as forwarding VLAN ID for both downlink and uplink ports as well as the inner VLAN ID is used in traditional VLAN application. Inner VLAN and outer VLAN decision flow will be described in following sections.

Realtek CONFIDENTIAL

2.3.1 Port-based VLAN

Each ingress port supports a VLAN_PORT_PB_VLAN register to specify the inner, outer port-based VLAN ID and the packet format to apply port-based VLAN configuration.

Table 1: VLAN_PORT_PB_VLAN Register

Field Name	Bits	Description
OPVID	12	Outer port-based VLAN ID.
OPVID_FMT	2	Apply outer port-based VLAN ID on: 2'b00: outer untagged and outer priority tagged packet 2'b01: outer untagged packet 2'b10: all packet(outer untagged, outer priority-tagged, outer tagged) 2'b11: reserved
IPVID	12	Inner port-based VLAN ID.
IPVID_FMT	2	Apply inner port-based VLAN ID on: 2'b00: inner untagged and inner priority tagged packet 2'b01: inner untagged packet 2'b10: all packet(inner untagged, inner priority-tagged, inner tagged) 2'b11: reserved

API REFERENCE

```
rtk_vlan_portPvidMode_get(uint32 unit, rtk_port_t port, rtk_vlan_pbVlan_mode_t *pMode);
rtk_vlan_portPvidMode_set(uint32 unit, rtk_port_t port, rtk_vlan_pbVlan_mode_t mode);
rtk_vlan_portOuterPvidMode_get(uint32 unit, rtk_port_t port, rtk_vlan_pbVlan_mode_t *pMode);
rtk_vlan_portOuterPvidMode_set(uint32 unit, rtk_port_t port, rtk_vlan_pbVlan_mode_t mode);
rtk_vlan_portPvid_get(uint32 unit, rtk_port_t port, uint32 *pPvid);
rtk_vlan_portPvid_set(uint32 unit, rtk_port_t port, uint32 pvid);
rtk_vlan_portOuterPvid_get(uint32 unit, rtk_port_t port, rtk_vlan_t *pPvid);
rtk_vlan_portOuterPvid_set(uint32 unit, rtk_port_t port, rtk_vlan_t pvid);
```

2.3.2 Protocol-and-Port-based VLAN

The feature bases on the Frame Type and EtherType of the packet to assign the VLAN specified by ingress ACL. Protocol-and-port-based VLAN of the device should only assigns inner VLAN and doesn't support to assign outer VLAN. The Protocol-and-port-based VLAN ID should apply to all the packet except the packet with inner tag (VID != 0).

Three frame types Ethernet, LLC_SNAP, LLC_Other should be supported and the flow to determine the frame type is illustrated as below:

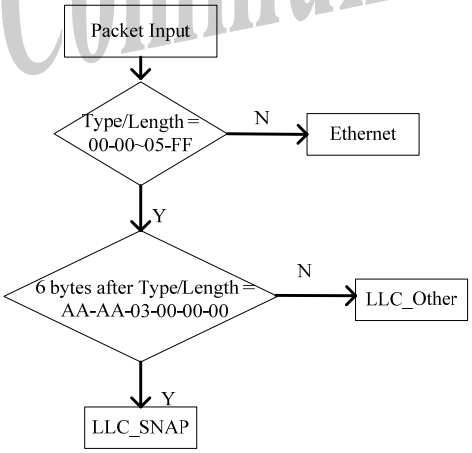


Figure 3: Frame Type Decision Flow

Realtek CONFIDENTIAL CAMEO Communications, Inc.

The EtherType field position of different packet encapsulation is illustrated as below:

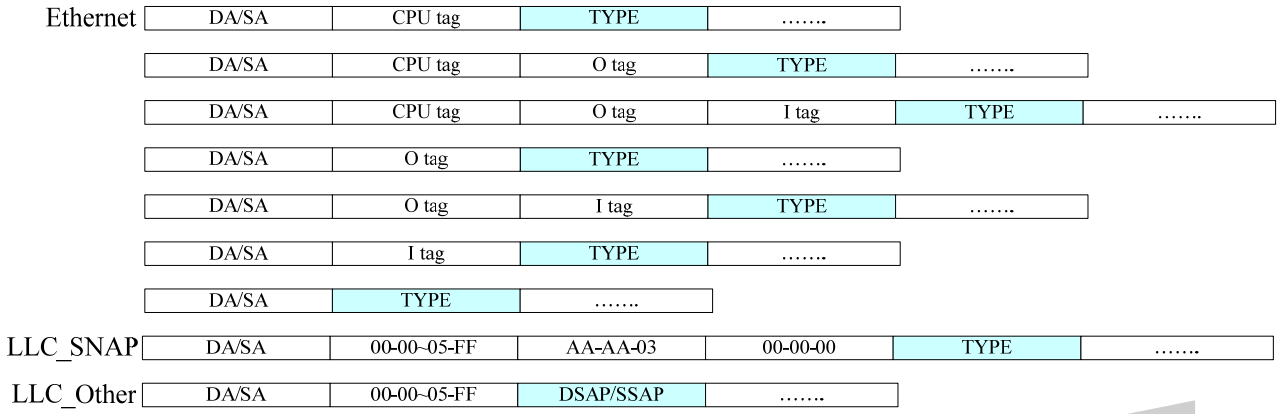


Figure 4: EtherType Field Position

2.3.3 Forwarding VLAN Decision

The device assigns two VLAN IDs (inner and outer VLAN ID) to each packet received, and one of them is chose as forwarding VLAN ID. Forwarding VLAN ID is used to lookup VLAN table and doing VLAN ingress/egress filtering. The device per ingress port supports a VLAN_PORT_FWD register to dictate which VLAN ID is used as learning and forwarding VLAN ID.

The flow to determine the ingress inner and outer VLAN ID is illustrated below.

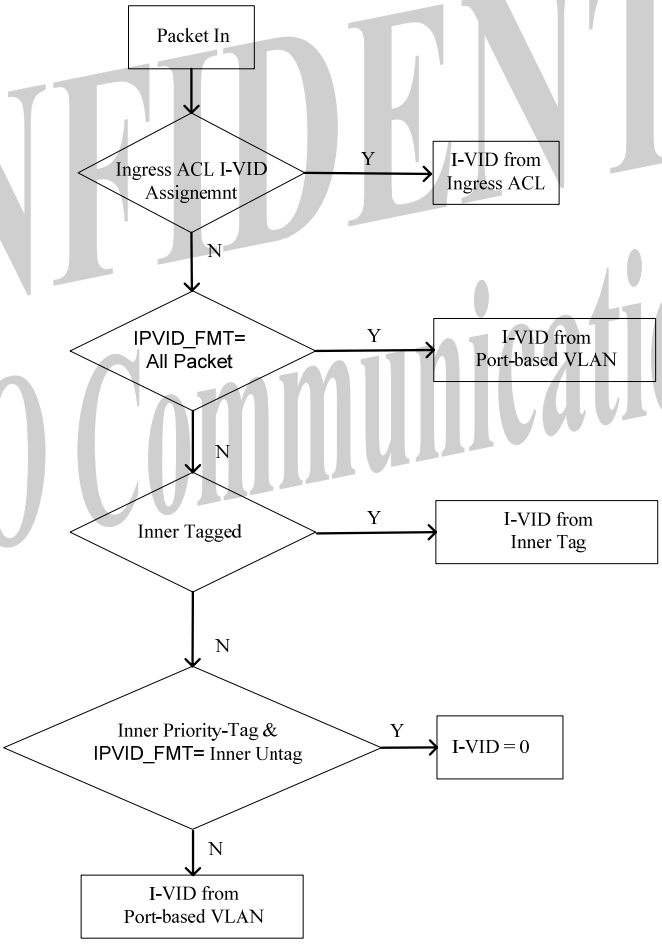


Figure 5: Ingress Inner VLAN ID Decision Flow

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

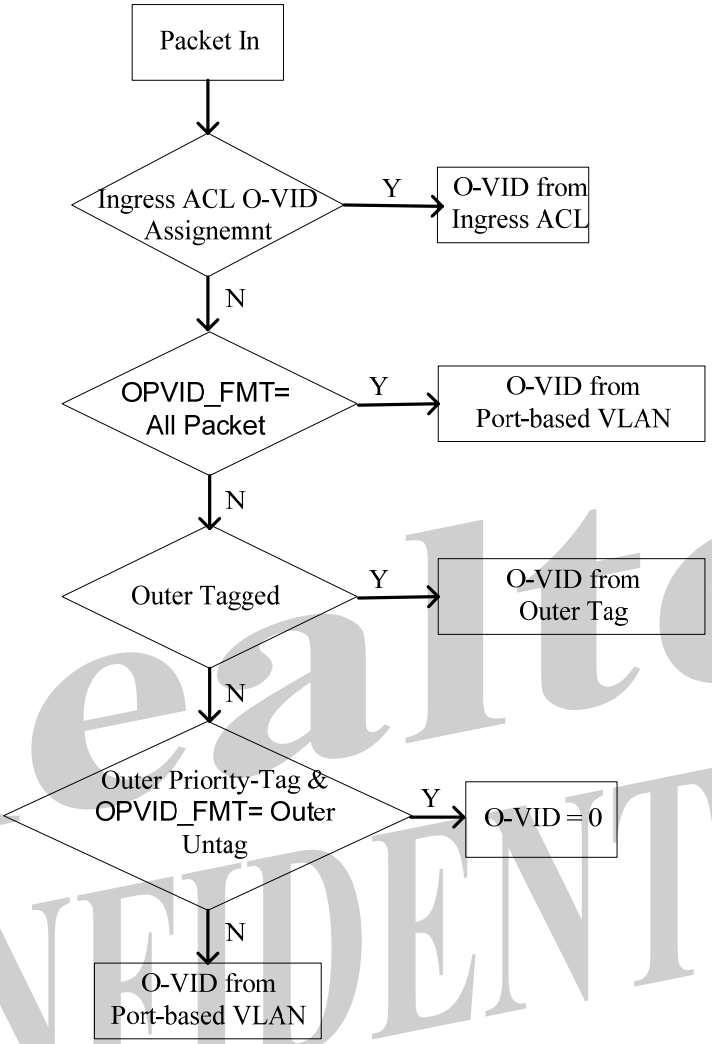


Figure 6: Ingress Outer VLAN ID Decision Flow

2.4 VLAN Lookup

When the forwarding VLAN is determined by VLAN decision flow, it is used as index to lookup the VLAN table for retrieving the member ports and untag member ports.

2.4.1 VLAN Table

The device supports a 4K (0-4095) entries VLAN table. The format of VLAN table entry is shown as below:

Table 2: VLAN Table

Field Name	Bits	Description
MBR	29	Member port.
UNTAG_MBR	29	Untag member port.
FID_MSTI	6	Filtering Database ID/Multiple Spanning Tree Instance ID.
L2_HASH_KEY_UC	1	Specify the VLAN learning mode for L2 unicast and broadcast traffic. 1'b0: IVL. VID is used for learning MAC address. 1'b1: SVL. FID_MSTI is used for learning MAC address.
L2_HASH_KEY_MC	1	Specify the VLAN learning mode for L2 and IP multicast traffic. 1'b0: IVL. VID is used for learning MAC address.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

		1'b1: SVL. FID_MSTI is used for learning MAC address.
VLAN_PROFILE	3	VLAN profile index.

MBR — Define the VLAN member port. VLAN member port is utilized by VLAN ingress and egress filtering.

UNTAG_MBR — Define the VLAN untag member port. The VLAN tag status of an outgoing packet is determined by both UNTAG_MBR setting and egress port tag status configuration. Please refer to Egress VLAN Tagging Status section for the detail.

FID_MSTI — The field represents both Filtering Database ID and Multiple Spanning Tree Instance ID. FID is used for SVL while MSTI is used for Spanning Tree ingress/egress filtering. There are 64 FID/MSTI supported by the device.

L2_HASH_KEY_UC — Per VLAN specify the VLAN learning mode for L2 unicast and broadcast traffic. IVL/SVL mixed mode can be supported through configuring the field.

L2_HASH_KEY_MC — Per VLAN specify the VLAN learning mode for L2 and IP multicast traffic. IVL/SVL mixed mode can be supported through configuring the field.

VLAN_PROFILE — Please refer to VLAN Profile section for the detail.

API REFERENCE

```
rtk_vlan_create(uint32 unit, rtk_vlan_t vid);
rtk_vlan_destroy(uint32 unit, rtk_vlan_t vid);
rtk_vlan_destroyAll(uint32 unit, uint32 restore_default_vlan);
rtk_vlan_port_add(uint32 unit, rtk_vlan_t vid, rtk_port_t port, uint32 is_untag);
rtk_vlan_port_del(uint32 unit, rtk_vlan_t vid, rtk_port_t port);
rtk_vlan_port_get(
    uint32          unit,
    rtk_vlan_t      vid,
    rtk_portmask_t *pMember_portmask,
    rtk_portmask_t *pUntag_portmask);
rtk_vlan_port_set(
    uint32          unit,
    rtk_vlan_t      vid,
    rtk_portmask_t *pMember_portmask,
    rtk_portmask_t *pUntag_portmask);
rtk_vlan_stg_get(uint32 unit, rtk_vlan_t vid, rtk_stg_t *pStg);
rtk_vlan_stg_set(uint32 unit, rtk_vlan_t vid, rtk_stg_t stg);
rtk_vlan_l2UcastLookupMode_get(uint32 unit, rtk_vlan_t vid, rtk_l2_ucastLookupMode_t *pMode);
rtk_vlan_l2UcastLookupMode_set(uint32 unit, rtk_vlan_t vid, rtk_l2_ucastLookupMode_t mode);
rtk_vlan_l2McastLookupMode_get(uint32 unit, rtk_vlan_t vid, rtk_l2_mcastLookupMode_t *pMode);
rtk_vlan_l2McastLookupMode_set(uint32 unit, rtk_vlan_t vid, rtk_l2_mcastLookupMode_t mode);
rtk_vlan_profileIdx_get(uint32 unit, rtk_vlan_t vid, uint32 *pIdx);
rtk_vlan_profileIdx_set(uint32 unit, rtk_vlan_t vid, uint32 idx);
```

2.4.2 VLAN Profile

The device supports 8 VLAN profiles. VLAN profile is used to specify the SA learning behavior and unknown multicast flooding port mask. VLAN profile structure is shown as below:

Table 3: VLAN Profile

Field Name	Bits	Description
L2_LRN_EN	1	Enable MAC address learning.
L2_UNKN_MC_FLD_PMSK	9	An index points to multicast table for retrieving unknown L2 multicast flooding port mask.
IP4_UNKN_MC_FLD_PMSK	9	An index points to multicast table for retrieving unknown IPv4 multicast flooding port mask.
IP6_UNKN_MC_FLD_PMS	9	An index points to multicast table for retrieving unknown IPv6 multicast

K	flooding port mask.

Because each VLAN maps to a VLAN profile, disable MAC address learning or have different flooding port mask on specific VLAN is practicable.

API REFERENCE

```
rtk_vlan_profile_get(uint32 unit, uint32 idx, rtk_vlan_profile_t *pProfile);
rtk_vlan_profile_set(uint32 unit, uint32 idx, rtk_vlan_profile_t *pProfile);
```

Example 1: VLAN Profile Setting

Condition:
Set VLAN profile index for L2/IP4/IP6 unknown multicast entry indexs.

```
rtk_vlan_profile_t    profile;
int32 index = -1;
uint32 unit = 0;
/* set VLAN profile Index */
osal_memset(profile, 0, sizeof(rtk_vlan_profile_t));
profile.learn = 1;

/* free the occupy index 0 before alloc new */
/* input parameter index = -1, can make sure alloc a new multicast table entry */
rtk_l2_mcastFwdIndex_free(uint, 0);
index = -1;
rtk_l2_mcastFwdIndex_alloc(uint, &index);
profile. l2_mcast_dlf_pm_idx = index;

rtk_l2_mcastFwdIndex_free(uint, 0);
lndex = -1;
rtk_l2_mcastFwdIndex_alloc(uint, &index);
profile. ip4_mcast_dlf_pm_idx = index;

rtk_l2_mcastFwdIndex_free(uint, 0);
lndex = -1;
rtk_l2_mcastFwdIndex_alloc(uint, &index);
profile. ip6_mcast_dlf_pm_idx = index;

/* set VLAN profile */
lndex = 4; /* choose which profile you want to configure */
rtk_vlan_profile_set(uint, index, &profile);
```

2.5 VLAN Filtering

A packet is examined by VLAN ingress filtering when it is received and examined by VLAN egress filtering before it is transmitted.

2.5.1 VLAN Ingress Filtering

A packet is asserted by VLAN ingress filtering if its source port is not in the member ports. The device per ingress port supports a VLAN_PORT_IGR_FLTR register to handle the packet.

Table 4: VLAN_PORT_IGR_FLTR Register

Field Name	Bits	Description

IGR_FLTR_ACT	2	VLAN ingress filtering action. 2'b00: forward 2'b01: drop 2'b10: trap 2'b11: reserved

API REFERENCE

*rtk_vlan_portIgrFilter_get(uint32 unit, rtk_port_t port, rtk_vlan_ifilter_t *pIgr_filter);*
rtk_vlan_portIgrFilter_set(uint32 unit, rtk_port_t port, rtk_vlan_ifilter_t igr_filter);

2.5.2 VLAN Egress Filtering

A packet should only be forwarded to the VLAN where it came from. This is achieved by VLAN egress filtering. However, the device per egress port supports a VLAN_PORT_EGR_FLTR register to disable the VLAN egress filtering. When the VLAN egress filtering is disabled at a certain port, all the traffic (unicast/multicast/broadcast) forwarded to that port will bypass the VLAN egress filtering.
When VLAN egress filtering is enabled, the device globally supports a VLAN_CTRL.LKY field to disable the VLAN egress filtering for known L2/IPv4/IPv6 multicast traffic to across VLAN. The VLAN Leaky configuration (VLAN_CTRL.LKY) doesn't apply to L2 unicast and broadcast traffic.

API REFERENCE

*rtk_vlan_portEgrFilterEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);*
rtk_vlan_portEgrFilterEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
*rtk_vlan_mcastLeakyEnable_get(uint32 unit, rtk_enable_t *pLeaky);*
rtk_vlan_mcastLeakyEnable_set(uint32 unit, rtk_enable_t leaky);

2.6 Egress VLAN Assignment

In addition to assign two ingress VLAN IDs (inner and outer VLAN ID) to each packet received, the device also determines two egress VLAN IDs for the packet. Unlike ingress VLAN ID, the egress VLAN ID doesn't participate VLAN forwarding and VLAN ingress/egress filtering processing but encapsulates to the packet before transmitting.
Each packet is determined two egress VLAN IDs before the egress tag status decision is made. They are used for encapsulation only if the transmitting packet carries VLAN tags. Regarding the egress tag status decision flow, please refer to Egress VLAN Tagging Status section.

2.6.1 Egress VLAN Translation Table

The device provides an egress VLAN translation table which contains 256K entries to support downstream VLAN translation and Q-in-Q applications. Egress VLAN translation table is used for VLAN translation downstream traffic and the VLAN assigned by egress VLAN translation table is encapsulated to the downstream packet.
It can be used to support the 1:1 VLAN translation and Q-in-Q **downstream** models listed below but not limited to:

1:1 VLAN Translation/Q-in-Q Downstream Models
C' → C
S → C

A configuration register VLAN_EGR_CNVT_CTRL is provided to specify the VLAN source and specify whether translate the double tag packet.

Table 5: VLAN_EGR_CNVT_TBL_CTRL Register

Field Name	Bits	Description
DBL_TAG_EN	1	Apply egress VLAN translation to double (outer + inner) tag packet.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	1'b0: disable 1'b1: enable

Table 6: VLAN_PORT_CNVT_CTRL Register

Field Name	Bits	Description
ORGVID	1	Egress port VID conversion original vid selection. 1'b0: inner VID 1'b1: outer VID Note: If the original tag does not exist, VID conversion will not be done.
TGTVID	1	Egress port VID conversion target vid selection. 1'b0: inner should be translated 1'b1: outer should be translated Note: If packet hit egress conversion table and its target tag does not exist, it will be inserted a target tag.
LKMISS_ACT	1	Egress port VID conversion table lookup miss action. 1'b0: normal forwarding 2'b0: the packet should not be transmitted from this egress port Note: If the original tag does not exist, it is treated as lookup miss, if ingress ACL or address table has determined the vlan conversion result, the egress VID conversion module will be bypassed.

Egress VLAN translation table is shown as below:

Table 7: Egress VLAN Translation Table

Field Name	Bits	Description
Comparing Key		
VALID	1	Valid bit.
ORGVID	12	VLAN ID either from outer tag or inner tag, refer to VLAN_EGR_CNVT_CTRL.VID_SRC. The field should not be used with field VID_RANGE_CHK.
ORGPRI		Inner or Outer User Priority, refer to VLAN_EGR_CNVT_CTRL.VID_SRC.
ORGVID_RNG_CHK	32	VLAN ID Range Check Result (Refer to VLAN Range Check section for the detail). Each bit corresponds to a range check comparison result. The field should not be used with field VID.
PORT_ID	6	Egress Port.
ORGVID_MASK	12	VLAN ID mask.
ORGVID_RNG_CHK_MASK	32	VLAN ID Range Check Result mask.
ORGPRI_MASK	3	User Priority mask.
PORT_ID_MASK	6	Egress Port mask.
Translation Data		
VID_SHIFT	1	1'b0: translate the packet's VID to NEWVID 1'b1: shift the packet's VID with NEWVID
NEWVID	12	VID_SHIFT = 0, NEWVID will be the translated inner VID. VID_SHIFT = 1, NEWVID is the offset value for translation. That is, translated inner VID = Packet's VID + NEWVID. Note: Use wrap around for a negative shift. If VID_SHIFT=1 & NEWVID=0xFFF, the target tag should be untagged. Example: If VLAN 100(X) shifts to VLAN 10(Y) is desirous, the NEWVID value should be set to 4006(Z). Formula is Z = 4096 – X + Y.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

PRI_ASSIGN	1	1'b0: do not assign the priority. 1'b1: assign the priority.
NEWPRI	3	NEWPRI is encapsulated to VLAN tag and supersede the egress tag remarking. It doesn't participate ingress priority decision. The field is used if PRI_ASSIGN=1.

The comparing keys are implemented by TCAM, so each field has a corresponding mask which can mask off the comparing key. The comparing keys are used to qualify the packet while the translation data are used for doing egress VLAN translation. In addition to translate to a new inner VID, inner priority and new outer VID, outer priority can also be assigned. As for egress tag status, packet hit egress VLAN translation table can be forced to be inner or outer tag.
The egress VLAN translation table takes effect for tagged (exclude priority-tag) packet only and if there are multiple entries hit, the entry with lowest index is used.

API REFERENCE

```
rtk_vlan_egrVlanCnvtDblTagEnable_get(uint32 unit, rtk_enable_t *pEnable);
rtk_vlan_egrVlanCnvtDblTagEnable_set(uint32 unit, rtk_enable_t enable);
rtk_vlan_egrVlanCnvtEntry_get(uint32 unit, uint32 index, rtk_vlan_egrVlanCnvtEntry_t *pData);
rtk_vlan_egrVlanCnvtEntry_set(uint32 unit, uint32 index, rtk_vlan_egrVlanCnvtEntry_t *pData);
rtk_vlan_egrVlanCnvtEntry_delAll(uint32 unit);
rtk_vlan_portEgrVlanCnvtVidSource_get(uint32 unit, rtk_port_t port, rtk_l2_vlanMode_t *pSrc);
rtk_vlan_portEgrVlanCnvtVidSource_set(uint32 unit, rtk_port_t port, rtk_l2_vlanMode_t src);
rtk_vlan_portEgrVlanCnvtVidTarget_get(uint32 unit, rtk_port_t port, rtk_l2_vlanMode_t *pTgt);
rtk_vlan_portEgrVlanCnvtVidTarget_set(uint32 unit, rtk_port_t port, rtk_l2_vlanMode_t tgt);
rtk_vlan_portEgrVlanCnvtLookupMissAct_get(uint32 unit, rtk_port_t port, rtk_vlan_lookupMissAct_t *pAct);
rtk_vlan_portEgrVlanCnvtLookupMissAct_set(uint32 unit, rtk_port_t port, rtk_vlan_lookupMissAct_t act);
```

2.6.2 Egress VLAN Decision

The flow to determine the egress inner and outer VLAN ID is illustrated below.

CONFIDENTIAL
CAMEO Communications ,Inc.

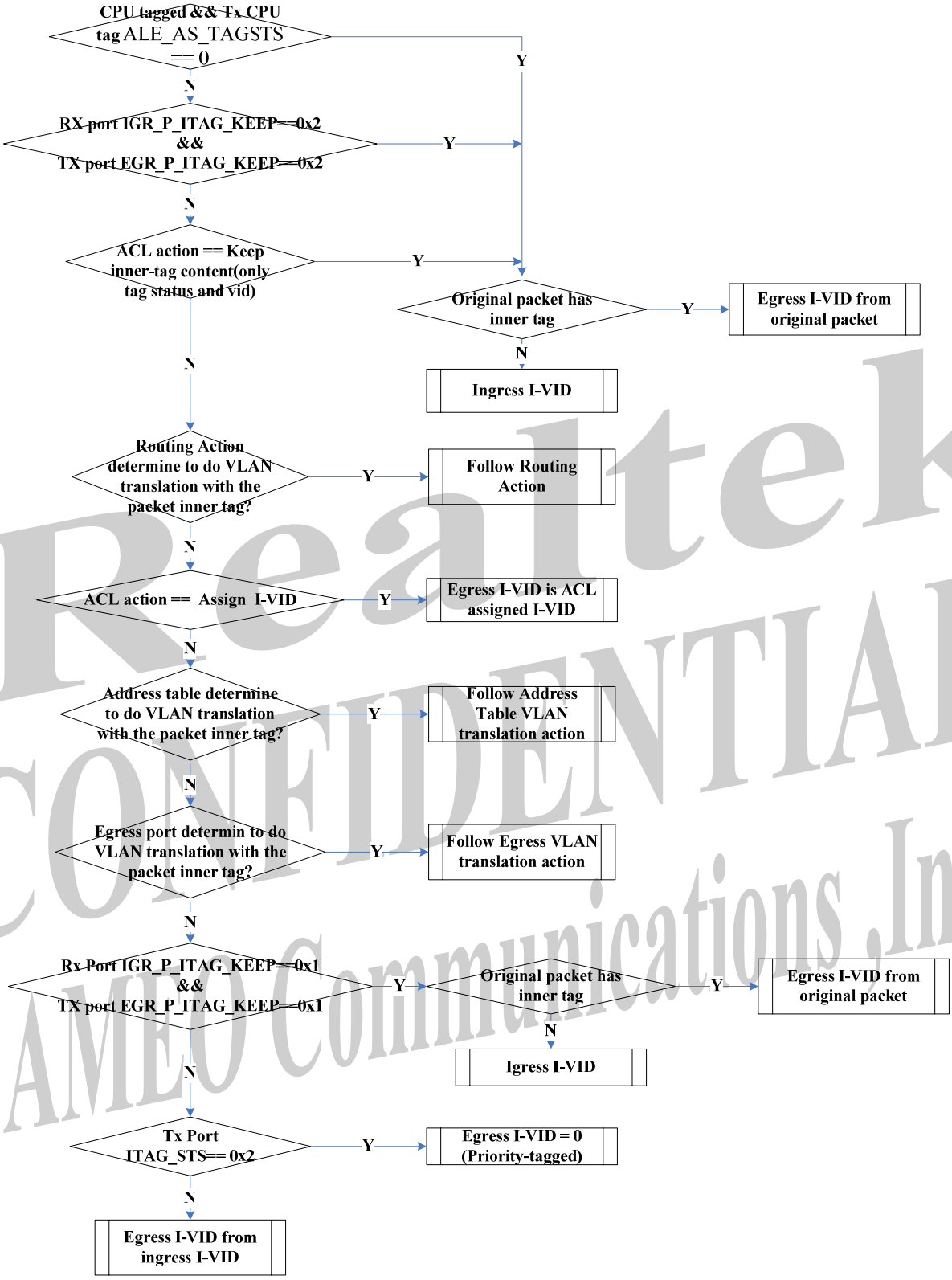


Figure 7: Egress Inner VLAN ID Decision Flow

Realtek CONFIDENTIAL CAMEO Communications, Inc.

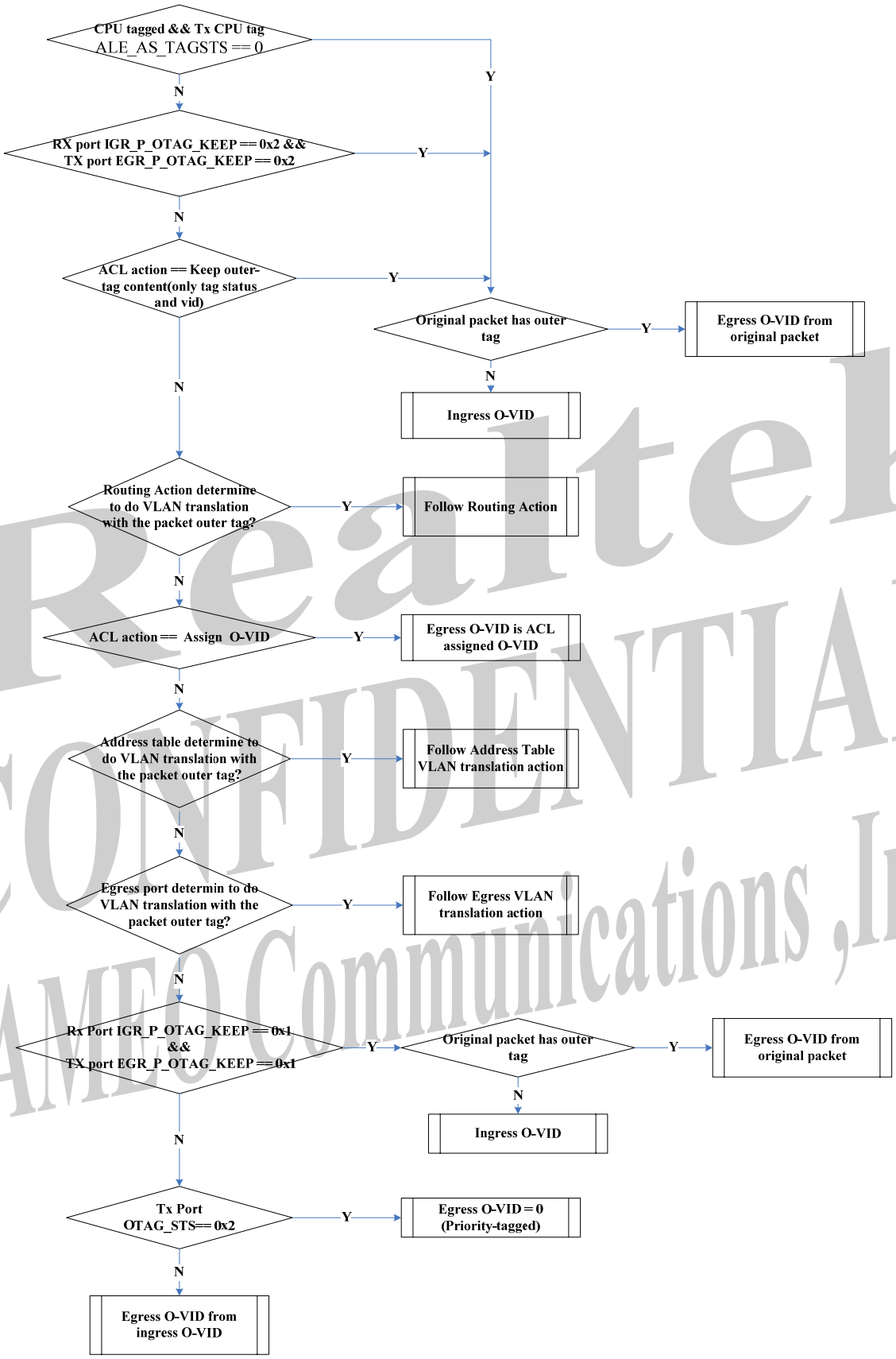


Figure 8: Egress Outer VLAN ID Decision Flow

The section only describes the egress VLAN ID decision flow. Regarding the priority and CFI field decision flow,

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

please refer to Egress Remarking Developer's Guide.

2.7 Egress VLAN Tag Manipulation

The device determines egress inner and outer tag status for a packet before transmitting. Double VLAN tag insertion, remove, keep operations are supported for flexible Q-in-Q VLAN application. In addition to Port-based and VLAN-based tag status assignment, flow-based is also supported by Ingress ACL.

2.7.1 Egress VLAN Tagging Status

The device per port supports a VLAN_PORT_TAG_STS_CTRL register to specify the VLAN tag processing behavior.

Table 8: VLAN_PORT_TAG_STS_CTRL Register

Field Name	Bits	Description
IGR_P_ITAG_KEEP	2	Per ingress port specify to keep inner tag content(VID/CFI/Priority are all kept). 2'b00: normal mode(follow VLAN untag set and ITAG_STS configuration) 2'b01: keep but could be overwritten by ASIC internal decision 2'b10: always keep, could not overwritten by ASIC internal decision 2'b11: reserved
IGR_P_OTAG_KEEP	2	Per ingress port specify to keep outer tag content(VID/CFI/Priority are all kept). 2'b00: normal mode(follow VLAN untag set and ITAG_STS configuration) 2'b01: keep but could be overwritten by ASIC internal decision 2'b10: always keep, could not overwritten by ASIC internal decision 2'b11: reserved
EGR_P_ITAG_KEEP	2	Per egress port specify to keep inner tag content(VID/CFI/Priority are all kept). 2'b00: normal mode(follow VLAN untag set and ITAG_STS configuration) 2'b01: keep but could be overwritten by ASIC internal decision 2'b10: always keep, could not overwritten by ASIC internal decision 2'b11: reserved
EGR_P_OTAG_KEEP	2	Per egress port specify to keep outer tag content(VID/CFI/Priority are all kept). 2'b00: normal mode(follow VLAN untag set and ITAG_STS configuration) 2'b01: keep but could be overwritten by ASIC internal decision 2'b10: always keep, could not overwritten by ASIC internal decision 2'b11: reserved
ITAG_STS	2	Inner VLAN tag status. 2'b00: inner untag 2'b01: inner tag 2'b10: inner priority-tag 2'b11: reserved
OTAG_STS	2	Outer VLAN tag status. 2'b00: inner untag 2'b01: inner tag 2'b10: inner priority-tag 2'b11: reserved

The inner VLAN tag content and format of a packet, including VID/Priority/CFI, are kept if the configuration IGR_P_ITAG_KEEP of packet's ingress port and the configuration EGR_P_ITAG_KEEP of packet's egress port are BOTH set to 0x2. If the configuration IGR_P_ITAG_KEEP of packet's ingress port and the configuration EGR_P_ITAG_KEEP of packet's egress port are BOTH set to 0x1, the tag status is kept, but it can be modified by ASIC internal decision.

The outer VLAN tag content and format of a packet, including VID/Priority/DEI, are kept if the configuration IGR_P_OTAG_KEEP of packet's ingress port and the configuration EGR_P_OTAG_KEEP of packet's egress port are BOTH set to 0x2. If the configuration IGR_P_OTAG_KEEP of packet's ingress port and the configuration EGR_P_OTAG_KEEP of packet's egress port are BOTH set to 0x1, the tag status is kept, but it can be modified by ASIC internal decision.

REALTEK CONFIDENTIAL CAMEO Communications, Inc.

If IGR_P_ITAG_KEEP(IGR_P_OTAG_KEEP) of packet's ingress port and EGR_P_ITAG_KEEP (EGR_P_OTAG_KEEP) of packet's egress port is the other setting, the VLAN tag status is then determined by ASIC internal decision, VLAN untag set and ITAG_STS (OTAG_STS) configuration of egress port. According to forwarding VLAN configuration (VLAN_PORT_FWD), the VLAN tag processing is different.
If forwarding VLAN is from inner VLAN, the packet is determined to be inner untag if its egress port is in the VLAN untag member, otherwise, the inner tag status is directly determined by ITAG_STS field. However, outer tag status is simply determined by OTAG_STS field.
If forwarding VLAN is from outer VLAN, the packet is determined to be outer untag if its egress port is in the VLAN untag member, otherwise, the outer tag status is directly determined by OTAG_STS field. However, inner tag status is simply determined by ITAG_STS field.

The flow to determine the egress inner and outer VLAN tag status is illustrated below.

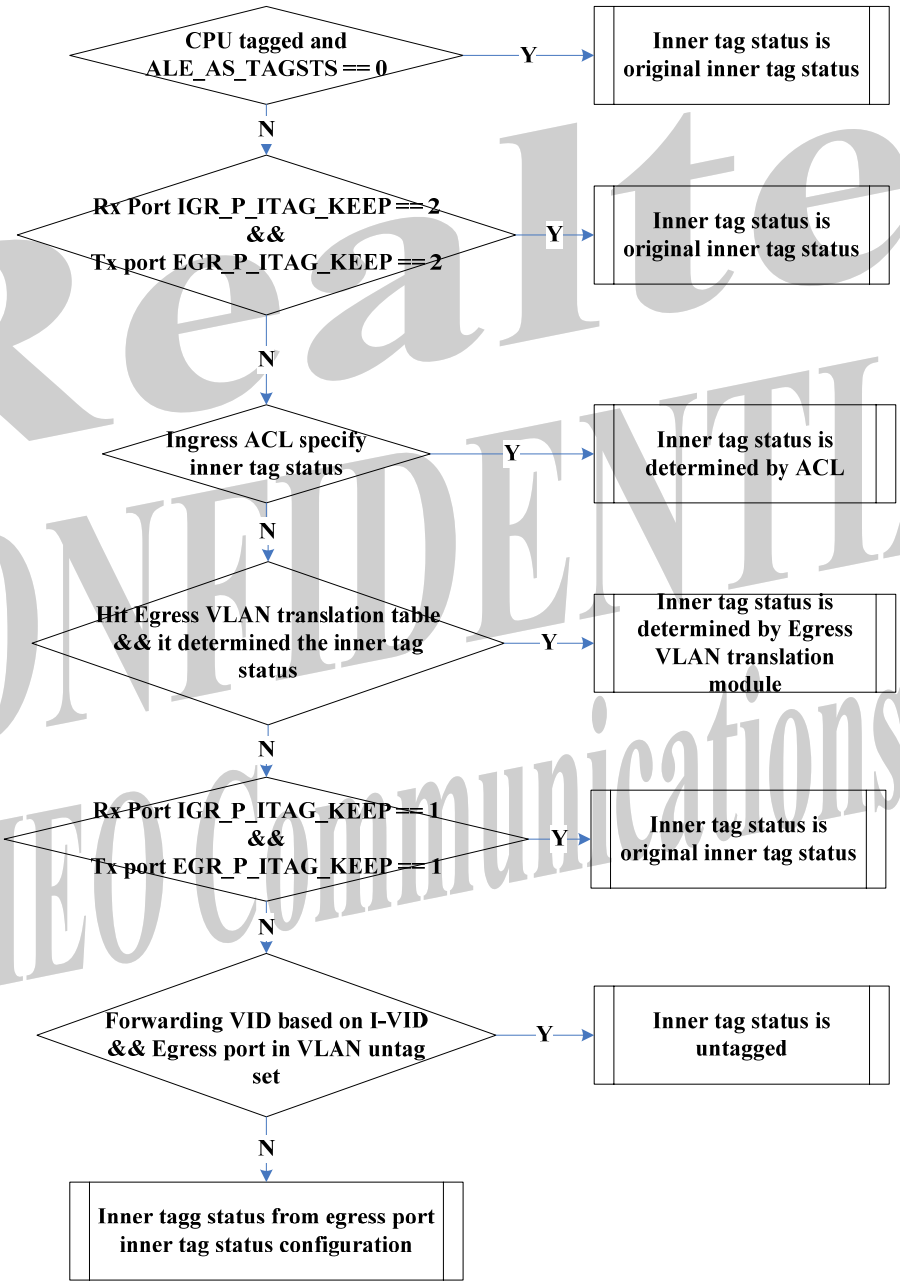


Figure 9: Egress Inner VLAN Tag Status Decision Flow

Realtek CONFIDENTIAL CAMEO Communications, Inc.

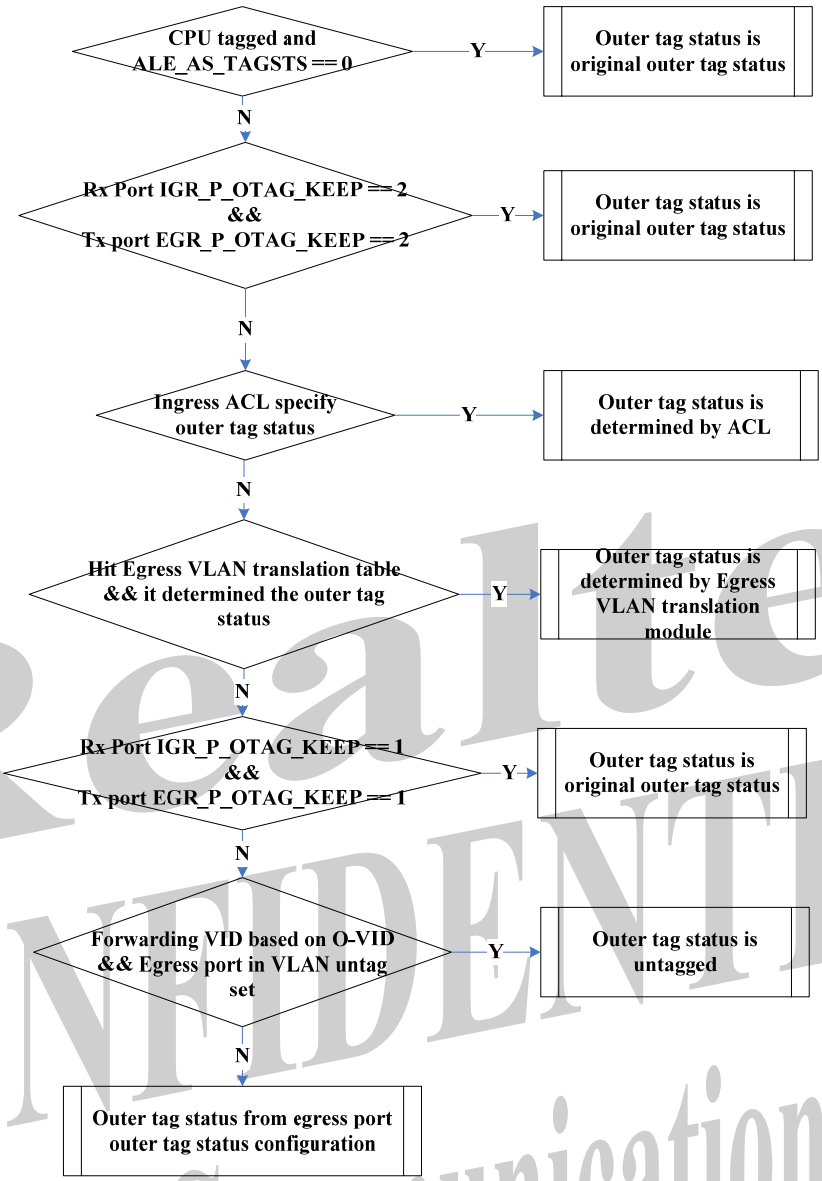


Figure 10: Egress Outer VLAN Tag Status Decision Flow

API REFERENCE

```
rtk_vlan_portIgrTagKeepEnable_get(
    uint32          unit,
    rtk_port_t      port,
    rtk_enable_t    *pKeepOuter,
    rtk_enable_t    *pKeepInner);
rtk_vlan_portIgrTagKeepEnable_set(
    uint32          unit,
    rtk_port_t      port,
    rtk_enable_t    keepOuter,
    rtk_enable_t    keepInner);
rtk_vlan_portEgrTagKeepEnable_get(
    uint32          unit,
    rtk_port_t      port,
    rtk_enable_t    *pKeepOuter,
    rtk_enable_t    *pKeepInner);
```

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

```
rtk_vlan_portEgrTagKeepEnable_set(
    uint32              unit,
    rtk_port_t          port,
    rtk_enable_t        keepOuter,
    rtk_enable_t        keepInner);
rtk_vlan_portIgrTagKeepType_get(
    uint32              unit,
    rtk_port_t          port,
    rtk_vlan_tagKeepType_t   *pKeeptypeOuter,
    rtk_vlan_tagKeepType_t   *pKeeptypeInner);
rtk_vlan_portIgrTagKeepType_set(
    uint32              unit,
    rtk_port_t          port,
    rtk_vlan_tagKeepType_t   keeptypeOuter,
    rtk_vlan_tagKeepType_t   keeptypeInner);
rtk_vlan_portEgrTagKeepType_get(
    uint32              unit,
    rtk_port_t          port,
    rtk_vlan_tagKeepType_t   *pKeeptypeOuter,
    rtk_vlan_tagKeepType_t   *pKeeptypeInner);
rtk_vlan_portEgrTagKeepType_set(
    uint32              unit,
    rtk_port_t          port,
    rtk_vlan_tagKeepType_t   keeptypeOuter,
    rtk_vlan_tagKeepType_t   keeptypeInner);
rtk_vlan_portEgrInnerTagSts_get(uint32 unit, rtk_port_t port, rtk_vlan_tagSts_t *pSts);
rtk_vlan_portEgrInnerTagSts_set(uint32 unit, rtk_port_t port, rtk_vlan_tagSts_t sts);
rtk_vlan_portEgrOuterTagSts_get(uint32 unit, rtk_port_t port, rtk_vlan_tagSts_t *pSts);
rtk_vlan_portEgrOuterTagSts_set(uint32 unit, rtk_port_t port, rtk_vlan_tagSts_t sts);
```

2.7.2 Egress VLAN Tagging TPID

Port-based, VLAN-based and flow-based VLAN TPID is supported. The device per egress port supports a VLAN_PORT_EGR_ITPID_CTRL.ITPID_IDX and VLAN_PORT_EGR_OTPID_CTRL.OTPID_IDX register field to specify the inner and outer TPID respectively. Please note instead specify the TPID value directly; the field specify the index to TPID list. The configuration doesn't take effect if received packet originally carries inner (outer) tag, that is, inner (outer) TPID is kept if received packet originally carries inner (outer) tag.
VLAN-based TPID is supported by ingress and egress VLAN translation table while flow-based TPID is supported by egress ACL. The flow to determine the egress inner and outer VLAN TPID is illustrated below.

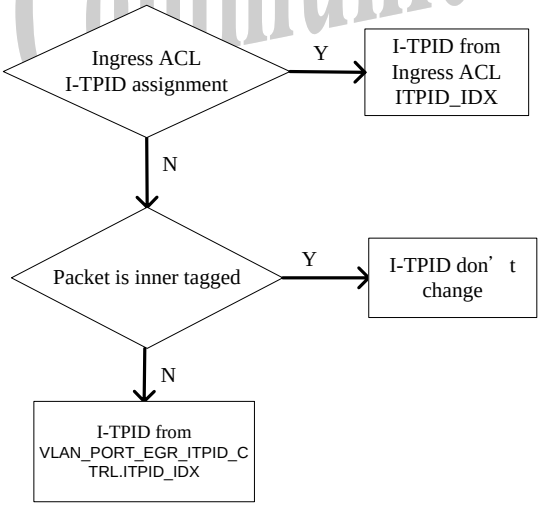


Figure 11: Egress Inner VLAN TPID Decision Flow

CONFIDENTIAL
CAMEO Communications, Inc.

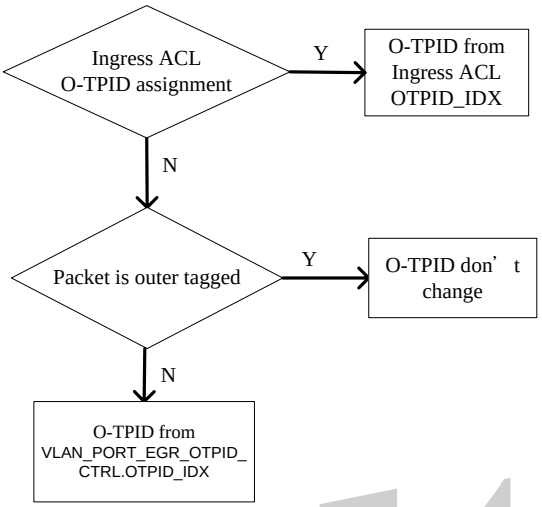


Figure 12: Egress Outer VLAN TPID Decision Flow

API REFERENCE

```
rtk_vlan_portEgrInnerTpid_get(uint32 unit, rtk_port_t port, uint32 *pTpid_idx);
rtk_vlan_portEgrInnerTpid_set(uint32 unit, rtk_port_t port, uint32 tpid_idx);
rtk_vlan_portEgrOuterTpid_get(uint32 unit, rtk_port_t port, uint32 *pTpid_idx);
rtk_vlan_portEgrOuterTpid_set(uint32 unit, rtk_port_t port, uint32 tpid_idx);
```

2.8 VLAN Translation

The device provides 256 egress VLAN translation entries to support VLAN translation and flexible Q-in-Q applications.Egress VLAN translation table is for VLAN translation/Q-in-Q downstream traffic. The VLAN assigned by egress VLAN translation table just encapsulated to the downstream packet and do not participate the VLAN forwarding process.

2.8.1 VLAN Range Check

There are also 16 dedicated VLAN range check configurations RNG_CHK_VID_EGR_XLATE_CTRL supported for egress VLAN translation table. The received packet is compared by 16 VLAN range check configurations concurrently and the 16-bit comparison result is then examined by the field VID_RANGE_CHK of egress VLAN translation table.

Table 9: RNG_CHK_VID_EGR_XLATE_CTRL Register

Field Name	Bits	Description
VID_UPPER	12	VLAN range upper bound.
VID_LOWER	12	VLAN range lower bound.
REVERSE	1	Reverse comparison result.
TYPE	1	Specify the VLAN source. 1'b0: inner VLAN 1'b1: outer VLAN

The comparison result is valid if the specified VLAN source is available of the received packet and reverse operation only takes effect for valid comparison result.

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_acl_rangeCheckVid_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_vid_t *pData);
rtk_acl_rangeCheckVid_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_vid_t *pData);
rtk_vlan_egrVlanCnvtRangeCheckVid_get(
    uint32                                  unit,
    uint32                                  index,
    rtk_vlan_egrVlanCnvtRangeCheck_vid_t    *pData);
rtk_vlan_egrVlanCnvtRangeCheckVid_set(
    uint32                                  unit,
    uint32                                  index,
    rtk_vlan_egrVlanCnvtRangeCheck_vid_t    *pData);
```

2.8.2 VLAN Translation Example

Example 1: VLAN 1:1 Mapping

Condition:
Packet from downlink port #1 with inner VLAN 10, and translates to inner VLAN 100 before sending out uplink port #2.

```
rtk_vlan_egrVlanCnvtEntry_t    data ;
/* set egress VLAN translation */
/* set customer uplink translation information (10->100) */
osal_memset(&data, 0, sizeof(rtk_vlan_egrVlanCnvtEntry_t));
/* search key setting */
data.vid_ignore = 0;
data.vid = 10;
data.orgpri_ignore = True;
data.port_ignore = True;
data.rngchk_result_mask = 0;
/* assign new value */
data.vid_shift = 0;
data.vid_new = 100;
data.pri_assign = 0;
data.valid = True;
rtk_vlan_egrVlanCnvtEntry_set(unit, 0, &data);
/* set customer downlink translation information (100->10) */
osal_memset(&data, 0, sizeof(rtk_vlan_egrVlanCnvtEntry_t));
/* search key setting */
data.vid_ignore = 0;
data.vid = 100;
data.orgpri_ignore = True;
data.port_ignore = True;
data.rngchk_result_mask = 0;
/* assign new value */
data.vid_shift = 0;
data.vid_new = 10;
data.pri_assign = 0;
data.valid = True;
rtk_vlan_egrVlanCnvtEntry_set(unit, 1, &data)
```

Example 2: VLAN 1:1 Shift Mapping

Condition:
Packet from downlink port with inner VLAN 10, and translates to inner VLAN 210 before sending out uplink port.
Packet from downlink port with inner VLAN 11, and translates to inner VLAN 211 before sending out uplink port.

```
rtk_vlan_egrVlanCnvtEntry_t    data ;
rtk_vlan_egrVlanCnvtRangeCheck_vid_t vRng;
int32 rngIdx;
/* set egress VLAN translation */
/* set customer uplink translation information (10->210, 11->211) */
rngIdx = 0;
osal_memset(&vRng, 0, sizeof(rtk_vlan_egrVlanCnvtRangeCheck_vid_t));
vRng. vid_lower_bound = 10;
vRng. vid_upper_bound = 11;
vRng. type = 0;
rngIdx = 0;
rtk_vlan_egrVlanCnvtRangeCheckVid_set(0, rngIdx, & vRng);
osal_memset(&data, 0, sizeof(rtk_vlan_egrVlanCnvtEntry_t));
/* search key setting */
data.vid_ignore = True;
data.orgpri_ignore = True;
data.port_ignore = True;
data.rngchk_result = (1 << rngIdx);
data.rngchk_result_mask = (1 << rngIdx);
data.vid_cnvt_sel = 0;
data.vid_new = 200;
data.vid_shift = True;
data.vid_shift_sel = 0;
data.valid = True;
rtk_vlan_egrVlanCnvtEntry_set(unit, 0, &data);

/* set customer downlink translation information (210->10, 211->11) */
rngIdx = 1;
osal_memset(&vRng, 0, sizeof(rtk_vlan_egrVlanCnvtRangeCheck_vid_t));
vRng. vid_lower_bound = 210;
vRng. vid_upper_bound = 211;
vRng. type = 0;
rtk_vlan_egrVlanCnvtRangeCheckVid_set(0, rngIdx, &vRng);
osal_memset(&data, 0, sizeof(rtk_vlan_egrVlanCnvtEntry_t));
/* search key setting */
data.vid_ignore = True;
data.orgpri_ignore = True;
data.port_ignore = True;
data.vid_ignore = True;
data.rngchk_result = (1 << rngIdx);
data.rngchk_result_mask = (1 << rngIdx);
data.vid_new = 200;
data.vid_shift = True;
data.vid_shift_sel = 1;
data.valid = True;
rtk_vlan_egrVlanCnvtEntry_set(unit, 1, &data);
```

Example 3: VLAN N:1 Mapping

Condition:
Packet from downlink port with inner VLAN 10~20, and translates to outer VLAN 300 before sending out uplink port.

```
rtk_vlan_egrVlanCnvtEntry_t    data ;
rtk_vlan_egrVlanCnvtRangeCheck_vid_t vRng;
int32 rngIdx;
/* set egress VLAN translation */
/* set customer uplink translation information (10~20->300) */
rngIdx = 0;
osal_memset(&vRng, 0, sizeof(rtk_vlan_egrVlanCnvtRangeCheck_vid_t));
vRng. vid_lower_bound = 10;
```

```
vRng. vid_upper_bound = 20;
rtk_vlan_egrVlanCnvtRangeCheckVid_set(0, rngIdx, & vRng);
osal_memset(&data, 0, sizeof(rtk_vlan_egrVlanCnvtEntry_t));
/* search key setting */
data.vid_ignore = True;
data.orgpri_ignore = True;
data.port_ignore = True;
data.rngchk_result = (1 << rngIdx);
data.rngchk_result_mask = (1 << rngIdx);
data.vid_new = 300;
data.valid = True;
rtk_vlan_egrVlanCnvtEntry_set(unit, 0, &data);

/* set uplinkport translation is on outer VID */
rtk_vlan_portEgrVlanCnvtVidTarget_set(0, uplinkPort, BASED_ON_OUTER_VLAN);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

3 Layer 2 switch

3.1 Layer 2 forwarding

The Layer 2 packet forwarding based on Layer 2 header is basis of a switch. Per system provide an Layer 2 table with 8K entries to record unicast/multicast MAC and port mapping. For multicast forwarding needs more than one destination port, system provide L2 port-Mask table with 512 entries for port mask (This table is also extended for VLAN profile and ACL reference).

The Layer 2 entry is able to set aging out range from 0.1 ~ 1.68M second.

System also provides Static, Dynamic Port Move and Port Move Forbidding mechanism to handle port movement for source port handling of recorded layer 2 entries.

When an incoming packet lookup miss, system also provides L2 Unicast Lookup Miss, L2 multicast Lookup Miss IPv4 Multicast Lookup Miss and IPv6 Multicast Lookup Miss operations, and is able to define flooding port mask independently by VIan.

For security concern, there are also mac constraint to limit system learning count, per-port base Layer 2 address learning count, and another 8 entries per-vlan and port address learning count.

As to manage connected user in the network, there is SA block and DA block supported to implement host block list. In addition, the SA block resource can change to SA secure as host permission list.

3.2 Layer 2 table

To support Layer 2 unicast and Layer 2 multicast, IPv4/IPv6 multicast and forwarding in 8K table (Routing entry also need to take Layer 2 table entry resource, detail may reference Routing Developer Guide), each type of entry have its own format.

3.2.1 Table format

The L2 table format will as follow:

Type	Hash Key		Table Content (width = 76bits)											
L2 Unicast	MAC (48)	FID/VID (12)	IP_Multi (1)	IP6 (1)	MAC (48)	FVID (1)	SLP (5)	Age(2)	SABLK (1)	DABLK (1)	Static (1)	Suspending (1)	Next hop (1)	VID (12)
L2 Multicast			IP_Multi (1)	IP6 (1)	MAC (48)	FVID (1)		IDX(9)	VID (12)					
IPv4/6 Multicast	GIP (32)	SIP (32)	IP_Multi (1)	IP6 (1)	GIP (32)	SIP (21)		IDX(9)	FVID (12)					
	0+GIP (20+32)	FID/VID (12)	IP_Multi (1)	IP6 (1)	GIP (32)	FVID (1)	Rsvd (16)	IDX(9)						

Definition of L2 Unicast Fields

Define: IP_Multi = 0b, MAC[40] bit = 0b

SLP: The source logical port.

Age: The aging time of this entry.

SABLK / SASecure: Source MAC blocking.

DABLK: Destination MAC blocking.

Static: Static Bit.

Next_Hop: The Next Hop entry. Routing Application note.

VID: The field is added to support N:1 VLAN aggregation(refer to VLAN spec). It is auto written by ASIC while learning a L2 unicast entry.

Table 10: L2 Unicast Entry Fields

Realtek CONFIDENTIAL

Field Name	Bits	Description
FID_RVID	12	FID/RVID
MAC	48	MAC address
RSVD	4	This field is reserved and perserved so that RTK API can clear this field.
IP_MC	1	IP Multicast 0b0: Non IP multicast entry 0b1: IP multicast entry
IP6_MC	1	IPv6 Multicast 0b0: Non IPv6 multicast entry 0b1: IPv6 multicast entry
SLP	5	Source Logical Port
AGE	2	Aging Time
SA_BLK	1	Source MAC address blocking
DA_BLK	1	Destination MAC address blocking
STATIC	1	Static entry
SUSPEND	1	Suspend (wait to be removed)
NEXT_HOP	1	Next Hop (need to routing)
VID	12	VLAN ID (used by VLAN aggregation)

Definition of L2 Multicast Fields

Define: IP_Multi = 0b, MAC[40] bit = 1b
IDX: A pointer to the index of L2 Multicast table.

Table 11: L2 Multicast Entry Fields

Field Name	Bits	Description
FID_RVID	12	FID/RVID
MAC	48	MAC address
IP_MC	1	IP Multicast 0b0: Non IP multicast entry 0b1: IP multicast entry
IP6_MC	1	IPv6 Multicast 0b0: Non IPv6 multicast entry 0b1: IPv6 multicast entry
MC_PMSK_IDX	9	Multicast Portmask Index
VID	12	VLAN ID (used by VLAN aggregation)

IPv4 Multicast Fields

Define: IP_Multi = 1b, IP6 = 0b
IDX: A pointer to the index of L2 Multicast table.

Table 12: L2 IP Multicast Entry Fields With SIP/GIP As Key

Field Name	Bits	Description
SIP	32	Source IP address
GIP	32	Group IP address (LSB 28bits)
IP_MC	1	IP Multicast 0b0: Non IP multicast entry 0b1: IP multicast entry
IP6_MC	1	IPv6 Multicast

Realtek CONFIDENTIAL CAMEO Communications, Inc.

		0b0: Non IPv6 multicast entry 0b1: IPv6 multicast entry
FID_RVID	12	FID/RVID
MC_PMSK_IDX	9	Multicast Portmask Index

Table 13: L2 IP Multicast Entry Fields With FID/GIP As Key

Field Name	Bits	Description
FID_RVID	12	FID/RVID
GIP	32	Group IP address (LSB 28bits)
IP_MC	1	IP Multicast 0b0: Non IP multicast entry 0b1: IP multicast entry
IP6_MC	1	IPv6 Multicast 0b0: Non IPv6 multicast entry 0b1: IPv6 multicast entry
MC_PMSK_IDX	9	Multicast Portmask Index

IPv6 Multicast Fields

Define: IP_Multi = 1b, IP6 = 1b
IDX: A pointer to the index of L2 Multicast table.

3.2.2 Access

Layer 2 table can access via table access, and system support to get/delete L2 entry by MAC+VID or by index.

API REFERENCE

```
rtk_l2_addr_add(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_del(uint32 unit, rtk_vlan_t vid, rtk_mac_t *pMac);
rtk_l2_addr_get(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_set(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_delAll(uint32 unit, uint32 include_static);
rtk_l2_nextValidAddr_get(uint32 unit, int32 *pScan_idx, uint32 include_static, rtk_l2_ucastAddr_t
*pL2_data);

rtk_l2_mcastAddr_add(uint32 unit, rtk_l2_mcastAddr_t *pMcast_addr);
rtk_l2_mcastAddr_del(uint32 unit, rtk_vlan_t vid, rtk_mac_t *pMac);
rtk_l2_mcastAddr_get(uint32 unit, rtk_l2_mcastAddr_t *pMcast_addr);
rtk_l2_mcastAddr_set(uint32 unit, rtk_l2_mcastAddr_t *pMcast_addr)
rtk_l2_mcastAddr_add_with_index(uint32 unit, rtk_l2_mcastAddr_t *pMcast_addr);
rtk_l2_mcastAddr_get_with_index(uint32 unit, rtk_l2_mcastAddr_t *pMcast_addr);

rtk_l2_ipMcastAddr_add(uint32 unit, rtk_l2_ipMcastAddr_t *pIpmcast_addr);
rtk_l2_ipMcastAddr_del(uint32 unit, ipaddr_t sip, ipaddr_t dip, rtk_vlan_t vid);
rtk_l2_ipMcastAddr_get(uint32 unit, rtk_l2_ipMcastAddr_t *pIpmcast_addr);
rtk_l2_ipMcastAddr_set(uint32 unit, rtk_l2_ipMcastAddr_t *pIpmcast_addr);
rtk_l2_ipMcastAddr_add_with_index(uint32 unit, rtk_l2_ipMcastAddr_t *pIpMcast_addr);
rtk_l2_ipMcastAddr_get_with_index(uint32 unit, rtk_l2_ipMcastAddr_t *pIpMcast_addr);
rtk_l2_nextValidMcastAddr_get(uint32 unit, int32 *pScan_idx, rtk_l2_mcastAddr_t *pL2_data);
rtk_l2_nextValidIpMcastAddr_get(uint32 unit, int32 *pScan_idx, rtk_l2_ipMcastAddr_t *pL2_data);
```

Realtek CONFIDENTIAL

```
rtk_l2_ip6McastAddr_add(uint32 unit, rtk_l2_ip6McastAddr_t *pIpmcast_addr);
rtk_l2_ip6McastAddr_del(uint32 unit, rtk_ipv6_addr_t sip, rtk_ipv6_addr_t dip, rtk_vlan_t vid);
rtk_l2_ip6McastAddr_get(uint32 unit, rtk_l2_ip6McastAddr_t *pIpmcast_addr);
rtk_l2_ip6McastAddr_set(uint32 unit, rtk_l2_ip6McastAddr_t *pIpmcast_addr);
rtk_l2_ip6McastAddr_add_with_index(uint32 unit, rtk_l2_ip6McastAddr_t *pIpMcast_addr);
rtk_l2_ip6McastAddr_get_with_index(uint32 unit, rtk_l2_ip6McastAddr_t *pIpMcast_addr);
rtk_l2_nextValidIp6McastAddr_get(uint32 unit, int32 *pScan_idx, rtk_l2_ip6McastAddr_t *pL2_data);
```

3.2.3 Flush

System provides control register L2_TBL_FLUSH_CTRL to flush L2 table entries, user can set FVID_CMP and PORT_CMP to specify which FID and/or Port should be cleared or replaced. ENTRY_TYPE can be used to determine whether clear the software entry. All of them are regarded as additional conditions. If ACT is triggered without specifying FVID_CMP, PORT_CMP, all unicast entries in L2 table will be flushed/replaced.

Table 14: L2_TBL_FLUSH_CTRL Register

Field Name	Bits	Description
STS	1	Action status. It is used to trigger ASIC to start flush/replace action. It will be auto cleared after flush/replace process is finished. 0b0: finish 0b1: start (busy)
ACT	1	The action to take. 0b0: flush 0b1: replace
FVID_CMP	1	Set this bit 1 to flush/replace by specified FID/VID.
PORT_CMP	1	Set this bit 1 to flush/replace by specified port.
ENTRY_TYPE	1	0b0: Whole L2 table (include dynamic and static entries). 0b1: Dynamic L2 unicast entry
FVID	12	Assign one FID/VID that will be flushed/replaced.
PORT	5	Assign the port id that will be flushed (Act = 0) or replaced (Act = 1).
REPLACING_PORT	5	Assign the port id as the new port (This field meanful only when Act = 1).

API REFERENCE

```
rtk_l2_ucastAddr_flush(uint32 unit, rtk_l2_flushCfg_t *pConfig);
```

3.2.4 Aging

Each unicast L2 table entry has the "Age" field. This field will be updated when learned by system or SA lookup hits. When the value decreases to zero, the entry is aged out.

System provide global configuration to adjust the period through L2_CTRL_1.AGE_UNIT. The adjustable range of aging out period is about 0.1s ~ 1.68M seconds. The actual aging time will be around L2_CTRL_1.AGE_UNIT *2 ~ L2_CTRL_1.AGE_UNIT *3.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 15: L2_CTRL_1.AGE_UNIT Register

Field Name	Bits	Description
AGE_UNIT	23	The granularity of aging bits in L2 table. Aging bits reduce one every AGEUNIT* 0.1 seconds. Disable aging when AGE_UNIT == 0.

The L2 table entries learnt from a port can be optionally aged-out by per-port setting: L2_PORT_AGING_OUT. AGING_OUT_EN.

Table 16: L2_PORT_AGING_OUT.AGING_OUT_EN Register

Field Name	Bits	Description
AGING_OUT_EN	1	Decide whether aging out those MACs associated to this port after a period of time. 0: disable 1: enable

In addition, there is L2_CTRL_0.LINK_DOWN_P_INVLD (1bit) used to determine whether invalidate the corresponding L2 entry when a port has link down.

Table 17: L2_CTRL_0.LINK_DOWN_P_INVLD Register

Field Name	Bits	Description
LINK_DOWN_P_INVLD	1	Invalidate the L2 entries whose associated ports are link down 0b0: disable 0b1: enable

API REFERENCE

```
rtk_l2_aging_get(uint32 unit, uint32 *pAging_time);
rtk_l2_aging_set(uint32 unit, uint32 aging_time);
rtk_l2_portAgingEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable) ;
rtk_l2_portAgingEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable) ;
rtk_l2_flushLinkDownPortAddrEnable_get(uint32 unit, rtk_enable_t *pEnable);
rtk_l2_flushLinkDownPortAddrEnable_set(uint32 unit, rtk_enable_t enable);
```

3.2.5 Address Lookup and Learning

When packet received, the packet will be parsed to find SA and DA to lookup in L2 table, if the SA is new in this device, a layer 2 entry with this SA and incoming port will be learnt. Then use DA and some catachrestic to judge packet type to L2 unicast, L2 multicast, IPv4 multicast or IPv6 multicast. After packet type is defined, the corresponding search key will be used to find matched Layer 2 entry to get the destination port information. If the lookup process fail to find any match entry, the lookup miss process will performed.

Address Lookup

SA Lookup is the method of verifying whether unicast SA is existed. If unicast SA exists, system will get information from this entry for decision later. If not, Auto Learning proceeds. FID/VID and MAC are used as hash key to do SA lookup since only unicast SA would do SA lookup process.

DA lookup is the method of searching L2 table and decides which port the packet should be forwarded. Since DA could be unicast, multicast (L2/ IPv4/ IPv6), or broadcast and the hash key of different types of packets are different, the contents of the pakets should be parsed to detect the type and decide the look up hash key.

Realtek CONFIDENTIAL CAMEO Communications

The DA lookup process of different type of traffic is as follows:

- Broadcast

1. DMAC = FF-FF-FF-FF-FF-FF
2. System will flood this packet according the L2_FLD_PMSK.L2_BC_FLD_PMSK configuration.

- IPv4 Multicast:

1. DMAC = 01-00-5E-XX-XX-XX
2. DMAC[23] = 0
3. Ethertype = 0x0800
4. If IP_MC_DIP_CHK = 1, check whether DIP belongs to Class-D scope (224~239.XX.XX.XX), and DIP[22:0] = DMAC[22:0].
5. This packet belongs to neither EAV Class A nor Class B traffic.(i.e., HSB.EAV_CLASS_A = 0 and HSB.EAV_CLASS_B = 0). Because EAV traffic is registered by mac address, hence should not lookup by IP address.
6. Search key={FID/VID, MAC} / {SIP, GIP} / {FID/VID, 0, GIP} defined by L2_CTRL_0. IPV4_MC_HASH_KEY_FMT
7. If hits, then forward this packet according to the portmask which IDX points to.
8. If not hit, take action as what IP_MC_LM_ACT indicates even if IPV4_MC_HASH_KEY_FMT is set to L2 Multicast mode.

- IPv6 Multicast:

1. DMAC = 33-33-XX-XX-XX-XX
2. Ethertype = 0x86DD
3. This packet belongs to neither EAV Class A nor Class B traffic. (i.e., HSB.EAV_CLASS_A = 0 and HSB.EAV_CLASS_B = 0)
4. Search key={FID/VID, MAC} / {SIP, GIP} / {FID/VID, 0, GIP} defined by L2_CTRL_0. IPV6_MC_HASH_KEY_FMT.
5. If hits, then forward this packet according to the portmask which IDX points to.
6. If not hit, take action as what IP_MC_LM_ACT indicates even if IPV6_MC_HASH_KEY_FMT is set to L2 Multicast mode.

For different IGMP snooping and IPv6 forwarding, system supports three IPv4 and IPv6 Multicast modes configured by IPV4_MC_HASH_KEY_FMT and IPV6_MC_HASH_KEY_FMT:

1. {FID/VID, MAC} Hash mode:

Tread IPv4/v6 Multicast as L2 Multicast. The fields are the same as that of L2 Multicast entry.

2. {SIP, GIP} Hash mode:

- Use the SIP(32) and GIP(32) of an IPv4 Multicast packet as the Hash Key.
- For IPv6 multicast packet, take the 4 bytes of SIP and GIP respectively in IPv6 Multicast packet as the Hash Key. The IPV6_MC_CARE_BYTE.SIP_CARE_BYTE and IPV6_MC_CARE_BYTE.DIP_CARE_BYTE are both four 4-bit nibbles which map to 4 of IPv6 IP's 16 byte. ASIC use them to decide which 4 Bytes in SIP and DIP are taken. Defaultly, we take bytes [5][2][1][0] from SIP, [3][2][1][0] from DIP as the hash key. Thus the value of SIP_CARE_BYTE is 0x5210 while DIP_CARE_BYTE is 0x3210. Note that CARE_BYTE[0] maps to hash key[7:0], CARE_BYTE[1] maps to hash key[15:8], CARE_BYTE[2] maps to hash key[23:16] and CARE_BYTE[3] maps to hash key[31:24]. Software should guarantee the correctness of CARE_BYTE.

The indexing of system for SIP/DIP byte is as the following diagram, so the value of byte[5][2][1][0] is 0x11445566:

3. {FID/VID, 0, GIP} Hash mode:

Use the FID/VID and GIP(32) of an IPv4 Multicast packet as the Hash Key. For IPv6 multicast packet, the 4 bytes GIP retrieving is same as {SIP, GIP} Hash mode.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 18: IPV4/6_MC_HASH_KEY_FMT Register

Field Name	Bits	Description
IPV4_MC_HASH_KEY_FMT	2	IPv4 multicast hash key format. 0b00: same as L2 multicast (FID/VID + MAC) 0b01: SIP + GIP 0b10: FID/VID + 0 + GIP 0b11: reserved
IPV6_MC_HASH_KEY_FMT	2	IPv6 multicast hash key format. 0b00: same as L2 multicast (FID/VID + MAC) 0b01: SIP + GIP 0b10: FID/VID + 0 + GIP 0b11: reserved

Table 19: L2_IPV6_MC_IP_CARE_BYTE Register

Field Name	Bits	Description
DIP_CARE_BYTE	16	Cared bytes mask of DIP of IPv6 multicast packets that would be taken as DIP hash key. Four nibbles represent the positions of 4 bytes in DIP. Software must make sure the correctness.
SIP_CARE_BYTE	16	Cared bytes mask of SIP of IPv6 multicast packets that would be taken as SIP hash key. Four nibbles represent the positions of 4 bytes in SIP. Software must make sure the correctness.

API REFERENCE

```
rtk_l2_ipmcMode_get(uint32 unit, rtk_l2_ipmcMode_t *pMode);
rtk_l2_ipmcMode_set(uint32 unit, rtk_l2_ipmcMode_t mode);
rtk_l2_ip6mcMode_get(uint32 unit, rtk_l2_ipmcMode_t *pMode);
rtk_l2_ip6mcMode_set(uint32 unit, rtk_l2_ipmcMode_t mode);
rtk_l2_ip6CareByte_get(uint32 unit, rtk_l2_ip6_careByte_type_t type, uint32 *pCareByte);
rtk_l2_ip6CareByte_set(uint32 unit, rtk_l2_ip6_careByte_type_t type, uint32 careByte);
```

Address Learning

The packet will be learnt follow per-port configuration L2_PORT_SALRN.SALRN and if this SA is new the packet will be tread follow the action that L2_PORT_NEW_SA_FWD. NEW_SA_FWD defines.

When the SALRN = 0 (auto learn), only the first packet which is new to ASIC will apply the NEW_SA_FWD. For the same SMAC packets comes later, they are no longer new to ASIC so they apply normal forwarding behavior.

When the SALRN = 1 (learn as suspend), ASIC will write the L2 entry as usual and set the Suspend bit to 1 additionally.

The suspending entry has the following characteristic:

- It could be timed out after aging time.
- It should not be replaced by other SMAC before it times out.
- For the 1st packet, ASIC takes action according to NEW_SA_FWD.

For the 2nd, 3rd… packets, they are still regarded as new SMAC so that ASIC updates the L2.age field but take action partly according to NEW_SA_FWD:

(i). NEW_SA_FWD = 0, => Forward
(ii). NEW_SA_FWD = 1, => Drop
(iii). NEW_SA_FWD = 2, => Drop
(iv). NEW_SA_FWD = 3, => Forward

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 20: L2_PORT_SALRN.SALRN Special SA Control Register

Field Name	Bits	Description
SALRN	1	L2 table learning behavior when receives a packet with new SMAC. 00b: ASIC Auto Learn 01b: learn as a suspend entry 10b: not Learn 11b: reserved

Table 21: L2_PORT_NEW_SALRN. NEW_SA_FWD Special SA Control Register

Field Name	Bits	Description
NEW_SA_FWD	2	Packet forwarding behavior when receives a packet with new SMAC. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

If SA is Broadcast (FF-FF-FF-FF-FF-FF) or Multicast (MAC[40] = 1), SA Lookup/Learning is not performed and use the configuration L2_CTRL_0.MC_BC_SA_DROP and L2_CTRL_0.MC_BC_SA_TRAP to decide the forwarding behavior for the packet:

- If L2_CTRL_0.MC_BC_SA_TRAP is 1, the BC/MC SA packet is trapped.
- If L2_CTRL_0.MC_BC_SA_ TRAP is 0 and L2_CTRL_0.MC_BC_SA_DROP is 1, the packet is dropped.
- Otherwise, the packet is forwarded.

And if SA is all zero, system use the configuration L2_CTRL_0.ZERO_SA_ACT to decide the forwarding behavior for the packet:

Whethert to perform SA Learning depends on the configuration of L2_CTRL_0.SA_ALL_ZERO_LRN.

Table 22: L2_CTRL_0 Special SA Control Register

Field Name	Bits	Description
MC_BC_SA_TRAP	1	Decide the trap action of the packet whose SA is multicast or broadcast SA If enable Trap, ignore MC_BC_SA_DROP setting 0b00: Disable(not Trap) 0b01: Enable(Trap)
MC_BC_SA_DROP	1	Decide the drop action of the packet whose SA is multicast or broadcast SA 0b00: Disable(not Drop) 0b01: Enable(Drop)
ZERO_SA_ACT	2	Decide the action of the packet whose SA is all 0 0b00: Forward 0b01: Drop 0b10: Trap 0b11: Reserved
SA_ALL_ZERO_LRN	1	The learning behavior for an all-zero mac-address (00-00-00-00-00-00). 0b0: not learn 0b1: learn

Realtek CONFIDENTIAL CAMEO Communications, Inc.

API REFERENCE

```
rtk_l2_newMacOp_get(uint32 unit, rtk_port_t port, rtk_l2_newMacLrnMode_t *pLrnMode, rtk_action_t *pFwdAction);
rtk_l2_newMacOp_set(uint32 unit, rtk_port_t port, rtk_l2_newMacLrnMode_t lrnMode, rtk_action_t fwdAction) ;
rtk_l2_exceptionAddrAction_get(uint32 unit, rtk_l2_exceptionAddrType_t exceptType, rtk_action_t *pAction) ;
rtk_l2_exceptionAddrAction_set(uint32 unit, rtk_l2_exceptionAddrType_t exceptType, rtk_action_t action) ;
```

3.2.6 Lookup miss

For those lookup miss packets system, system has the following action:

(1) Normal forward (i.e. Flooding).
(2) Drop.
(3) Trap to CPU.
(4) Copy to CPU

For Unicast, L2 Multicast, IPv4 Multicast, IPv6 Muticast packets, system has separated configurations (through L2_PORT_LM_ACT.L2_UC_LM_ACT, L2_PORT_LM_ACT.L2_MC_LM_ACT, L2_PORT_LM_ACT.IP_MC_LM_ACT and L2_PORT_LM_ACT.IP6_MC_LM_ACT register).

If ACT is forward, the forwarding port mask is decided by Flooding to L2_UNKN_UC_FLD_PMSK, L2_UNKN_MC_FLD_PMSK, IP4_UNKN_MC_FLD_PMSK, or IP6_UNKN_MC_FLD_PMSK for unknown unicast, L2 unknown multicast, IPv4 unknown multicast, and IPv6 unknown multicast packets respectively. It should be noted that L2_UNKN_UC_FLD_PMSK is an L2 global register but L2_UNKN_MC_FLD_PMSK, IP4_UNKN_MC_FLD_PMSK, and IP6_UNKN_MC_FLD_PMSK are fields in VLAN profile.

Table 23: L2_PORT_LM_ACT Register

Field Name	Bits	Description
L2_UC_LM_ACT	2	L2 unicast Packet lookup miss action Note: A L2 unicast packet is deemed to KNOWN if its DMAC and forwarding VID are equal to switch's MAC and the PVID of CPU port respectively. 0b00: forward (Flooding to L2_UNKN_UC_FLD_PMSK) 0b01: drop 0b10: trap 0b11: copy to CPU
L2_MC_LM_ACT	2	L2 Multicast Packet lookup miss action *L2 Multicast Packet means DMAC[40]=1 0b00: forward (Flooding to L2_UNKN_MC_FLD_PMSK) 0b01: drop 0b10: trap 0b11: copy to CPU
IP_MC_LM_ACT	2	IPv4 Multicast Packet lookup miss action 0b00: forward (Flooding to IP4_UNKN_MC_FLD_PMSK) 0b01: drop 0b10: trap 0b11: copy to CPU
IP6_MC_LM_ACT	2	IPv6 Multicast Packet lookup miss action 0b00: forward (Flooding to IP6_UNKN_MC_FLD_PMSK)

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	0b01: drop 0b10: trap 0b11: copy to CPU

Table 24: Lookup Miss Forwarding Portmask Register

Field Name	Bits	Description
L2_UNKN_UC_FLD_PMSK	9	An index points to multicast table for retrieving unknown unicast flooding port mask.
L2_UNKN_MC_FLD_PMSK	9	An index points to multicast table for retrieving the unknown L2 multicast flooding port mask.
IP4_UNKN_MC_FLD_PMSK	9	An index points to multicast table for retrieving unknown IPv4 multicast flooding port mask.
IP6_UNKN_MC_FLD_PMSK	9	An index points to multicast table for retrieving unknown IPv6 multicast flooding port mask.

API REFERENCE

*rtk_l2_portLookupMissAction_get(uint32 unit, rtk_port_t port, rtk_l2_lookupMissType_t type, rtk_action_t *pAction);*
rtk_l2_portLookupMissAction_set(uint32 unit, rtk_port_t port, rtk_l2_lookupMissType_t type, rtk_action_t action) ;
*rtk_l2_lookupMissFloodPortMask_get(uint32 unit, rtk_l2_lookupMissType_t type, rtk_portmask_t *pFlood_portmask);*
rtk_l2_lookupMissFloodPortMask_add(uint32 unit, rtk_l2_lookupMissType_t type, rtk_port_t flood_port) ;
rtk_l2_lookupMissFloodPortMask_del(uint32 unit, rtk_l2_lookupMissType_t type, rtk_port_t flood_port) ;
*rtk_l2_lookupMissFloodPortMask_set_with_idx(uint32 unit, rtk_l2_lookupMissType_t type, uint32 idx, rtk_portmask_t *pFlood_portmask) ;*

3.2.7 Multicast Forwarding Table

The table is used for multicast port mapping and also share with ACL and VLAN profile port mask assignment. So the table size is 512, each entry can assign port mask. All Layer 2 multicast entry, ACL multi-port redirect or VLAN profile entry index to the associated entry.

Table 3.2: L2 Port-Mask Table

Index [8:0]	Port Mask [28:0]
0	
1	
2	
511	

Table 25: Multicast Forwarding Portmask Format

Field Name	Bits	Description
MC_PMSK	29	Multicast port mask.

CONFIDENTIAL
CAMEO Communications, Inc.

API REFERENCE

```
rtk_l2_mcastFwdPortmask_get(uint32 unit, int32 index, rtk_portmask_t *pPortmask, uint32 *pCrossVlan) ;
rtk_l2_mcastFwdPortmask_set(uint32 unit, int32 index, rtk_portmask_t *pPortmask, uint32 crossVlan) ;
```

*: crossVlan parameter is not supported in rtl8380

3.3 Port move

3.3.1 Port move for static entry

For static mac entry, there is a per-port setting L2_PORT_STATIC_MV_ACT.ACT that will check all static layer 2 entries with SLP on this port, if there is packet come from any of the other ports, the packet will tread follow L2_PORT_STATIC_MV_ACT.ACT configuration of original SLP.

For example, let an entry with Static = 1, SLP = 3 and ACT of port 3 is 2 (trap) . If a packet match this entry but come from port 5, then it will be trap to CPU.

Table 26: L2_PORT_STATIC_MV_ACT Register

Field Name	Bits	Description
ACT	2	Port move action. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

API REFERENCE

```
rtk_l2_staticPortMoveAction_get(uint32 unit, rtk_port_t port, rtk_action_t *pFwdAction) ;
rtk_l2_staticPortMoveAction_set(uint32 unit, rtk_port_t port, rtk_action_t fwdAction) ;
```

3.3.2 Port move for dynamic entry

For dynamic port move, system has per-port setting L2_PORT_MV_ACT.ACT. When port moving occurs, the setting of L2_PORT_MV_ACT.ACT of moved-to-port (new port) would be refered to forward (normal action), or drop/trap/copy to CPU (copy to CPU will be forward and trap). It should be noted that SLP and Aging time will only be updated when action is forward(0) or copy to CPU(3).

Table 27: L2_PORT_MV_ACT Register

Field Name	Bits	Description
ACT	2	Port move action. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

And a special design for web authentication application may useful, the per-port L2_PORT_MV_INVALIDATE.P_MV_INVLD is used to remove the L2 entry when receiving packets from ports other than original SLP, This configuration also follows the moved-to-port (new port) setting.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 28: L2_PORT_MV_INVALIDATE Register

Field Name	Bits	Description
P_MV_INVLD	2	Port move entry invalidate 0b00: disable 0b01: enable

API REFERENCE

*rtk_l2_legalPortMoveAction_get(uint32 unit, rtk_port_t port, rtk_action_t *pFwdAction) ;*
rtk_l2_legalPortMoveAction_set(uint32 unit, rtk_port_t port, rtk_action_t fwdAction);
*rtk_l2_legalPortMoveFlushAddrEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable) ;*
rtk_l2_legalPortMoveFlushAddrEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable) ;

3.4 Learning Constraint

System provide system, per-port based and 8 vlan based learning constraint, When a port receives a packet with new SMAC, ASIC will check all the learned numbers of the system, per-port, and vlan at same time. When any one of them reaches its own constraint number first, the ASIC will stop learning this MAC. Then take the corresponding constraint action with this packet, if more than one constraint number overflow at the same time, the action priority will be: VLAN > port > whole system.

Here is an example,

(1) If a packet trigger port's and system's constraint action at the same time, and if ACT on port is forward and on system is drop, then the packet will be forwarded.

(2) If a packet trigger VLAN's and system's constraint action at the same time, and if ACT on VLAN is drop and on system is forward, then the packet will be dropped.

The following entries should not increase learning counter:

- L2. IP_Multi = 1
- Mac[40] = 1
- L2.Static = 1
- L2.DABlock = 1
- L2.SABlock = 1
- L2.SASecure = 1
- Suspending Entry
- Next_hop Entry

It should be noted that the global/per-port/per-VLAN learning constraint only takes effect when both L2_PORT_SALRN and L2_PORT_NEW_SA_FWD of the packet receiving port are set to 0 (ASIC Auto learn, forward), but ASIC maintain the counter in any configuration.

3.4.1 System Learning Constraint

System provides global register L2_LRN_CONSTRT with CONSTRT_NUM and LRN_CNT. If the LRN_CNT counter counts over the CONSTRT_NUM, the packet with new SA should not be learnt and takes action according to L2_LRN_CONSTRT.ACT.

Table 29: L2_LRN_CONSTRT Register

Field Name	Bits	Description

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

LRN_CNT	14	Learnt count.
CONSTRT_NUM	14	Learning constraint number. 0x0: never learning 0x3FFF: unlimited
ACT	2	Learning constraint Action. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

API REFERENCE

```
rtk_l2_limitLearningCnt_get(uint32 unit, uint32 *pMac_cnt);
rtk_l2_limitLearningCnt_set(uint32 unit, uint32 mac_cnt);
rtk_l2_limitLearningCntAction_get(uint32 unit, rtk_l2_limitLearnCntAction_t *pAction) ;
rtk_l2_limitLearningCntAction_set(uint32 unit, rtk_l2_limitLearnCntAction_t action) ;
```

3.4.2 Per-Port Learning Constraint

System provides per-port 15 bit CONSTRT_NUM and 15 bits LRN_CNT register to limit and display L2 leant number. Whenever ASIC learns a SMAC from a port, it increases LRN_CNT counter of the port. If the counter counts over the CONSTRT_NUM, the new SMAC will not be learnt and takes action according to L2_PORT_LRN_CONSTRT.ACT.

If user moves to another port, the learned counter should decrease the original port and increase the new port. Suppose user decreases the limit so that the learnt counter beyond it, ASIC should not learn the new SMAC but still forward those learnt packets and update the L2 table. And if user moves to another lower-limited port, ASIC does NOT update it SLP and Age fields but let it time-out naturally. Besides, before age-out the action for incoming packets of this MAC follows the L2_PORT_LRN_CONSTRT.ACT of new port.

Table 30: L2_PORT_LRN_CONSTRT Register

Field Name	Bits	Description
LRN_CNT	14	Learnt count.
CONSTRT_NUM	14	Learning constraint number. 0x0: never learning 0x3FFF: unlimited
ACT	2	Learning constraint Action. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

API REFERENCE

```
rtk_l2_portLimitLearningCnt_get(uint32 unit, rtk_port_t port, uint32 *pMac_cnt);
rtk_l2_portLimitLearningCnt_set(uint32 unit, rtk_port_t port, uint32 mac_cnt);
rtk_l2_portLimitLearningCntAction_get(uint32 unit, rtk_port_t port, rtk_l2_limitLearnCntAction_t *pAction) ;
rtk_l2_portLimitLearningCntAction_set(uint32 unit, rtk_port_t port, rtk_l2_limitLearnCntAction_t action) ;
```

3.4.3 VLAN Based Learning Constraint

Besides learning constraint on port, system also provides a table of size 8 for learning constraint on VLAN (or VLAN and port). The table entry definition is as follows:

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 31:VLAN Based Learning Constrant Entry Format

Field Name	Bits	Description
FID/VID	12	The FID/VID to be compared.
PORT	5	The Port Id to be compared. 0x1f: Don't care
LRN_CNT	14	Learnt count.
CONSTRT_NUM	14	Learning constraint number. 0x0: never learning 0x3FFF: unlimited

And global define the constraint action behavior:

Table 32:VLAN Based Learning Constrant Action Register

Field Name	Bits	Description
ACT	2	Learning constraint on VLAN Action. System wiil perform this action when a new-SA packet comes after the learnt number reaches the learning constraint. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

Before a VLAN constraint is configured, user must flush those related L2 entries and clear the LRN_CNT to 0.

API REFERENCE

```
rtk_l2_fidLimitLearningEntry_get(uint32 unit, uint32 fid_macLimit_idx, rtk_l2_fidMacLimitEntry_t *pFidMacLimitEntry);
rtk_l2_fidLimitLearningEntry_set(uint32 unit, uint32 fid_macLimit_idx, rtk_l2_fidMacLimitEntry_t *pFidMacLimitEntry);
rtk_l2_fidLearningCnt_get(uint32 unit, uint32 fid_macLimit_idx, uint32 *pNum) ;
rtk_l2_fidLearningCnt_reset(uint32 unit, uint32 fid_macLimit_idx) ;
rtk_l2_fidLearningCntAction_get(uint32 unit, rtk_l2_limitLearnCntAction_t *pAction);
rtk_l2_fidLearningCntAction_set(uint32 unit, rtk_l2_limitLearnCntAction_t action);
```

3.5 Blocking

In Layer 2 table unicast entry format, there are SA Block/SA Secure bit and DA block bit for SA Block/SA Secure and DA block function. SA Block and SA Secure share the same bit of the table entry and is switched by global configuration register L2_CTRL_0.SECURE_SA.

3.5.1 SA Block

When L2_CTRL_0.SECURE_SA is set to 0 and the SA block bit of a L2 table entry is set to 1, the packet which source MAC is the same as that of this entry will be dropped. This kind of entries could be time-out but the entry will not be removed from L2 table. Instead, the SLP will set to 0x1f. Initializes such entries with SLP = 0x1f to can flood the packet with DA hit this entry to all port to make sure that other host can still send packet to the SA blocked hosts (SA Block bit will default set to 0 when ASIC auto learn).

Realtek CONFIDENTIAL CAMEO

Table 33: L2_CTRL_0.SECURE_SA Register

Field Name	Bits	Description
SECURE_SA	1	Change L2 table to secure SA mode. 0b0: disable 0b1: enable

API REFERENCE

```
rtk_l2_secureMacMode_get(uint32 unit, rtk_enable_t *pEnable);
rtk_l2_secureMacMode_set(uint32 unit, rtk_enable_t enable);
```

3.5.2 SA Secure Enable

When L2_CTRL_0.SECURE_SA is set to 1, system will block all traffic by default, only the SA Secure bit of a L2 table entry is set to 1 can permitted to forward to it destination, all the other traffic which matched entries whose SA Secure bit is 0 will be dropped (SA Secure bit will default set to 0 when ASIC auto learn).

Table 34: L2_CTRL_0.SECURE_SA Register

Field Name	Bits	Description
SECURE_SA	1	Change L2 table to secure SA mode. 0b0: disable 0b1: enable

API REFERENCE

```
rtk_l2_secureMacMode_get(uint32 unit, rtk_enable_t *pEnable);
rtk_l2_secureMacMode_set(uint32 unit, rtk_enable_t enable);
```

3.5.3 DA Block

When DA block of a Layer 2 table entry is set to 1, the packet whose destination MAC lookup hits this entry will be dropped (But SMAC still be learned because the SA Learning is earlier than DA Lookup).

3.6 Application Example

Here are some applications mostly used in a layer 2 switch for quick reference.

3.6.1 Add/Delete Static entry

To add a static entry with MAC=00:11:22:33:44:55, VID=1

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_ucastAddr_t l2_uAddr;
uint32 sa_block = FALSE;
uint32 da_block = FALSE;
uint32 is_static = TRUE;
uint32 nexthop = FALSE;
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
uint32 suspend = FALSE;

/* unicast entry configuration */
memcpy(l2_uAddr.mac.octet, 0x001122334455, 6);
l2_uAddr.vid = 1;
l2_uAddr.port = 0;
l2_uAddr.agg_vid = 0 ;
l2_uAddr.flags |= sa_block << RTK_L2_UCAST_FLAG_SA_BLOCK;
l2_uAddr.flags |= da_block << RTK_L2_UCAST_FLAG_DA_BLOCK;
l2_uAddr.flags |= is_static << RTK_L2_UCAST_FLAG_STATIC;
l2_uAddr.flags |= nexthop << RTK_L2_UCAST_FLAG_NEXTHOP;
l2_uAddr.state |= suspend << RTK_L2_UCAST_STATE_SUSPEND;

/* add unicast entry */
rtk_l2_addr_add(unit, &l2_uAddr);
```

To delete a static entry with MAC=00:11:22:33:44:55, VID=1

```
/* variable declaration */
uint32 unit = 0;
rtk_mac_t mac;

/* unicast entry configuration */
memcpy(mac.octet, 0x001122334455, 6);

/* delete unicast entry */
rtk_l2_addr_del(unit, 1, & mac);
```

3.6.2 Add/Delete L2 Multicast entry

To add a L2 multicast entry with MAC=01:00:22:33:44:55, VID=1, forward portmask=port 1,3,5:

```
/* variable declaration */
rtk_l2_mcastAddr_t    mcast_data;

/* multicast entry configuration */
memcpy(mcast_data.mac.octet, 0x010022334455, 6);
mcast_data.rvid = 1;
mcast_data.portmask.bits[0] = (1 << 0) | (1 << 2) | (1 << 4);

/* add multicast entry */
rtk_l2_mcastAddr_add(unit, &mcast_data);
```

To delete a L2 multicast entry with MAC=01:00:22:33:44:55, VID=1:

```
/* variable declaration */
uint32 unit = 0;
rtk_mac_t mac;

/* multicast entry configuration */
memcpy(mac.octet, 0x010022334455, 6);

/* delete multicast entry */
rtk_l2_mcastAddr_del (unit, 1, & mac);
```

CONFIDENTIAL

3.6.3 Add/Delete IPv4 Multicast entry

To add an IPv4 multicast entry with MAC+VID for MAC=01:00:22:33:44:55, VID=1, forward portmask=port 1,3,5:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_mcastAddr_t mcast_data;

/* multicast entry configuration */
memcpy(mcast_data.mac.octet, 0x010022334455, 6);
mcast_data.rvid = 1;
mcast_data.portmask.bits[0] = (1 << 0) | (1 << 2) | (1 << 4);

/* set ip multicast hash mode to MAC+VID */
rtk_l2_ipmcMode_set(unit, LOOKUP_ON_FVID_AND_MAC)

/* add multicast entry */
rtk_l2_mcastAddr_add(unit, &mcast_data);
```

To add an IPv4 multicast entry with GIP+SIP for SIP=140.0.0.1,DIP=239.1.1.1, forward portmask=port 2,4,6:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_ipMcastAddr_t ip_mcast_data;

/* multicast entry configuration */
ip_mcast_data.sip = (140 << 24) |( 0 << 16) | (0 << 8) | 1;
ip_mcast_data.dip = (239 << 24) |( 1 << 16) | (1 << 8) | 1;
ip_mcast_data.portmask.bits[0] = (1 << 1) | (1 << 3) | (1 << 5);

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ipmcMode_set(unit, LOOKUP_ON_DIP_AND_SIP)

/* add multicast entry */
rtk_l2_ipMcastAddr_add(unit, &ip_mcast_data);
```

To add an IPv4 multicast entry with VID+GIP for VID=100, DIP=239.1.1.1, forward portmask=port 2,4,6:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_ipMcastAddr_t ip_mcast_data;

/* multicast entry configuration */
ip_mcast_data.rvid = 100;
ip_mcast_data.dip = (239 << 24) |( 1 << 16) | (1 << 8) | 1;
ip_mcast_data.portmask.bits[0] = (1 << 1) | (1 << 3) | (1 << 5);

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ipmcMode_set(unit, LOOKUP_ON_DIP_AND_FVID)
```

```
/* add multicast entry */
rtk_l2_ipMcastAddr_add(unit, &ip_mcast_data);
```

To delete an IPv4 multicast entry with GIP+SIP for SIP=140.0.0.1,DIP=239.1.1.1:

```
/* variable declaration */
uint32 unit = 0;
uint32 sip;
uint32 dip;

/* multicast entry configuration */
sip = (140 << 24) |( 0 << 16) | (0 << 8) | 1;
dip = (239 << 24) |( 1 << 16) | (1 << 8) | 1;

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ipmcMode_set(unit, LOOKUP_ON_DIP_AND_SIP)

/* delete multicast entry */
rtk_l2_ipMcastAddr_del(unit, sip, dip, 0);
```

3.6.4 Add/Delete IPv6 Multicast entry

To add an IPv6 multicast entry with MAC+VID for MAC=01:00:22:33:44:55, VID=1, forward portmask=port 1,3,5:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_mcastAddr_t mcast_data;

/* multicast entry configuration */
memcpy(mcast_data.mac.octet, 0x010022334455, 6);
mcast_data.rvid = 1;
mcast_data.portmask.bits[0] = (1 << 0) | (1 << 2) | (1 << 4);

/* set ip multicast hash mode to MAC+VID */
rtk_l2_ip6mcMode_set( unit, LOOKUP_ON_FVID_AND_MAC)

/* add multicast entry */
rtk_l2_mcastAddr_add(unit, &mcast_data);
```

To add an IPv6 multicast entry with GIP+SIP for SIP=2001::73, DIP= ff09::5, forward portmask=port 2,4,6:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_ip6McastAddr_t ip6_mcast_data;

/* multicast entry configuration */
ip_mcast_data.sip.ipv6_addr[0] = 0x20;
ip_mcast_data.sip.ipv6_addr[1] = 0x01;
ip_mcast_data.sip.ipv6_addr[15] = 0x73;
```

```
ip_mcast_data.dip.ipv6_addr[0] = 0xff;
ip_mcast_data.dip.ipv6_addr[1] = 0x09;
ip_mcast_data.dip.ipv6_addr[15] = 0x5;
ip_mcast_data.portmask.bits[0] = (1 << 1) | (1 << 3) | (1 << 5);

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ip6mcMode_set (unit, LOOKUP_ON_DIP_AND_SIP)

/* add multicast entry */
rtk_l2_ip6McastAddr_add(unit, &ip6_mcast_data);
```

To add an IPv6 multicast entry with VID+GIP for VID=100, DIP=ff09::5, forward portmask=port 2,4,6:

```
/* variable declaration */
uint32 unit = 0;
rtk_l2_ip6McastAddr_t ip6_mcast_data;

/* multicast entry configuration */
ip_mcast_data.rvid = 100;
ip_mcast_data.dip.ipv6_addr[0] = 0xff;
ip_mcast_data.dip.ipv6_addr[1] = 0x09;
ip_mcast_data.dip.ipv6_addr[15] = 0x5;
ip_mcast_data.portmask.bits[0] = (1 << 1) | (1 << 3) | (1 << 5);

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ip6mcMode_set (unit, LOOKUP_ON_DIP_AND_FVID)

/* add multicast entry */
rtk_l2_ip6McastAddr_add(unit, &ip_mcast_data);
```

To delete an IPv6 multicast entry with GIP+SIP for SIP=2001::73, DIP= ff09::5:

```
/* variable declaration */
uint32 unit = 0;
rtk_ipv6_addr_t sip;
rtk_ipv6_addr_t dip;

/* multicast entry configuration */
sip.ipv6_addr[0] = 0x20;
sip.ipv6_addr[1] = 0x01;
sip.ipv6_addr[15] = 0x73;
dip.ipv6_addr[0] = 0xff;
dip.ipv6_addr[1] = 0x09;
dip.ipv6_addr[15] = 0x5;

/* set ip multicast hash mode to GIP+SIP */
rtk_l2_ip6mcMode_set(uint32 unit, LOOKUP_ON_DIP_AND_SIP)

/* delete multicast entry */
rtk_l2_ip6McastAddr_del(unit, sip, dip, 0);
```

3.6.5 Web Authentication

```
/* variable declaration */
uint32 unit = 0;
rtk_port_t port ;
rtk_l2_newMacLrnMode_t lrnMode ;
rtk_action_t fwdAction;

/* variable configuration */
lrnMode = NOT_LEARNING;
fwdAction = ACTION_TRAP2CPU;

/* disable L2 learning and trap new SA/ port move packet, enable port move flush */
FOR_EACH_LOGIC_PORT(port)
{
    rtk_l2_newMacOp_set(unit, port, lrnMode, fwdAction);
    rtk_l2_legalPortMoveAction_set(unit, port, fwdAction);
    rtk_l2_legalPortMoveFlushAddrEnable_set(unit, port, ENABLED) ;
}

/* software add authorized L2 entry */
rtk_l2_addr_add(unit, &l2_uAddr);
```

3.6.6 Port Base mac constraint

Limit port 10 to learn at most 50 dynamic entries, and drop the further new SA packets:

```
/* variable declaration */
uint32 unit = 0;
rtk_port_t port = 9;
uint32 mac_cnt = 50;
rtk_l2_limitLearnCntAction_t action = LIMIT_LEARN_CNT_ACTION_DROP ;

/* add multicast entry */
rtk_l2_portLimitLearningCnt_set(unit, port, mac_cnt);
rtk_l2_portLimitLearningCntAction_set( unit, port, action);
```

3.6.7 VLAN base mac constraint

Limit VLAN 100 to learn at most 50 dynamic entries, and trap the further new SA packets:

```
/* variable declaration */
uint32 unit = 0;
uint32 fid_macLimit_idx = 0;
rtk_l2_fidMacLimitEntry_t fidMacLimitEntry ;
rtk_l2_limitLearnCntAction_t action;

/* variable configuration */
fidMacLimitEntry.fid = 100;
fidMacLimitEntry.maxNum = 50
action = ACTION_TRAP2CPU;

/* set VLAN based mac constraint entry and action */
rtk_l2_fidLimitLearningEntry_set(unit, fid_macLimit_idx, &fidMacLimitEntry);
rtk_l2_fidLearningCntAction_set(unit, action);
```

Realtek CONFIDENTIAL
CAMEO Communications, Inc.

3.6.8 SA Block

Enable SA block on all ports and add an SA block entry:

```
/* variable declaration */
uint32 unit = 0;
rtk_port_t port;

/* unicast entry configuration with SA block*/
memcpy(l2_uAddr.mac.octet, 0x001122334455, 6);
l2_uAddr.vid = 1;
l2_uAddr.port = 0x1f;
l2_uAddr.agg_vid = 0 ;
l2_uAddr.flags |= TRUE << RTK_L2_UCAST_FLAG_SA_BLOCK;

/* add unicast entry */
rtk_l2_addr_add(unit, &l2_uAddr);
```

3.6.9 DA Block

Enable DA block on all ports and add an DA block entry:

```
/* variable declaration */
uint32 unit = 0;
rtk_port_t port;

/* unicast entry configuration with DA block*/
memcpy(l2_uAddr.mac.octet, 0x001122334455, 6);
l2_uAddr.vid = 1;
l2_uAddr.port = 0x1f;
l2_uAddr.agg_vid = 0 ;
l2_uAddr.flags |= TRUE << RTK_L2_UCAST_FLAG_DA_BLOCK;

/* add unicast entry */
rtk_l2_addr_add(unit, &l2_uAddr);
```

3.6.10 SA Secure

Enable SA secure and add an SA secure entry:

```
/* variable declaration */
uint32 unit = 0;
rtk_port_t port;

/* unicast entry configuration with SA secure*/
memcpy(l2_uAddr.mac.octet, 0x001122334455, 6);
l2_uAddr.vid = 1;
l2_uAddr.port = 0x1f;
l2_uAddr.agg_vid = 0 ;
l2_uAddr.flags |= TRUE << RTK_L2_UCAST_FLAG_SA_BLOCK; /*SA secure and SA block share the same bit and the definition of the bit is decided by SA secure enable status*/

/* enable SA secure */
```

Realtek CONFIDENTIAL
CAMEO Communications

```
rtk_l2_secureMacMode_set(unit, ENABLED);

/* add unicast entry */
rtk_l2_addr_add(unit, &l2_uAddr);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

4 Access Control List

The device supports an ACL engine to process and classify packets at wire speed. ACL is used for operating and managing traffic for flow-based applications such as flow-based access control list (ACL), flow-based Quality of Service (QoS), flow-based rate limiting, flow-based mirroring, and so on.

ACL engine is mainly composed of compare engine and action engine. The ACL compare engine filters packet data down to bit level and the action engine executes the command for the qualified packet for specific applications.

4.1 ACL Overview

The device supports 1536 (1.5K) entries in total which are divided into 12 ACL blocks, therefore, each block contains 128 entries. Each block provides an independent lookup and power enable configurations.

Table 35: ACL_BLK_LOOKUP_CTRL Register

Field Name	Bits	Description
LUT_EN	1	Enable lookup for a specific ACL block. 1'b0: disable 1'b1: enable

Table 36: PS_ACL_PWR_CTRL Register

Field Name	Bits	Description
ACL_PWR_EN	1	Enable power for a specific ACL block. 1'b0: disable 1'b1: enable

The device provides 5 set of pre-defined and 3 set of user-defined (configurable) templates. The template is the compare key to filter packet. Each block can be matched for 3 set of templates and each entry has a fixed field to further select one of these three templates to map.

Table 37: ACL_BLK_TMPLTE_CTRL Register

Field Name	Bits	Description
BLK_TMPLATE1	3	Maps block to the first template.
BLK_TMPLATE2	3	Maps block to the second template.
BLK_TMPLATE3	3	Maps block to the third template.

API REFERENCE

```
rtk_acl_blockLookupEnable_get(uint32 unit, uint32 block_idx, rtk_enable_t *pEnable)
rtk_acl_blockLookupEnable_set(uint32 unit, uint32 block_idx, rtk_enable_t enable)

rtk_acl_blockPwrEnable_get(uint32 unit, uint32 block_idx, rtk_enable_t *pEnable)
rtk_acl_blockPwrEnable_set(uint32 unit, uint32 block_idx, rtk_enable_t enable)

rtk_acl_templateSelector_get(uint32 unit, uint32 block_idx, rtk_acl_templateIdx_t *pTemplate_idx)
rtk_acl_templateSelector_set(uint32 unit, uint32 block_idx, rtk_acl_templateIdx_t template_idx)
```

4.2 ACL Functional Block

Realtek CONFIDENTIAL CAMEO Communications, Inc.

The receiving packet is first processed by Compare Engine and outputs hit results. The results are then further processed by ACL entry operations and block operations. Once the final hit results are given, the Action Engine executes the actions for the hits entries after running action arbitration.

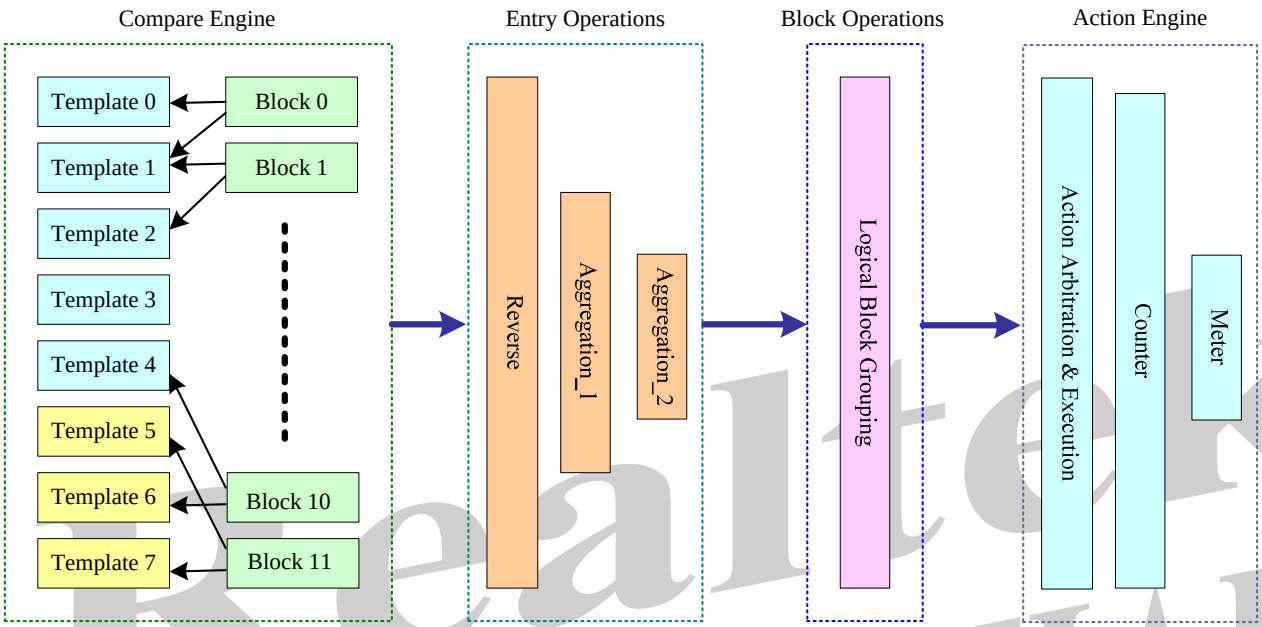


Figure 1: ACL Functional Block

4.3 ACL Template

An ACL entry is composed of Rule Data, Operations and Actions. The length of each entry's rule data (compare key) is 216–bits (192-bits template and 24-bits fixed field).

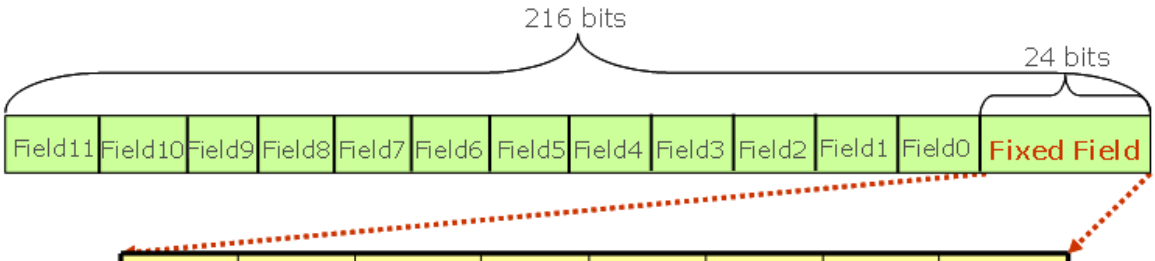
The 192-bits width template is composed of 12 16-bits width template fields. For user-defined templates, each field can be configured one template field type to filter packet characteristics. There are two kinds of field types: fixed and shared. We will take a look at these field types in following sections.

4.3.1 Fixed Field Type

The 192-bits rule data of the ACL entry is defined according to the mapped template. Each ACL entry also contains a 24-bits width fixed field which is independent of templates. It is used to qualify following characteristics:

- Template ID: select the mapping template
- Inner Tag or Inner Priority Tag Packet
- Outer Tag or Outer Priority Tag Packet
- Inner Untag or Inner Priority Tag Packet
- Outer Untag or Outer Priority Tag Packet
- Frame Type: ARP (exclude RARP), IPv4, IPv6, Other
- L2 Frame Type: Ethernet, LLC_SNAP or LLC_OTHER
- L4 Frame Type: UDP, TCP, ICMP/ICMPv6, IGMP or L4 unknown
- IP Non-Zero Offset: IPv4 or IPv6 packet offset value is non-zero.
- Switch MAC: DMAC of a packet is destined to one of the switch MAC address.
- Management VLAN: Packet's forwarding VID matches CPU port's P-VID. The forwarding VID is either from tagged packet or P-VID of untagged packet.
- SPN: Source port number
- SPMM01: The results of source port mask entry 0 and 1.

Realtek CONFIDENTIAL CAMEO Communications, Inc.



Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Outer Untag or Outer Pri-Tag	Inner Untag or Inner Pri-Tag	Outer Tag or Outer Pri-Tag	Inner Tag or Inner Pri-Tag	00: Ethernet 01: LLC_SNAP 10: LLC_Other 11: Reserved		**Template ID** 0b11 is reserved	
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
MGNT VLAN	**Switch MAC**	IP4/IP6 offset!=0	000: UDP 001: TCP 010: ICMP/ICMPv6 011: IGMP 1XX: L4 unknown			00: ARP 01: L2 ONLY 10: IPv4 11: IPv6	
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
Result of Source Portmask Entry 0,1		Source Logical Port					

Figure 2: ACL Entry Rule Data

API REFERENCE

Corresponding Field Name of Fixed Field Type:
USER_FIELD_TEMPLATE_ID
USER_FIELD_ITAG_EXIST
USER_FIELD_OTAG_EXIST
USER_FIELD_ITAG_FMT
USER_FIELD_OTAG_FMT
USER_FIELD_FRAME_TYPE
USER_FIELD_FRAME_TYPE_L2
USER_FIELD_L4_PROTO
USER_FIELD_IP_NONZEROOFFSET
USER_FIELD_SWITCHMAC
USER_FIELD_MGNT_VLAN
USER_FIELD_SPN
USER_FIELD_SPMM_0_1

*rtk_acl_ruleEntryField_read(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx, rtk_acl_fieldType_t type, uint8 *pData, uint8 *pMask)*
*rtk_acl_ruleEntryField_write(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx, rtk_acl_fieldType_t type, uint8 *pData, uint8 *pMask)*

```
rtk_acl_ruleEntryField_get(
    uint32                  unit,
    rtk_acl_phase_t         phase,
    rtk_acl_id_t            entry_idx,
    uint8                   *pEntry_buffer,
    rtk_acl_fieldType_t     type,
```

REALTEK CONFIDENTIAL

```
    uint8                   *pData,
    uint8                   *pMask)
rtk_acl_ruleEntryField_set(
    uint32                   unit,
    rtk_acl_phase_t          phase,
    rtk_acl_id_t             entry_idx,
    uint8                   *pEntry_buffer,
    rtk_acl_fieldType_t      type,
    uint8                   *pData,
    uint8                   *pMask)
```

4.3.2 Share Field Type

The share field types shown in below table are for Ingress ACL. The table is organized by four columns:
Template Field: describes the template field name.
Field Type: describes the field type name.
Description: describes the meaning of the field.
Template Field Position Limitation: defines the specific index (0~11) that the template field can be located for user-defined templates. If it is empty, the template field can appear in any index.

The table also shows the relationship between template field and field type. One filed type may consume several template fields while several field types may be located in the same template field.

Table 38: Share Field Type and Template Field Mapping Table

Field Type (USER_FIELD_XXX)	Template Field (TMPLTE_FIELD_XXX)	Description	Template Field Position Limitation
SPMM	SPMMASK	Comparison result of source port bitmap mask configurations	
SPM	SPM0	Source Port Mask [15:0]	0/4/8/10
	SPM1	Source Port Mask [28:16] {12:0}	1/5/9/11
DATYPE	SPM1	{unicast, multicast, broadcast} {14:13}	1/5/9/11
PPPoE		Packet with PPPoE header	
DMAC	DMAC0	DMAC[15:0]	0/3/6/9
	DMAC1	DMAC[31:16]	1/4/7/10
	DMAC2	DMAC[47:32]	2/5/8/11
SMAC	SMAC0	SMAC[15:0]	0/3/6/9
	SMAC1	SMAC[31:16]	1/4/7/10
	SMAC2	SMAC[47:32]	2/5/8/11
ETHERTYPE	ETHERTYPE	Type/Length	
OTAG_PRI	OTAG	Outer Tag priority/Port-based priority	
DEI_VALUE		DEI	
OTAG_VID		Outer Tag VID/Outer P-VID	
ITAG_PRI	ITAG	Inner Tag priority/Port-based priority	
CFI_VALUE		CFI	
ITAG_VID		Inner Tag VID/Inner P-VID	
IP4_SIP	SIP0	IPv4 source IP[15:0]	0/2/4/6/8/10
	SIP1	IPv4 source IP[31:16]	1/3/5/7/9/11
IP4_DIP	DIP0	IPv4 destination IP[15:0]	0/2/4/6/8/10

	DIP1	IPv4 destination IP[31:16]	1/3/5/7/9/11
IP6_SIP	SIP0	IPv6 SIP [15:0]	0/2/4/6/8/10
	SIP1	IPv6 SIP [31:16]	1/3/5/7/9/11
	SIP2	IPv6 SIP[47:32]	2/6
	SIP3	IPv6 SIP[63:48]	3/7
	SIP4	IPv6 SIP[79:64]	0/4/8
	SIP5	IPv6 SIP[95:80]	1/5/9
	SIP6	IPv6 SIP[111:96]	2/6/10
	SIP7	IPv6 SIP[127:112]	3/7/11
IP6_DIP	DIP0	IPv6 DIP [15:0]	0/2/4/6/8/10
	DIP1	IPv6 DIP [31:16]	1/3/5/7/9/11
	DIP2	IPv6 DIP[47:32]	2/6
	DIP3	IPv6 DIP[63:48]	3/7
	DIP4	IPv6 DIP[79:64]	0/4/8
	DIP5	IPv6 DIP[95:80]	1/5/9
	DIP6	IPv6 DIP[111:96]	2/6/10
	DIP7	IPv6 DIP[127:112]	3/7/11
IP4TOS_IP6TC	IP_TOS_PROTO	IPv4 TOS/IPv6 Traffic Class	
IP4PROTO_IP6NH		IPv4 Protocol/IPv6 Next Header	
ARPOPCODE	IP_TOS_PROTO	ARP/RARP OPCode	
L4_SRC_PORT	L4_SPORT	TCP/UDP Source Port	
ICMP_CODE	L4_SPORT	ICMP Code/ICMPv6 Code	
ICMP_TYPE		ICMP Type/ICMPv6 Type/IGMP Type	
IGMP_GROUPIP	L4_DPORT L4_SPORT ICMP_IGMP	IGMP Group IP[15:0] in L4_DPORT, IGMP Group IP[23:16] in L4_SPORT {7:0}, IGMP Group IP[27:24] in ICMP_IGMP{3:0},	
L4_DST_PORT	L4_DPORT	TCP/UDP Destination Port	
IP_FLAG	L34_HEADER	[1:0]: IPv4 header Flags(bit1 DF, bit0 MF)	
IP6_MOB_HDR_EXIST		IPv6 packet with mobility header	
IP6_AUTH_HDR_EXIST		IPv6 packet with authentication header	
IP6_DEST_HDR_EXIST		IPv6 packet with destination option header	
IP6_FRAG_HDR_EXIST		IPv6 packet with fragment header	
IP6_ROUTING_HDR_EXIST		IPv6 packet with routing header	
IP6_HOP_HDR_EXIST		IPv6 packet with hop-by-hop option header	
IP4_TTL_IP6_HOPLIMIT		IPv4 TTL/IPv6 Hop Limit 2b'00: TTL = 0 2b'01: TTL = 1 2b'10: 2<=TTL<255 2b'11: TTL = 255	
TCP_FLAG		TCP Flag	
ICMP_CODE	ICMP_IGMP	ICMP Code/ICMPv6 Code	
ICMP_TYPE		ICMP Type/ICMPv6 Type	
IGMP_TYPE	ICMP_IGMP	IGMP Type	
TCP_NONZEROSEQ	ICMP_IGMP	TCPNonZeroSequence	
TCP_ECN		TCP ECN ([1]: ECE, [0]: CWR)	

TCP_FLAG		TCP Flag ([5]: URG, [0]: FIN)	
TELNET		TCP & L4 destination port = 23	
SSH		TCP & L4 destination port = 22	
HTTP		TCP & L4 destination port = 80	
HTTPS		TCP & L4 destination port = 443	
SNMP		UDP & L4 destination port = 161/10161	
UNKNOWN_L7		Protocol is TCP/UDP & NOT TELNET/SSH/HTTP/HTTPS/SNMP	
VID_RANGE0	RANGE_CHK	O-VID/I-VID Range Check Mask for range check configuration 0-15	
PORT_RANGE		TCP/UDP Port Range Check Mask	
PKT_LEN_RANGE		Packet length Range Check Mask	
FIELD_SELECTOR_RANGE		Field Selector Range Check Mask	
IP_RANGE	IP_RANGE	IPv4/IPv6 Address Range Check Mask	
FLOW_LABEL	FLOW_LABEL IP_RANGE	Flow label LSB 16-bit in FLOW_LABEL, Flow label MSB 4-bit {13:10} in IP_RANGE	
FWD_VID	FWD_VID	Forwarding VID in VLAN tag or port-based VID {11:0}	
FIELD_SELECTOR_VALID_MSK	FIELD_SELECTOR_VALID	Field Selector Valid Mask	
L2_CRC_ERROR		L2 CRC Error	
ETAG_EXIST		Extra-Tagged	
FIELD_SELECTOR0	FIELD_SELECTOR_0	User defined 16-bit field selector 0	
FIELD_SELECTOR1	FIELD_SELECTOR_1	User defined 16-bit field selector 1	
FIELD_SELECTOR2	FIELD_SELECTOR_2	User defined 16-bit field selector 2	
FIELD_SELECTOR3	FIELD_SELECTOR_3	User defined 16-bit field selector 3	

Fixed field SPN is used to filter a single ingress port while SPM can be used to filter multiple ingress ports by configuring the mask bits.

For ARP/RARP packet, the DMAC field represents the destination hardware address and the DIP field represents the destination protocol address in ARP/RARP header.
For ARP/RARP packet, the SMAC field represents the source hardware address and the SIP field represents the source protocol address in ARP/RARP header.

OTAG_VID and ITAG_VID qualify VID field in tag for tagged packet or port-based vid for priority-tagged and untagged packet.
OTAG_PRI and ITAG_PRI qualify Priority field in tag for tagged and priority-tagged packet or port-based priority for untagged packet.

The value of ETHERTYPE can follow with CPU Tag、Outer Tag、Inner Tag or the first Type/Length value follows with RFC_1042 header(AA-AA-03-00-00-00)。

CONFIDENTIAL
CAMEO Communications, Inc.

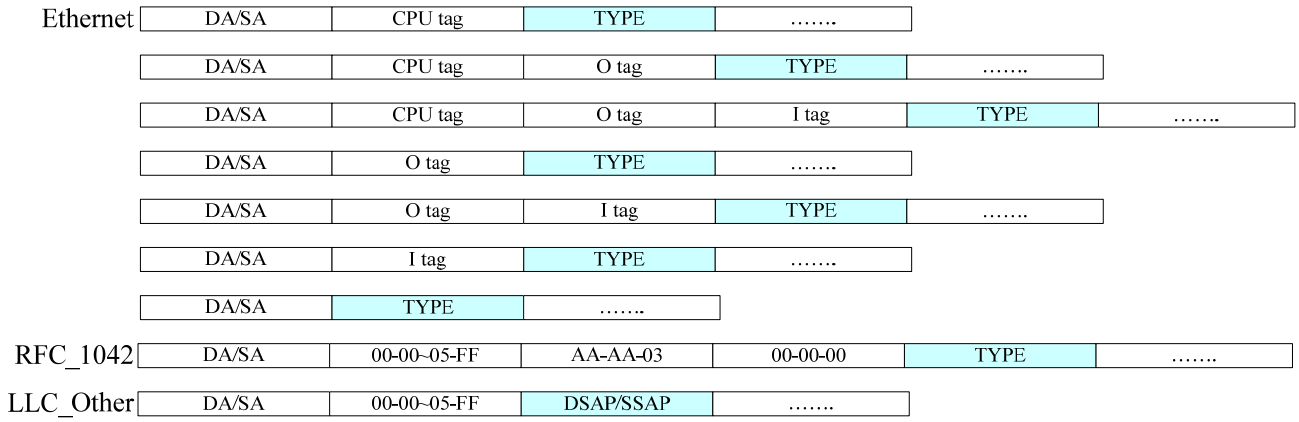


Figure 3: Ether Type Field Position

In default, L2 CRC error packets aren't processed by ACL. For qualifying L2_CRC_ERROR, there is a register field ACL_GLB_LOOKUP_CTRL.CRCERRPIELK to enable ACL for L2 CRC error packets.

The ICMP_IGMP represents the ICMP/ICMPv6 code, type for ICMP/ICMPv6 packet and represents IGMP type for IGMP packet and represents TCP ECN, TCP Flag, and management applications for TCP/UDP packet.

IP4PROTO_IP6NH is from Protocol field for an IPv4 packet and from the last known Next Header if there are multiple Next Headers appeared for an IPv6 packet.

API REFERENCE

```
rtk_acl_ruleEntryField_read(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx, rtk_acl_fieldType_t
type, uint8 *pData, uint8 *pMask)
rtk_acl_ruleEntryField_write(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx, rtk_acl_fieldType_t
type, uint8 *pData, uint8 *pMask)

rtk_acl_ruleEntryField_get(
    uint32                  unit,
    rtk_acl_phase_t         phase,
    rtk_acl_id_t            entry_idx,
    uint8                  *pEntry_buffer,
    rtk_acl_fieldType_t     type,
    uint8                  *pData,
    uint8                  *pMask)
rtk_acl_ruleEntryField_set(
    uint32                  unit,
    rtk_acl_phase_t         phase,
    rtk_acl_id_t            entry_idx,
    uint8                  *pEntry_buffer,
    rtk_acl_fieldType_t     type,
    uint8                  *pData,
    uint8                  *pMask)
```

4.3.3 Source Port Bitmap Mask

The device supports 16 source port bitmap configurations (RNG_CHK_SPM_CTRL) and each configuration contains a source port bitmap.
The source port of received packet is used to compare with 16 source port bitmap configurations simultaneously, and the 16 bits comparison result (each bit represents a comparison result of a source port bitmap configuration)

can be examined by filed type SPMM. The comparison result is true if the source port is in the configured bitmap.

Table 39: RNG_CHK_SPM_CTRL Register

Field Name	Bits	Description
SPM_0	29	Source port bitmap represents port 0 to 28

Source port bitmap configurations reduce the used template fields from two (SPM0-1) to one (SPMM) to filter multiple source ports.

API REFERENCE

*int32 rtk_acl_rangeCheckSrcPort_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_portMask_t *pData)*
*int32 rtk_acl_rangeCheckSrcPort_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_portMask_t *pData)*

4.3.4 Range Check

The device supports five kinds of range check function: VLAN, IP, L4 port, packet length and field selector to filter a packet within specific range.

The device supports 16 VID / L4 port / packet length / field selector range check entries. For priority-tag and untag packet, the device takes port-based VID to process VID range check. The packet length includes 4-bytes CRC field. If the content of field selector is invalid, the comparison result is mismatch. The 16-bits result of VID / L4 port / packet length / field selector range check is examined by the RANGE_CHK field types.

Table 40: RNG_CHK_CTRL Register

Field Name	Bits	Description
UPPER	16	range upper bound
LOWER	16	range lower bound
TYPE	3	Range check field type: 3'h0: I-VID range check 3'h1: O-VID range check 3'h2: layer4 source port range check 3'h3: layer4 destination port range check 3'h4: either layer4 source or layer4 destination port 3'h5: packet length (bytes,including CRC) 3'h6: field selector 0 3'h7: field selector 1

The device supports 8 IP range check entries which can be used as 8 IPv4 or 2 IPv6 or 4 IPv6 suffix range check. The 8-bits result of IP range check is examined by IP_RANGE field types.

Table 41: RNG_CHK_IP_CTRL Register

Field Name	Bits	Description
TYPE	3	3'h0: IPv4 SIP 3'h1: IPv4 DIP 3'h2: IPv6 SIP 3'h3: IPv6 DIP 3'h4: IPv6 SIP [63:0] suffix 3'h5: IPv6 DIP [63:0] suffix 3'h6~7: reserved

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

Table 42: RNG_CHK_IP_RNG Register

Field Name	Bits	Description
IP_UPPER	32	IP address upper bound
IP_LOWER	32	IP address lower bound

For IPv6 range check, entry 0/4 represents SIP/DIP [31:0], entry 1/5 represents SIP/DIP [63:32], entry 2/6 represents SIP/DIP [95:64], and entry 3/7 represents SIP/DIP [127:96]. The device compares each 32-bits individually for IPv6 range check.

For IPv6 suffix range check, entry 0/2/4/6 represents SIP/DIP [31:0] and entry 1/3/5/7 represents SIP/DIP [63:32]. When 2n and (2n+1) entry of IP range check are configured to the same IPv6 suffix type, the device compares 64-bits altogether and the result is reflected to entry 2n (result of entry 2n+1 is cleared to 0).
The IPv6 bit notation is defined as: *2001[127:112]:0db8:1f70:2633:0999:0de8:7648:06e8[15:0]*

Range check result is valid only if the packet format is complied with the range check data type.

API REFERENCE

```
int32 rtk_acl_rangeCheckVid_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_vid_t *pData)
int32 rtk_acl_rangeCheckVid_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_vid_t *pData)

int32 rtk_acl_rangeCheckIp_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_ip_t *pData)
int32 rtk_acl_rangeCheckIp_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_ip_t *pData)

int32 rtk_acl_rangeCheckL4Port_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_l4Port_t *pData)
int32 rtk_acl_rangeCheckL4Port_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_l4Port_t *pData)

int32 rtk_acl_rangeCheckPacketLen_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_packetLen_t
*pData)
int32 rtk_acl_rangeCheckPacketLen_set(uint32 unit, uint32 index, rtk_acl_rangeCheck_packetLen_t
*pData)

int32 rtk_acl_rangeCheckFieldSelector_get(uint32 unit, uint32 index, rtk_acl_rangeCheck_fieldSelector_t
*pData)
int32 rtk_acl_ rangeCheckFieldSelector _set(uint32 unit, uint32 index, rtk_acl_rangeCheck_ fieldSelector _t
*pData)
```

4.3.5 Field Selector

The device supports 4 set of 16-bits field selectors which are used as user defined fields.
There are 4 kinds of start positions to support L2~L4 applications. Each field selector entry is composed of start position and offset.

Table 43: PARSER_FIELD_SELTOR_CTRL Register

Field Name	Bits	Description
OFFSET	8	Offset in bytes.
FMT	3	Define the start address for offset. 3'h0: Raw packet (start at SFD, begin with DA) 3'h1: L2 packet (start after Ether type(EthernetII /Length (RFC_1042 /SNAP_OTHER)) 3'h2: L3 packet (start at IPv4/IPv6 header or ARP/RARP header) 3'h3: L4 packet (start at TCP/UDP/ICMP/IGMP header)

Field selector doesn't support to locate the VLAN Tag.

CAMEO Communications, Inc.

API REFERENCE

```
int32 rtk_acl_fieldSelector_get(uint32 unit, uint32 fs_idx, rtk_acl_fieldSelector_data_t *pFs)
int32 rtk_acl_fieldSelector_set(uint32 unit, uint32 fs_idx, rtk_acl_fieldSelector_data_t *pFs)
```

4.3.6 Pre-defined Template

The device supports 5 pre-defined templates. They are assigned with template ID 0-4 while 5-7 are for user defined templates. The pre-defined templates are shown as below:

Table 44: Pre-defined Template 0

Field 0	Field 1	Field 2	Field 3	Field 4	Field 5
SPM0	SPM1	OTAG	SMAC0	SMAC1	SMAC2
Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
DMAC0	DMAC1	DMAC2	ETHER TYPE	ITAG	RANGECHK

This template is for L2 ACL and VLAN translation applications.

Table 45: Pre-defined Template 1

Field 0	Field 1	Field 2	Field 3	Field 4	Field 5
SIP0	SIP1	DIP0	DIP1	IPTOSPROTO	L4SPORT
Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
L4DPORT	ICMPIGMP	ITAG	RANGECHK	SPM0	SPM1

This template is for L3/L4 ACL applications.

Table 46: Pre-defined Template 2

Field 0	Field 1	Field 2	Field 3	Field 4	Field 5
DMAC0	DMAC1	DMAC2	ITAG	ETHER TYPE	IPTOSPROTO
Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
L4DPORT	L4SPORT	SIP0	SIP1	DIP0	DIP1

This template is for STP/LACP/LLDP/GVRP/PPPoE/DHCP/QoS applications.

Table 47: Pre-defined Template 3

Field 0	Field 1	Field 2	Field 3	Field 4	Field 5
DIP0	DIP1	DIP2	DIP3	DIP4	DIP5
Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
DIP6	DIP7	L4DPORT	L4SPORT	ICMPIGMP	IPTOSPROTO

This template is for L3&L4 IPv6 ACL applications.

Table 48: Pre-defined Template 4

Field 0	Field 1	Field 2	Field 3	Field 4	Field 5
SIP0	SIP1	SIP2	SIP3	SIP4	SIP5
Field 6	Field 7	Field 8	Field 9	Field 10	Field 11
SIP6	SIP7	ITAG	RANGECHK	SPM0	SPM1

This template is for IPV4/IPV6 IP-MAC-VLAN-Port binding/ARP Spoofing/Voice VLAN applications.

API REFERENCE

```
rtk_acl_template_get(uint32 unit, uint32 template_id, rtk_acl_template_t *pTemplate)
rtk_acl_template_set(uint32 unit, uint32 template_id, rtk_acl_template_t *pTemplate)
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

4.4 ACL Operation

Each ACL entry supports a REVERSE operation which is used to reverse the compare result.

To support a compare key larger than length of a single template (192-bits), the device supports AGGREGATION operations. The length of compare key can be double length (384-bits) by enabling AGGREGATION_1 or AGGREGATION_2 operation and quadruple length (768 bits) by enabling both AGGREGATION_1 and AGGREGATION_2 operations.

The AGGREGATION_1 operation is used to aggregate the result of two consecutive (0 & 1, 2 & 3, …, 2n & 2n+1) entries in the same block as one. The device executes the logical AND operation for these two aggregated entries and updates the result back to the entry with lower index and clear the result of the entry with higher index to 0. Each index 2N entry supports an AGGREGATION_1 operation and it makes sense if the two aggregated entries maps to different templates.

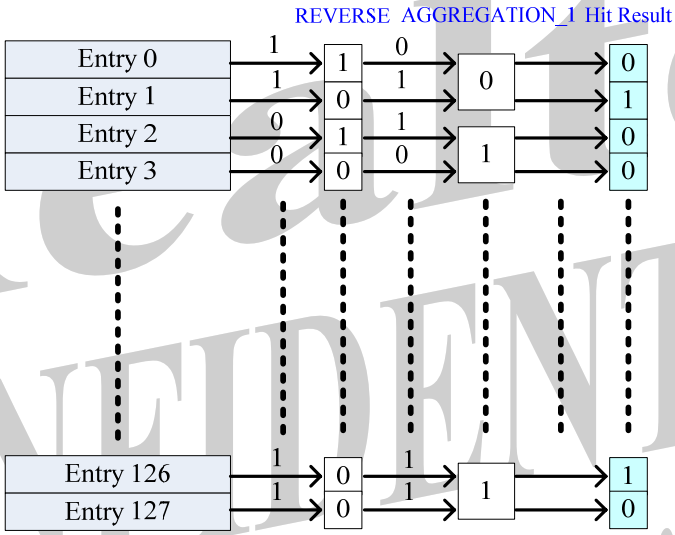


Figure 4: REVERSE and AGGREGATION_1 Operation Flow

The AGGREGATION_2 operation is used to aggregate the result of two entries in the two consecutive blocks as one. The device executes the logical AND operation for these two aggregated entries and updates the result back to the entry in the lower block and clear the result of the entry in the higher block to 0. Each index 2N+256M (where N,M = 0,1,2...) entry supports an AGGREGATION_2 operation and it makes sense if the two aggregated entries maps to different templates. It should pay attention to that only the first 6 blocks(entry 0~767) support AGGREGATION_2 operation.

When AND1 and AND2 operations are both enabled, the length of compare key is up to 768 bits.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

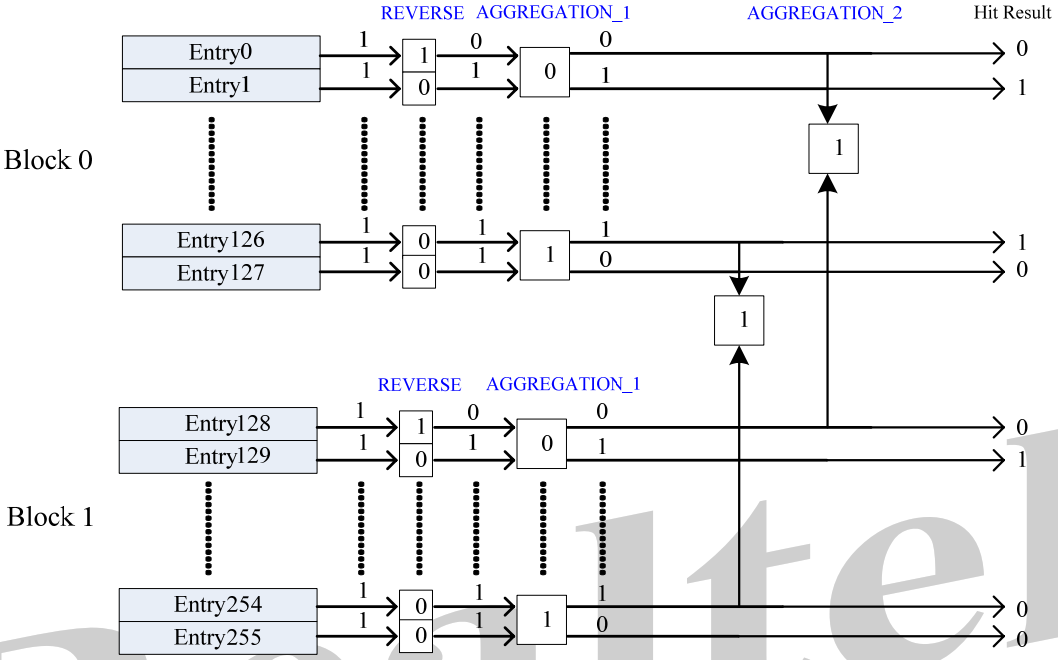


Figure 5: REVERSE, AGGREGATION_1 and AGGREGATION_2 Operation Flow

API REFERENCE

```
int32 rtk_acl_ruleOperation_get(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_operation_t *pOperation)
int32 rtk_acl_ruleOperation_set(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_operation_t *pOperation)
```

4.5 ACL Action

Each ACL entry has an action mask which is used to indicate the actions to be executed when the entry is hit. Ingress ACL supports up to 5 kind of actions which can be executed concurrently.

4.5.1 Ingress ACL Action

Ingress ACL action mask:

- Drop
- Forward
- Ingress Outer VLAN Assignment
- Ingress Inner VLAN Assignment
- Filter
- Statistic
- Remarking
- Meter
- Egress Tag Status
- Mirror
- Normal Priority Assignment
- CPU Priority Assignment
- Outer TPID Assignment
- Inner TPID Assignment

4.5.1.1 Drop Action

Drop action provides below configurations:

Realtek CONFIDENTIAL CAMEO Communications, Inc.

- Nop
- Drop
- Withdraw drop
- Reserved

The withdraw drop action can withdraw other ACL entries' Drop action setting (independent of drop in forward action below). The Drop action will be decided by all selected entries, not by the entry with lowest index.

4.5.1.2 Forward Action

Ingress Forward action provides below configurations:

- Permit
- Drop
- Copy packet to single port
- Copy packet to multiple ports
- Redirect packet to single port
- Redirect packet to multiple ports
- Unicast Routing
- VLAN Leaky

The Permit action can be used to avoid a specific flow from hitting other Forward actions. For instance, if traffic comes from subnet 192.168.1.X would be dropped except from host 192.168.1.1, a Permit entry should be inserted to the index lower than the Drop entry to grant the traffic from 192.168.1.1. The Permit action only takes effect within the ACL module.

The Copy action adds one or more egress ports to the original egress port list. For Copy to CPU applications, such as ARP, DHCP, just specify the CPU Port ID as the copied port. The packet copied to CPU has higher priority than traffic isolation, egress VLAN filtering, egress Spanning Tree filtering, lookup miss action and lookup miss flooding domains.

The Redirect action overrides the original egress port list. For Trap to CPU applications, just specify the CPU Port ID as the redirected port. The packet trapped to CPU has higher priority than traffic isolation, egress VLAN filtering, egress Spanning Tree filtering, lookup miss action and lookup miss flooding domains.

The VLAN leaky action make the packet ignore Ingress/Egress VLAN check.

Regarding the unicast routing, please refer to the Static Routing Developer Guide.

4.5.1.3 Ingress Inner VLAN Assignment Action

Ingress Inner VLAN assignment action provides below configurations:

- Assign inner VLAN
- Shift inner VLAN
- Assign inner VLAN by shifting outer VLAN
- Assign port-based inner VLAN

Assign inner VLAN action directly assigns an inner VLAN to the qualified packet.

Shift inner VLAN action assigns an inner VLAN to the qualified packet by shifting packet's inner VID with a specific value. The action is useful for the ingress VLAN translation application that shares the same translation offset. If VLAN 1~100 come from downlink port would be translated to VLAN 1001~1100 and forward to uplink port, only one ACL entry is needed. If assigned VLAN (the VLAN after shifting) is larger than 4095, then it is wrapped around. For instance, Packet's VID=4000 and shift value=100, then the assigned VLAN will be 4 (4000+100-4096).
Shift inner VLAN action only takes effect if the receiving packet is an inner tag packet.

Assign inner VLAN by shifting outer VLAN action is similar to Shift inner VLAN action except it assigns an inner VLAN to the qualified packet by shifting packet's outer VID with a specific value.
Assign inner VLAN by shifting outer VLAN action only takes effect if the receiving packet is an outer tag packet.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

4.5.1.4 Ingress Outer VLAN Assignment Action

Ingress Outer VLAN assignment action provides below configurations:
- Assign outer VLAN
- Shift outer VLAN
- Assign outer VLAN by shifting inner VLAN
- Assign port-based outer VLAN

The behavior of Ingress Outer VLAN assignment action is similar to Ingress Inner VLAN assignment action.

4.5.1.5 Filter Action

Statistic action provides two configurations:
- Filter a single port
- Filter multiple ports

The Filter action excludes the specified single port or multiple ports from the original egress port list, and it has higher priority than the Forward action.

4.5.1.6 Statistic Action

Statistic action provides two configurations:
- 32-bits packet-based counter
- 64-bits byte-based counter

The Statistic action specifies the counter type and an index to the Counter. The device supports 256 counters and each counter can be used as one 64-bits byte-based counter or two 32-bits packet-based counters which is specified by Statistic action. Thus, up to 512 32-bits packet-based or 256 64-bits byte-based counters can be supported.

If counter type is 32-bits packet-based counter, the counter is increased with the number of packets. If counter type is 64-bits byte-based counter, the counter is increased with the packet length.

4.5.1.7 Remarking Action

Remarking action provides below configurations:
- Remark inner tag
- Remark outer tag
- Remark DSCP
- Remark IP Precedence
- Remark inner tag by copying outer tag
- Remark outer tag by copying inner tag
- Keep priority of inner tag
- Keep priority of outer tag

Remark DSCP action takes effect only if receiving packet is an IPv4/IPv6 packet. For IPv6 packet, DSCP is resided in Traffic Class field.

Remark IP Precedence action takes effect only if receiving packet is an IPv4 packet.

Remark inner tag by copying outer tag action remarks inner tag by copying priority from outer tag, thus, the action takes effect only if receiving packet is an outer tag packet.

Remark outer tag by copying inner tag action remarks outer tag by copying priority from inner tag, thus, the action takes effect only if receiving packet is an inner tag packet.

Remark inner tag by copying outer tag action and Remark outer tag by copying inner tag action may be used in VLAN translation or Q-in-Q applications.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

Keep priority of inner tag action keeps the original priority of inner tag, thus, the action takes effect only if receiving packet is an inner tag packet.

Keep priority of outer tag action keeps the original priority of outer tag, thus, the action takes effect only if receiving packet is an outer tag packet.

4.5.1.8 Meter Action

The Meter action specifies the meter index for retrieving meter information.
The device supports 64 meters. The packet gets drop according to the meter configuration if traffic rate is over than the meter rate.

Each entry is composed of ENABLE, EXCEED_FLAG, TYPE, THRESHOLD_GROUP and RATE fields. TYPE decides the entry based on byte mode or packet mode. THRESHOLD_GROUP decides which threshold group works. The thresholds are global configuration by BYTE_LB_THR0 / BYTE_LB_THR1 for byte based mode and PKT__LB_THR0 / PKT_LB_THR1 for packet based mode. RATE is the rate of leaky bucket. The unit is decided by TYPE which could be byte based (16K bit per second as granularity) or packet based (1 packet per second as granularity). The effective bit length of rates is 16 bits for byte based mode and effective length of rates is 18 bits for packet based mode.

INCL_PREIFG register field decides if the rate detecting of byte based mode includes or excludes preamble and IFG. User can configure this register for demand.

Table 49: Meter Entry Fields

Field Name	Bits	Description
ENABLE	1	Enable this entry or not. Should set this bit each time changing the parameters of this entry.
EXCEED_FLAG	1	A latched flag indicating whether the meter entry ever exceeded the threshold or not. Write 1 to clear. 0b0: no packet is dropped by this meter 0b1: some packets are ever dropped by this meter
TYPE	1	Mode of leaky bucket calculation unit of this entry. 1b'0: based on byte mode 1b'1: based on packet mode
THRESHOLD_GROUP	1	Select threshold group. 1b'0: select group 0 1b'1: select group 1
RATE	18	A 18-bit wide rate setting in units of pps, or 16-bit rate setting in units of 16Kbps.

Table 50: METER_GLB_CTRL Register

Register Name	Field Name	Bits	Description
METER_GLB_CTRL	INCL_PREIFG	1	Packet length including preamble and IFG or not 0b0: exclude 0b1: include

Table 51: METER_BYTE_LB_THR_CTRL Register

Register Name	Field Name	Bits	Description
METER_BYTE_LB_THR_CTRL	BYTE_LB_THR1	16	Threshold of leaky bucket in unit of 128 bytes(Max 8M bytes). This takes effect when meter is byte mode.
METER_BYTE_LB_THR_CTRL	BYTE_LB_THR0	16	Threshold of leaky bucket in unit of 128 bytes(Max 8M bytes). This takes effect when meter is byte mode.

Table 52: METER_PKT_LB_THR_CTRL Register

Register Name	Field Name	Bits	Description
METER_PKT_LB_	PKT_LB_THR1	16	Threshold of leaky bucket in unit of 1 packet. This takes

Realtek CONFIDENTIAL CAMEO Communications, Inc.

THR_CTRL		effect when meter is packet mode.
METER_PKT_LB_ THR_CTRL	PKT_LB_THR0 16	Threshold of leaky bucket in unit of 1 packet. This takes effect when meter is packet mode.

4.5.1.9 Egress Tag Status Action

Egress tag status action specifies the inner and outer tag status for an outgoing packet. Four statuses can be specified: untag, tag, keep content, NOP. Keep content keeps tag format and tag content, that is, untag in then untag out, priority tag in then priority tag out, tag in then tag out, and the content (VID/Priority/CFI) is all kept. NOP represents NULL operation which doesn't specify the tag status. NOP is used if user just wants to specify only inner or outer tag status.

If egress inner/outer tag status is specified to be keep content, it overrides the egress inner/outer VLAN assignment action.

4.5.1.10 Mirror Action

Mirror action provides below configurations:

- Original
- Modified

The Mirror action specifies mirror entry index for retrieving the mirroring port. Other configurations of the indexed mirror entry are not utilized.
The action can decide to mirror original packet or modified packet.

4.5.1.11 Normal Priority Assignment Action

Normal Priority action assigns an internal priority to ingress packet. The assigned priority is one of the internal priority sources. The final internal priority which for mapping egress queue is made by Priority Decision module. Regarding Priority Decision module, please refer to Priority Decision Developer Guide. It applies to all ports, including CPU port.

4.5.1.12 CPU Priority Assignment Action

CPU Priority action assigns an internal priority to ingress packet for CPU port only. It has higher priority than Normal Priority action.

4.5.1.13 Inner TPID Assignment Action

Inner TPID Assignment action specifies inner TPID index for retrieving new inner TPID. The TPID of inner tag will be replaced by new inner TPID if out with inner tag.

4.5.1.14 Outer TPID Assignment Action

The behavior of Outer TPID assignment action is similar to Inner TPID assignment action.

API REFERENCE

```
int32 rtk_acl_ruleAction_get(uint32 unit, rtk_acl_phase_t    phase, rtk_acl_id_t entry_idx,
      rtk_acl_action_t *pAction)
int32 rtk_acl_ruleAction_set(uint32 unit, rtk_acl_phase_t    phase, rtk_acl_id_t entry_idx,
            rtk_acl_action_t *pAction)

rtk_acl_meterMode_get(uint32 unit, uint32 idx, rtk_acl_meterMode_t *pMeterMode)
rtk_acl_meterMode_set(uint32 unit, uint32 idx, rtk_acl_meterMode_t meterMode)

rtk_acl_meterIncludeIfg_get(uint32 unit, rtk_enable_t *pIfg_include)
rtk_acl_meterIncludeIfg_set(uint32 unit, rtk_enable_t ifg_include)
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_acl_meterEntry_get(uint32 unit, uint32 meterIdx, rtk_acl_meterEntry_t *pMeterEntry)
rtk_acl_meterEntry_set(uint32 unit, uint32 meterIdx, rtk_acl_meterEntry_t *pMeterEntry)

rtk_acl_meterBurstSize_get(uint32 unit, rtk_acl_meterMode_t meterMode, rtk_acl_meterBurstSize_t
*pBurstSize)
rtk_acl_meterBurstSize_set(uint32 unit, rtk_acl_meterMode_t meterMode, rtk_acl_meterBurstSize_t
*pBurstSize)

rtk_acl_meterExceed_get(uint32 unit, uint32 meterIdx, uint32 *pIsExceed);
```

4.6 Block Operation

There is a block operations supported by the device, grouping the physical ACL blocks as a logical block.

4.6.1 Logical Block Grouping

For some specific applications, 128 entries of a block may not be enough. The device supports block grouping function ACL_BLK_GROUP_CTRL for grouping physical blocks to a logical block. Each block except the last block has a bit in ACL_BLK_GROUP_CTRL register to determine group with next block or not.

A logical block only outputs a single hit entry which is the lowest index entry among hit entries.

Table 53: ACL_BLK_GROUP_CTRL Register

Field Name	Bits	Description
ENABLE	1	Enable the combination with next block to form a larger logical block. 1'b0: disable 1'b1: enable

API REFERENCE

```
int32 rtk_acl_blockGroupEnable_get(uint32 unit, uint32 block_idx, rtk_acl_blockGroup_t group_type,
    rtk_enable_t *pEnable)
int32 rtk_acl_blockGroupEnable_set(uint32 unit, uint32 block_idx, rtk_acl_blockGroup_t group_type,
    rtk_enable_t enable)
```

4.7 Action Arbitration and Execution

A packet may hit multiple ACL entries concurrently, thus, action arbitration for the hit entries is made before action execution. For the hit entries, the device per action group picks up the lowest index entries among blocks to execute the actions.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

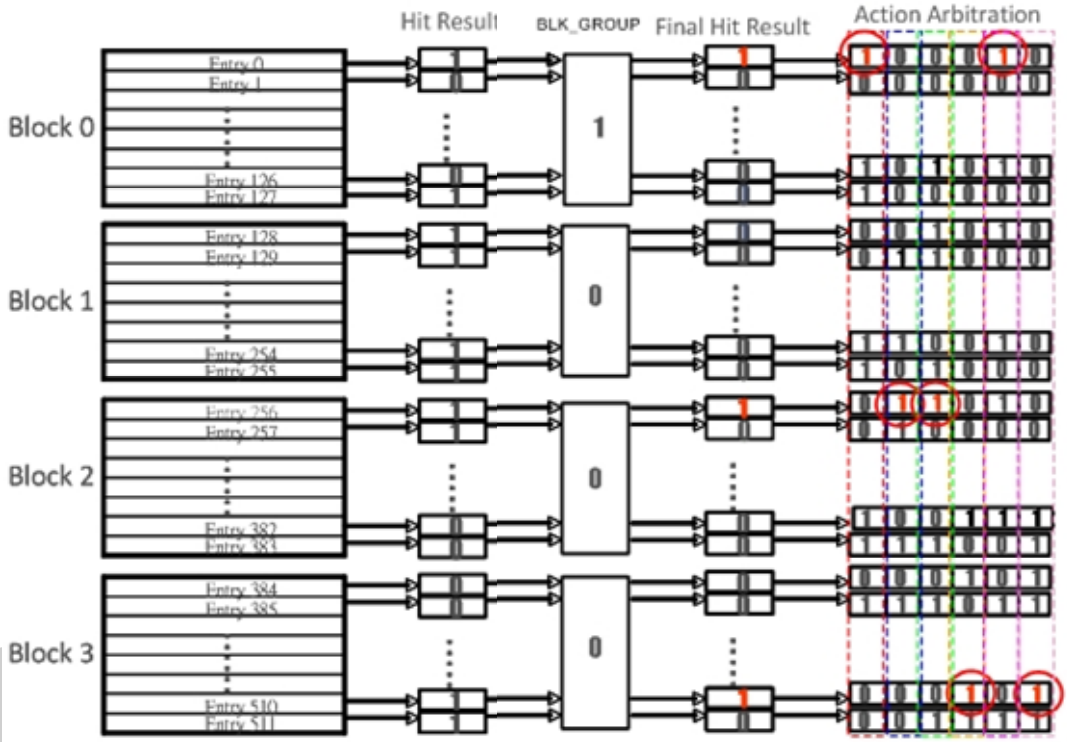


Figure 6: Action Arbitration and Execution

4.8 Clearance and Movement

The device supports the functionality to clear ACL entries to reduce CPU loading. Whole ACL entry includes ACL rule data, operations and actions are all cleared. The counters and meters indexed by the ACL entries will not be cleared.

Table 54: ACL_CLR_CTRL Register

Field Name	Bits	Description
CLR	1	Clear the ACL entries. Reset to 0 after clearing is completed.
CLR_TO	12	Clear to the index
CLR_FROM	12	Clear from the index

API REFERENCE

*int32 rtk_acl_rule_del(uint32 unit, rtk_acl_phase_t phase, rtk_acl_cleart_t *pClrIdx)*

ACL entry index represents priority because entry with lower index is hit first. Therefore, it is common to move entries to adjust the priority. The device supports the functionality to move ACL entries to reduce CPU loading. Whole ACL entry includes ACL rule data, operations and actions are all moved. The counters and meters indexed by the ACL entries will not be moved.

Table 55: ACL_MV_CTRL Register

Field Name	Bits	Description
MV	1	Move the ACL entries. Reset to 0 after moving is completed.
MV_TO	12	Move to the index

CONFIDENTIAL
CAMEO Communications, Inc.

MV_FROM	12	Move from the index

Table 56: ACL_MV_LEN_CTRL Register

Field Name	Bits	Description
MV_LEN	12	The number of entries to move

API REFERENCE

```
int32 rtk_acl_rule_move(uint32 unit, rtk_acl_phase_t phase, rtk_acl_move_t *pData)
```

4.9 Action Priority

- When the packet is hit statistic action, statistic action is always executed no matter the packet is drop/forward/trap by other modules.
- When the packet is hit mirror action, mirror action is executed in most cases except the packet is dropped by Ingress bandwidth control, Max frame size, Egress Port/Q congest modules.
- When the packet is hit both forward and meter actions, the meter action has higher priority.

4.10 Programming Example

Example 1: Copy ARP request to CPU and assign priority to ARP reply packet

Filter Condition:
ARP request is ARP packet and DMAC = broadcast MAC address.
ARP reply is ARP packet and DMAC = switch MAC address.

```
rtk_acl_templateIdx_t     template_info;
rtk_acl_igrAction_t       action ;

/* enable power and lookup function */
block_idx = 0;
rtk_acl_blockLookupEnable_set(unit, block_idx, ENABLED);
rtk_acl_blockPwrEnable_set(unit, block_idx, ENABLED);

/* use pre-defined template 0 */
template_info. template_id[0] = 0;
rtk_acl_templateSelector_set(unit, block_idx, template_info)

/** set entry 0 for ARP request packet **/
entry_id = 0

/* map entry to the first template */
data =0;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure frame type is ARP */
data = 0x0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);
```

Realtek CONFIDENTIAL

```
/* configure broadcast DMAC */
data = mask = 0xFFFFFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_IGR_ACTION_FWD_COPY_TO_PORTID;
action.igr_acl.fwd_data.info.copy_redirect_port_fwd_port_id = CPU_PORT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

/* validate the entry */
rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);

/** set entry 1 for ARP reply packet **/
entry_id = 0

/* configure block template index */
data =0;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure frame type is ARP */
data = 0x0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);

/* configure DMAC is switch MAC */
data = mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SWITCHMAC, data,
mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.pri_en = ENABLED;
action.egr_acl.pri_data.pri = ARP_REQ_PRIO ;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

/* validate the entry */
rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

Example 2: IPv4 control filter and MAC control filter

MAC Filter condition: SMAC, DMAC, Ether Type, outer VID, and ingress port.
For ingress port 2, grant packet with DMAC = 11:22:33:44:55:66 && ether type = 0x0800 && outer VID = 1, and drop packet with DMAC = 11:22:33:44:55:XX.

Entry 1: DMAC = 11:22:33:44:55:66, ether type = 0x0800, OVID = 1, ingress port = 2.
Entry 2: DMAC = 11:22:33:44:55:XX, ingress port = 2.

IPv4 Filter condition: IPv4, SIP, DIP, IP protocol, L4 source/destination port, ICMP type/code, IGMP type, L4 src/dest port range, TOS/DSCP, and ingress port.

For ingress port 3 and 4, allow packet with
1. SIP = 10.1.1.0 through 10.1.1.255 && protocol = TCP && L4 src port less than 100

2. DIP = 20.1.1.1 && protocol = ICMP && ICMP type = 1 && ICMP code = 0
3. SIP = 10.1.1.1 && protocol = UDP && L4 dest port = 300 && DSCP = 40
4. DIP = 224.x.x.x && protocol = IGMP && IGMP type = 1

Otherwise, drop the IPv4 packet.

```
#define ACL_BLOCK_MAX_ENTRY 128

rtk_acl_templateIdx_t                templateIdx_info;
rtk_acl_template_t                   template_info
rtk_acl_rangeCheck_portMask_t        port_range;
rtk_acl_rangeCheck_l4Port_t          l4port_range;

/** block configuration **/
/* the filter entry will be configured in block 1 */
block_idx = 1;
rtk_acl_blockLookupEnable_set(unit, block_idx, ENABLED);
rtk_acl_blockPwrEnable_set(unit, block_idx, ENABLED);

/* use fixed template id 0 and configurable template id 5 */
/* template id 0 is for MAC control filter */
/* configure template id 5 for IP control filter */
mac_block_template_idx = 0;
ip_block_template_idx = 1;
mac_template_id = 0;
ip_template_id = 5;

template_info.field[0] = TMPLTE_FIELD_SIP0;
template_info.field[1] = TMPLTE_FIELD_SIP1;
template_info.field[2] = TMPLTE_FIELD_DIP0;
template_info.field[3] = TMPLTE_FIELD_DIP1;
template_info.field[4] = TMPLTE_FIELD_IP_TOS_PROTO;
template_info.field[5] = TMPLTE_FIELD_L4_SPORT;
template_info.field[6] = TMPLTE_FIELD_L4_DPORT;
template_info.field[7] = TMPLTE_FIELD_ICMP_IGMP;
template_info.field[8] = TMPLTE_FIELD_RANGE_CHK;
template_info.field[9] = TMPLTE_FIELD_SPMMASK;

rtk_acl_template_set(unit, ip_template_id, &template_info);

templateIdx_info. template_id[mac_block_template_idx] = mac_template_id;
templateIdx_info. template_id[ip_block_template_idx] = ip_template_id;
rtk_acl_templateSelector_set(unit, block_idx, templateIdx_info)


/** MAC control entry 1 **/
/* ACL configuration */
mac_entry1_prio = 0;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + mac_entry1_prio;

/* configure entry block template index */
data = mac_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure DMAC */
data = 0x112233445566;
mask = 0xFFFFFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);
```

```
/* configure ether type */
data = 0x0800;
mask = 0xffff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_ETHERTYPE, data, mask);

/* configure outer VID */
data = 0x1;
mask = 0xfff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_OTAG_VID, data, mask);

/* configure ingress port */
data = 0x2;
mask = 0x3f;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPN, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);



/** MAC control entry 2 **/
/* ACL configuration */
mac_entry1_prio = 1;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + mac_entry1_prio;

/* configure entry block template index */
data = mac_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data, mask);

/* configure DMAC */
data = 0x112233445566;
mask = 0xFFFFFFFFFF00;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure ingress port */
data = 0x2;
mask = 0x3f;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPN, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_DROP;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

```
/** IP control entry 1 **/
/* ACL configuration */
ip_entry1_prio = 2;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ip_entry1_prio;

/* configure entry block template index */
data = ip_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv4 frame type */
data = 0x2
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);

/* configure SIP */
data = 0xA010100;
mask = 0xFFFFFF00;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4_SIP, data, mask);

/* configure protocol */
data = 0x6;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);

/* configure L4 port range */
l4port_range_id = 0;

l4port_range.l4port_dir = RNGCHK_L4PORT_DIRECTION_SRC;
l4port_range. lower_bound = 0;
l4port_range. upper_bound = 99;
rtk_acl_rangeCheckL4Port_set(unit, l4port_range_id, &l4port_range);

data = mask = (1 << l4port_range_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_PORT_RANGE, data,
mask);

/* configure ingress ports */
range_srcPort_id = 2;

LOGIC_PORTMASK_SET_PORT(port_range.portmask, 3) ;
LOGIC_PORTMASK_SET_PORT(port_range.portmask, 4) ;
rtk_acl_rangeCheckSrcPort_set(unit, range_srcPort_id, &port_range);

data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

```
/** IP control entry 2 **/
/* ACL configuration */
ip_entry2_prio = 3;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ip_entry2_prio;

/* configure entry block template index */
data = ip_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv4 frame type */
data = 0x2
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);

/* configure DIP */
data = 0x14010101;
mask = 0xFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4_DIP, data, mask);

/* configure protocol */
data = 0x1;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);

/* configure ICMP type */
icmp_type = 1;

data = icmp_type;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_ICMP_TYPE, data,
mask);

/* configure ICMP code */
icmp_code = 0;

data = icmp_code;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_ICMP_CODE, data,
mask);

/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

```
/** IP control entry 3 **/
/* ACL configuration */
ip_entry3_prio = 4;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ip_entry3_prio;

/* configure entry block template index */
data = ip_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv4 frame type */
data = 0x2
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);

/* configure SIP */
data = 0xA010101;
mask = 0xFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4_SIP, data, mask);

/* configure protocol */
data = 0x11;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);

/* configure L4 dest port */
l4dest_port = 300 ;
data = l4dest_port;
mask = 0xffff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_L4_DST_PORT, data,
mask);

/* configure DSCP */
data = 40;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP_DSCP, data, mask);

/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);


/** IP control entry 4 **/
/* ACL configuration */
ip_entry4_prio = 5;
```

```
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ip_entry4_prio;

/* configure entry block template index */
data = ip_block_template_idx;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv4 frame type */
data = 0x2
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);

/* configure DIP */
data = 0xE0000000;
mask = 0xFF000000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4_DIP, data, mask);

/* configure protocol */
data = 0x2;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);

/* configure IGMP type */
igmp_type = 1;

data = igmp_type;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IGMP_TYPE, data,
mask);

/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);



/** drop other IPv4 packet **/
/* ACL configuration */
ip_entry5_prio = 6;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ip_entry5_prio;

/* configure IPv4 frame type */
data = 0x2
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);
```

```
/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_DROP;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

Example 3: IPv6 control filter

IPv6 Filter condition: IPv6, SIPv6, DIPv6, next header, L4 source/destination port, ICMPv6 type/code, L4 src/dest port range, TOS/DSCP, and ingress port.

For ingress port 3 and 4, allow packet with
1. SIPv6 = EA03:0102::1 through EA03:0102::FFFF && DIPv6 = ED03:00A0::1234:3 && protocol = TCP && L4 src port great than 100
2. DIPv6 = link local && protocol = ICMPv6 && ICMPv6 type = 1 && ICMPv6 code = 0
3. SIPv6 = EA03:0102::1 && protocol = UDP && L4 dest port = 300 && DSCP = 40

Use Aggregation_1 operation to filter full SIPv6 and DIPv6.

```
#define ACL_BLOCK_MAX_ENTRY 128

rtk_acl_templateIdx_t              templateIdx_info;
rtk_acl_template_t                 template_info;
rtk_acl_operation_t                oper;
rtk_acl_rangeCheck_portMask_t      port_range;
rtk_acl_rangeCheck_l4Port_t        l4port_range;

/** block configuration **/
/* the filter entry will be configured in block 2 */
block_idx = 2;
rtk_acl_blockLookupEnable_set(unit, block_idx, ENABLED);
rtk_acl_blockPwrEnable_set(unit, block_idx, ENABLED);

/* use fixed template id 3 and configurable template id 6 */
/* DIPv6, L4 src/dest port, ICMP type/code, next header, traffic class in fixed templated 3 */
/* SIPv6, L4 src/dest port range, ingress port mask in configurable template 6 */

/* configure template id 6 */
template_id = 6;

template_info.field[0] = TMPLTE_FIELD_SIP0;
template_info.field[1] = TMPLTE_FIELD_SIP1;
template_info.field[2] = TMPLTE_FIELD_SIP2;
template_info.field[3] = TMPLTE_FIELD_SIP3;
template_info.field[4] = TMPLTE_FIELD_SIP4;
template_info.field[5] = TMPLTE_FIELD_SIP5;
template_info.field[6] = TMPLTE_FIELD_SIP6;
template_info.field[7] = TMPLTE_FIELD_SIP7;
template_info.field[8] = TMPLTE_FIELD_RANGE_CHK;
```

Realtek CONFIDENTIAL

```
template_info.field[9] = TMPLTE_FIELD_SPMMASK;

rtk_acl_template_set(unit, template_id, &template_info);

templateIdx_info. template_id[0] = 3;
templateIdx_info. template_id[1] = template_id;
rtk_acl_templateSelector_set(unit, block_idx, templateIdx_info)


/** IPv6 control entry 1 **/
/* ACL configuration */
ip6_entry1_prio = 0;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + (ip6_entry1_prio * 2);

/* configure entry block template index */
data = 0x0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv6 frame type */
data = 0x3
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_FRAME_TYPE,
data, mask);

/* configure DIPv6 */
data = 0xED0300A00000000000000000012340003;
mask = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP6_DIP, data, mask);

/* configure protocol */
data = 0x6;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);


/* configure entry block template index */
data = 0x1;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_TEMPLATE_ID,
data, mask);

/* configure SIPv6 */
data = 0xEA030102000000000000000000000000;
mask = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_IP6_SIP, data,
mask);

/* configure L4 port range */
l4port_range_id = 1;

l4port_range.l4port_dir = RNGCHK_L4PORT_DIRECTION_SRC;
l4port_range. lower_bound = 101;
l4port_range. upper_bound = 0xFFFF;
rtk_acl_rangeCheckL4Port_set(unit, l4port_range_id, &l4port_range);
```

```
data = mask = (1 << l4port_range_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_PORT_RANGE,
data, mask);

/* configure ingress ports */
range_srcPort_id = 2;

LOGIC_PORTMASK_SET_PORT(port_range.portmask, 3) ;
LOGIC_PORTMASK_SET_PORT(port_range.portmask, 4) ;
rtk_acl_rangeCheckSrcPort_set(unit, range_srcPort_id, &port_range);

data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_SPMM, data,
mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

/* configure AND1 operation */
oper.aggr_2 = ENABLED;
rtk_acl_ruleOperation_set(unit, ACL_PHASE_IGR_ACL, entry_id, &oper);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, (entryl_id + 1), ENABLED);



/** IPv6 control entry 2 **/
/* ACL configuration */
ip6_entry2_prio = 1;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + (ip6_entry2_prio * 2);

/* configure entry block template index */
data = 0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv6 frame type */
data = 0x3
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
mask);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_FRAME_TYPE,
data, mask);

/* configure DIPv6 */
data = 0xFE800000000000000000000000000000;
mask = 0xFFFF0000000000000000000000000000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP6_DIP, data, mask);

/* configure protocol */
data = 0x1;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
```

```
data, mask);

/* configure ICMP type */
icmp_type = 1;

data = icmp_type;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_ICMP_TYPE, data,
mask);

/* configure ICMP code */
icmp_code = 0;

data = icmp_code;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_ICMP_CODE, data,
mask);

/* configure entry block template index */
data = 1;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_SPMM, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

/* configure AND1 operation */
oper.aggr_2 = ENABLED;
rtk_acl_ruleOperation_set(unit, ACL_PHASE_IGR_ACL, (entry_id / 2), &oper);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, (entryl_id + 1), ENABLED);


/** IP control entry 3 **/
/* ACL configuration */
ip6_entry3_prio = 2;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + (ip6_entry3_prio * 2);

/* configure block template index */
data =0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure IPv6 frame type */
data = 0x3;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FRAME_TYPE, data,
```

```
mask);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_FRAME_TYPE,
data, mask);

/* configure protocol */
data = 0x11;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4PROTO_IP6NH,
data, mask);

/* configure DSCP=40 */
data = 40;
mask = 0xff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_IP4TOS_IP6TC, data,
mask);

/* configure L4 dest port */
l4dest_port = 300 ;
data = l4dest_port;
mask = 0xffff;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_L4_DST_PORT, data,
mask);

/* configure block template index */
data =1;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_TEMPLATE_ID,
data, mask);

/* configure SIP */
data = 0xEA0301020000000000000000000000;
mask = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_IP6_SIP, data,
mask);

/* configure ingress ports */
range_srcPort_id = 2;
data = mask = (1 << range_srcPort_id);
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, (entry_id + 1), USER_FIELD_SPMM, data,
mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
action.igr_acl.fwd_en = ENABLED;
action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_PERMIT;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

/* configure AND1 operation */
oper.aggr_2 = ENABLED;
rtk_acl_ruleOperation_set(unit, ACL_PHASE_IGR_ACL, (entry_id / 2), &oper);


rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, (entryl_id + 1), ENABLED);
```

Example 4: User define control filter

- Start 12 byte of packet is 0x1122
- Start 5 byte of L3 packet is 0x3344

```
#define ACL_BLOCK_MAX_ENTRY 128

rtk_acl_templateIdx_t                templateIdx_info;
rtk_acl_template_t                   template_info;
rtk_acl_fieldSelector_data_t         fs;

/** block configuration **/
/* the filter entry will be configured in block 3 */
block_idx = 3;
rtk_acl_blockLookupEnable_set(unit, block_idx, ENABLED);
rtk_acl_blockPwrEnable_set(unit, block_idx, ENABLED);

/* use configurable template id 7 */
template_id = 7;

template_info.field[0] = TMPLTE_FIELD_FIELD_SELECTOR_0;
template_info.field[1] = TMPLTE_FIELD_FIELD_SELECTOR_1;
template_info.field[2] = TMPLTE_FIELD_FIELD_SELECTOR_2;
template_info.field[3] = TMPLTE_FIELD_FIELD_SELECTOR_3;

rtk_acl_template_set(unit, template_id, &template_info);

templateIdx_info. template_id[0] = template_id;
rtk_acl_templateSelector_set(unit, block_idx, templateIdx_info)


/** user define control entry 1 **/
/* ACL configuration */
ud_entry1_prio = 0;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + ud_entry1_prio;

/* configure entry block template index */
data = 0x0;
mask = 0x3;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure offset info */
fs.start = FS_START_POS_RAW;
fs.offset = 12;
rtk_acl_fieldSelector_set(unit, 0, &fs);

/* configure offset filter */
data = 0x1122;
mask = 0xFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FIELD_SELECTOR0,
data, mask);


/* configure offset info */
fs.start = FS_START_POS_L3;
fs.offset = 5;
rtk_acl_fieldSelector_set(unit, 1, &fs);
```

```
/* configure offset filter */
data = 0x3344;
mask = 0xFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_FIELD_SELECTOR1,
data, mask);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

5 Unicast Routing

The device supports IPv4 and IPv6 unicast routing by ACL. Network route and host route are supported by qualifying network address and host address respectively. Longest prefix match routing can also be archived when the ACL entries are arranged properly.

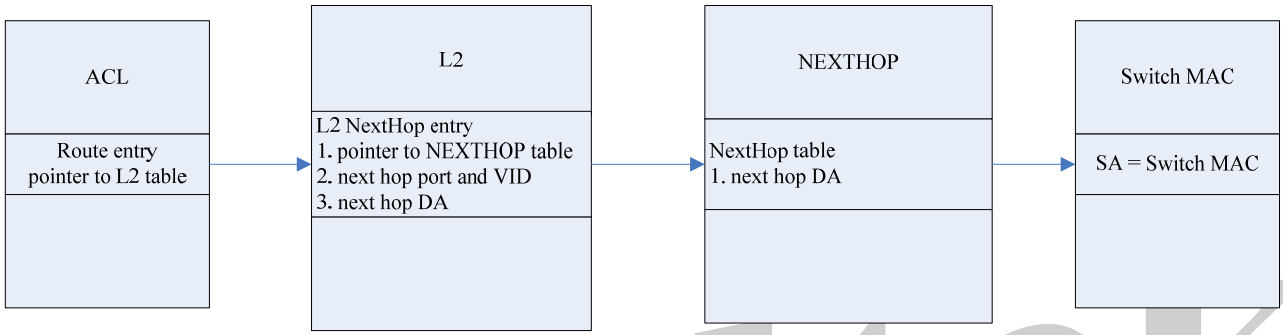


Figure 13: Routing Block Diagram

5.1 ACL Classification

Configure ACL rule data to match specific host/network address for host/network route. For the longest prefix match, the longer prefix entry should be inserted to lower ACL entry index for higher priority. In the situation, the default route entry is in the highest ACL entry index and the host route entry is in the lowest ACL entry index.

In addition to specify the rule data, configures ACL action to be routing and L2_NEXT_HOP index for pointing to L2 Next Hop entry in L2 table. If the pointed L2 entry is not a L2 Next Hop entry, the packet is dropped.

API REFERENCE

```
rtk_acl_ruleEntryField_read((uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_fieldType_t type, uint8 *pData, uint8 *pMask);
rtk_acl_ruleEntryField_write((uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_fieldType_t type, uint8 *pData, uint8 *pMask);

rtk_acl_ruleAction_get(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_action_t *pAction);
rtk_acl_ruleAction_set(uint32 unit, rtk_acl_phase_t phase, rtk_acl_id_t entry_idx,
    rtk_acl_action_t *pAction);
```

5.2 L2 Next Hop Entry

Type	Table Content (width = 75bits)											
Unicast	IP4 (1)	IP6 (1)	MAC (48)	FID_MSB (1)	SLP (5)	Age (2)	SABLK (1)	DABLK (1)	Static (1)	Suspend (1)	Next Hop (1)	NTO1_VID (12)

Type	Table Content (width = 75bits)												
NextHop	IP4 (1)	IP6 (1)	MAC (48)	FID_MSB (1)	DPN (5)	Age (2)	SABLK (1)	DABLK (1)	Static (1)	Suspend (1)	Next Hop (1)	VID_SEL (1)	NEXT_HOP_IDX (11)

Figure 2: L2 Unicast and NextHop Entry Format

The L2 next hop entry is a normal L2 unicast entry with next hop bit set by CPU. It is usually auto-learnt when receiving ARP packet from next hop and CPU should then set the next hop bit and NEXT_HOP_IDX before it is pointed by ACL entry.

Next hop port (DPN), VID (reverse calculated from L2 hash key) and NEXT_HOP_IDX are retrieved from L2 next hop entry for routing a packet. For the next hop VID, L2 next hop entry supports a bit VID_SEL to specify replacing inner or outer VLAN. And the NEXT_HOP_IDX is then used to lookup the NEXTHOP table for retrieving the next hop DMAC .

When port moving is occurred to a L2 next hop entry, the device updates the port information as usual. The L2 next hop entry can also be aged out, ROUTING_AGE action is triggered if a routing packet hits an aged out L2 next hop entry.

API REFERENCE

```
rtk_l2_addr_get(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr)
rtk_l2_addr_set(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr)
rtk_l2_addr_add(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr)
```

5.3 Next Hop Table

The device supports a 512 entries next hop table. Each entry contains a next hop DMAC address.

Table 57: NextHop Table

Field Name	Bits	Description
GATEWAY_MAC	48	Next hop DMAC address

API REFERENCE

```
rtk_l3_routeEntry_get(uint32 unit, uint32 index, rtk_l3_routeEntry_t *pEntry)
rtk_l3_routeEntry_set(uint32 unit, uint32 index, rtk_l3_routeEntry_t *pEntry)
```

5.4 Switch MAC Address

Per System provide only one switch MAC which is used to replace the source MAC of the routing packet.

Table 58: Switch MAC Register

Field Name	Bits	Description
MAC_ADDR_CTRL	48	Switch MAC address

API REFERENCE

```
rtk_switch_mgmtMacAddr_get(uint32 unit, rtk_mac_t *pMac);
rtk_switch_mgmtMacAddr_set(uint32 unit, rtk_mac_t *pMac);
```

5.5 Routing Exception

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Routing exception is asserted only if the packet is going to be routed. The supported routing exceptions are listed below:

- IPv4 TTL Exceed
  - Routing packet with TTL <= 1
- IPv4 Header with Option
- IPv6 Hop Limit Exceed
  - Routing packet with Hop Limit <= 1
- IPv6 packet with Hop-by-Hop header
- Gateway MAC Error
  - For IP routing, packet's DMAC should destine to the switch MAC address. If a packet is going to be routed but its DMAC address is not destined to one of the 16 switch MAC addresses, the packet will not be routed and the exception is asserted.
- Next Hop Aging Out

Table 59: ROUTING_EXCPT_CTRL Register

Field Name	Bits	Description
ROUTING_AGE	2	Action to take if L2 next hop entry is aged out 2'b00: drop 2'b01: forward 2'b10: trap 2'b11: reserved
GW_MAC_ERR	2	Action to take if packet's DMAC address is not destined to switch MAC address 2'b00: drop 2'b01: downgrade to L2 forwarding 2'b10: trap 2'b11: reserved
IP6_HOPBY_HOP	2	Action to take if IPv6 packet with hop-by-hop header 2'b00: drop 2'b01: forward 2'b10: trap 2'b11: reserved
IP6_HL_EXCEED	2	Action to take if hop limit reaches zero after doing routing. 2'b00: drop 2'b01: forward 2'b10: trap 2'b11: reserved
IP4_OPT	2	Action to take if IPv4 packet with option. 2'b00: drop 2'b01: forward 2'b10: trap 2'b11: reserved
IP4_TTL_EXCEED	2	Action to take if TTL reaches zero after doing routing. 2'b00: drop 2'b01: forward 2'b10: trap 2'b11: reserved

API REFERENCE

Corresponding Routing Exception Type:
ROUTE_EXCEPTION_TYPE_TTL_EXCEED
ROUTE_EXCEPTION_TYPE_WITH_OPT
ROUTE_EXCEPTION_TYPE_IP6_HL_EXCEED
ROUTE_EXCEPTION_TYPE_IP6_HOP_BY_HOP
ROUTE_EXCEPTION_TYPE_GW_MAC_ERR
ROUTE_EXCEPTION_TYPE_ENTRY_AGE_OUT

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_trap_routeExceptionAction_get(uint32 unit, rtk_trap_routeExceptionType_t type,
rtk_action_t *pAction)
rtk_trap_routeExceptionAction_set(uint32 unit, rtk_trap_routeExceptionType_t type,
rtk_action_t action)
```

5.6 Programming Example

Example:

ACL entry priority: host route > network route > default route

L3 entry 1: DIP = 10.1.1.2/32, VID = 2, MAC = 00:12:34:56:78:91, port = 3
L3 entry 2: DIP = 10.1.2.0/24, VID = 3, MAC = 00:12:34:56:78:92, port = 4
L3 entry 3: DIP = 10.1.0.0/16, VID = 4, MAC = 00:12:34:56:78:93, port = 5
L3 entry 4: DIP = 0.0.0.0/0, VID = 5, MAC = 00:12:34:56:78:94, port = 6

VLAN 1: VID = 2, MAC = 00:11:22:33:44:51
VLAN 2: VID = 3, MAC = 00:11:22:33:44:52
VLAN 3: VID = 4, MAC = 00:11:22:33:44:53
VLAN 4: VID = 5, MAC = 00:11:22:33:44:54

```
#define ACL_BLOCK_MAX_ENTRY 128

rtk_l3_routeEntry_t    route_entry;
rtk_l2_ucastAddr_t    l2_entry;

/** Next Hop entry configuration **/
/* L3 entry1 next hop */
route_entry_idx = 0;
route_entry.hostMac[0] = 0x00;
route_entry.hostMac[0] = 0x12;
route_entry.hostMac[0] = 0x34;
route_entry.hostMac[0] = 0x56;
route_entry.hostMac[0] = 0x78;
route_entry.hostMac[0] = 0x91;
rtk_l3_routeEntry_set(unit, route_entry_idx, &route_entry);

osal_memcpy(&l2_entry.mac, &route_entry.hostMac, ETHER_ADDR_LEN);
l2_entry.vid = 2;
l2_entry.flags |= RTK_L2_UCAST_FLAG_NEXTHOP;
l2_entry.route_idx = route_entry_idx;
rtk_l2_addr_add(unit, &l2_entry);
l2_nexthop_entry1 = l2_entry.l2_idx;

/* L3 entry2 next hop */
route_entry_idx = 1;
route_entry.hostMac[0] = 0x00;
route_entry.hostMac[0] = 0x12;
route_entry.hostMac[0] = 0x34;
route_entry.hostMac[0] = 0x56;
route_entry.hostMac[0] = 0x78;
route_entry.hostMac[0] = 0x92;
rtk_l3_routeEntry_set(unit, route_entry_idx, &route_entry);

osal_memcpy(&l2_entry.mac, &route_entry.hostMac, ETHER_ADDR_LEN);
```

```
l2_entry.vid = 3;
l2_entry.flags |= RTK_L2_UCAST_FLAG_NEXTHOP;
l2_entry.route_idx = route_entry_idx;
rtk_l2_addr_add(unit, &l2_entry);
l2_nexthop_entry2 = l2_entry.l2_idx;

/* L3 entry3 next hop */
route_entry_idx = 2;
route_entry.hostMac[0] = 0x00;
route_entry.hostMac[0] = 0x12;
route_entry.hostMac[0] = 0x34;
route_entry.hostMac[0] = 0x56;
route_entry.hostMac[0] = 0x78;
route_entry.hostMac[0] = 0x93;
rtk_l3_routeEntry_set(unit, route_entry_idx, &route_entry);

osal_memcpy(&l2_entry.mac, &route_entry.hostMac, ETHER_ADDR_LEN);
l2_entry.vid = 4;
l2_entry.flags |= RTK_L2_UCAST_FLAG_NEXTHOP;
l2_entry.route_idx = route_entry_idx;
rtk_l2_addr_add(unit, &l2_entry);
l2_nexthop_entry3 = l2_entry.l2_idx;

/* L3 entry4 next hop */
route_entry_idx = 3;
route_entry.swMac_idx = v3_switchMAC_idx;
route_entry.hostMac[0] = 0x00;
route_entry.hostMac[0] = 0x12;
route_entry.hostMac[0] = 0x34;
route_entry.hostMac[0] = 0x56;
route_entry.hostMac[0] = 0x78;
route_entry.hostMac[0] = 0x94;
rtk_l3_routeEntry_set(unit, route_entry_idx, &route_entry);

osal_memcpy(&l2_entry.mac, &route_entry.hostMac, ETHER_ADDR_LEN);
l2_entry.vid = 5;
l2_entry.flags |= RTK_L2_UCAST_FLAG_NEXTHOP;
l2_entry.route_idx = route_entry_idx;
rtk_l2_addr_add(unit, &l2_entry);
l2_nexthop_entry4 = l2_entry.l2_idx;

/** ACL configuration **/
/* block configuration */
block_idx = 0;
rtk_acl_blockLookupEnable_set(unit, block_idx, ENABLED);
rtk_acl_blockPwrEnable_set(unit, block_idx, ENABLED);

/* use fixed template id 1 */
route_block_template_idx = 1;

templateIdx_info. template_id[mac_block_template_idx] = route_template_id;
templateIdx_info. template_id[ip_block_template_idx] = route_template_id;
rtk_acl_templateSelector_set(unit, block_idx, templateIdx_info)

/** L3 entry 1 **/
entry1_prio = 0;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + entry1_prio;

/* configure entry block template index */
```

```
data = route_block_template_idx;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure DIP */
data = 0x0A010102;
mask = 0xFFFFFFFF;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
acl_action.igr_acl.fwd_en = ENABLED;
acl_action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_UNICAST_ROUTING;
acl_action.igr_acl.fwd_data.fwd_info = l2_nexthop_entry1;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);

/** L3 entry 2 **/
entry2_prio = 1;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + entry2_prio;

/* configure entry block template index */
data = route_block_template_idx;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure DIP */
data = 0x0A010200;
mask = 0xFFFFFF00;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
acl_action.igr_acl.fwd_en = ENABLED;
acl_action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_UNICAST_ROUTING;
acl_action.igr_acl.fwd_data.fwd_info = l2_nexthop_entry2;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);

/** L3 entry 3 **/
entry3_prio = 2;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + entry3_prio;

/* configure entry block template index */
data = route_block_template_idx;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure DIP */
data = 0x0A010000;
mask = 0xFFFF0000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure action */
```

```
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
acl_action.igr_acl.fwd_en = ENABLED;
acl_action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_UNICAST_ROUTING;
acl_action.igr_acl.fwd_data.fwd_info = l2_nexthop_entry3;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);

/** L3 entry 4 **/
entry4_prio = 3;
entry_id = (block_idx * ACL_BLOCK_MAX_ENTRY) + entry4_prio;

/* configure entry block template index */
data = route_block_template_idx;
mask = 0x1;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_TEMPLATE_ID, data,
mask);

/* configure DIP */
data = 0x00000000;
mask = 0x00000000;
rtk_acl_ruleEntryField_write(unit, ACL_PHASE_IGR_ACL, entry_id, USER_FIELD_DMAC, data, mask);

/* configure action */
osal_memset(&action, 0, sizeof(rtk_acl_igrAction_t));
acl_action.igr_acl.fwd_en = ENABLED;
acl_action.igr_acl.fwd_data.fwd_type = ACL_ACTION_FWD_UNICAST_ROUTING;
acl_action.igr_acl.fwd_data.fwd_info = l2_nexthop_entry4;
rtk_acl_ruleAction_set(unit, ACL_PHASE_IGR_ACL , entry_id, &action);

rtk_acl_ruleValidate_set(unit, ACL_PHASE_IGR_ACL, entryl_id, ENABLED);
```

CONFIDENTIAL
CAMEO Communications ,Inc.

6 Spanning Tree

Spanning tree is a Layer 2 link management protocol that provides path redundancy while preventing loops in the network. For a Layer 2 network to function properly, only one active path can exist between any two stations. The system provides a number of hardware features to support IEEE 802.1D STP, IEEE 802.1W RSTP, and IEEE 802.1s MSTP which includes per port per instance spanning port state configuration, the VLAN ID to instance mapping in 4K VLAN table.

6.1 Spanning Tree State Configuration

The system provides 64 instances of spanning tree port state configuration. Each instance contains port states of port 0 to port 28 (STATE_PORT0 to STATE_PORT28), and the state value could be disabled(0), blocking/listening(1), learning(2), forwarding(3) which can map to STP/ RSTP/ MSTP states as following table.

STP State Mapping	RSTP State Mapping	MSTP State Mapping	STATE_PORT Configuration
Disabled	Disabled	Disabled	Disabled(0)
Blocking/Listening	Discard	Discard	Blocking/Listening(1)
Learning	Learning	Learning	Learning(2)
Forwarding	Forwarding	Forwarding	Forwarding(3)

The port states have the following behaviors:

Port State	Rx Packet	Rx (Trap/Forward) BPDUs	Tx Packet (without ignore-stp)	Tx Packet (with ignore-stp) or Tx CPU tagged BPDU with AS_DPM=1	Learn Address	Forward Frame
Disabled	No	No	No	No	No	No
Blocking/Listening	No	Yes	No	Yes	No	No
Learning	No	Yes	No	Yes	Yes	No
Forwarding	Yes	Yes	Yes	Yes	Yes	Yes

Note: 'ignore-stp' is a signal from CPU tag (from CPU_TX_TAG.BP_FLTR_2 bit). Detail please refers to CPU tag developer guide.

00b – Disabled:

- All packets (including RMA/BPDUs) must be dropped on ingress/egress port.
- Do not learn the SA address.
- Do not transmit any packet on the egress port even with Ignore-STP.

01b – Blocking/Listening

- Only RMA/BPDU and RLDP/RLPP packets may/could be sent to the CPU, the other packets are dropped.
- Do not learn the SA address.
- Do NOT transmit the packet out without Ignore-STP.
- If RMA configuration specifies the particular packet to ignore the STP port state, then the packet can pass the STP ingress filter. Detail please refers to RMA developer guide. If RLDP/RLPP set trap to CPU, it can always bypass ingress STP filter.

10b – Learning

- Only RMA/BPDU and RLDP/RLPP packets may/could be sent to the CPU, the other packets are dropped after Learning Process.
- Do NOT transmit the packet out without Ignore-STP.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

· If RMA configuration specifies the particular packet to ignore the STP port state, then the packet can pass the STP ingress filter. Detail please refers to RMA developer guide. If RLDP/RLPP set trap to CPU, it can always bypass ingress STP filter.

11b – Forwarding
· All packets can be forwarded, and the SA address would be learnt.

Table 60: MSTI Table

Field Name	Bits	Description
STATE_PORT0 - STATE_PORT28	1	STP Port State. 2'b00: Disabled State 2'b01: Blocking/Listening State 2'b10: Learning State 2'b11: Forwarding State

API REFERENCE

*rtk_stp_mstpState_get(uint32 unit, uint32 msti, rtk_port_t port, rtk_stp_state_t *pStp_state);*
rtk_stp_mstpState_set(uint32 unit, uint32 msti, rtk_port_t port, rtk_stp_state_t stp_state);

6.2 Spanning Tree Instance

As 6.1 describes, the system supports 64 instances of spanning port states, and the instances could be mapped by VLAN to support MSTP per port per VLAN configuration. The field FID_MSTI of VLAN entry in 4k VLAN table decides the mapped instance ID (detail of configuring FID_MSTI please refer to VLAN developer guide). For example, if FID_MSTI of VLAN 100 is configured value 1, the spanning port states of instance 1 would take effect to affect the packets learning, receiving, and forwarding behavior of VLAN 100.

For the VLAN blink protocol STP and RSTP, only instance 0 (CIST) is needed to configured and the port state should take effect to all VLANs. To reduce the cost when the system switching between VLAN aware spanning tree and VLAN blink spanning tree, global register field MSTI_MODE is provided to decide the mapped instance ID of VLANs. If MSTI_MODE is normal mode (value=0), the mapped instance ID would follow FID_MSTI value which suits the behavior of MSTP; if MSTI_MODE is force 0 mode (value=1), the mapped instance ID of all VLANs would be forced 0 which suits the behavior of STP and RSTP. By configuring MSTI_MODE, software needs not modifying FID_MSTI of 4k VLAN table when user changes spanning tree mode from MSTP to STP or RSTP.

Table 61: VLAN_STP_CTRL Register

Field Name	Bits	Description
MSTI_MODE	1	Indicate instance ID is either from FID_MSTI of VLAN table or force using 0 (CIST). 0b0: normal mode (from VLAN table) 0b1: force mode (force using 0) Note: The configuration only affects MSTI but not FID.

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_stp_mstpInstance_create(uint32 unit, uint32 msti);
rtk_stp_mstpInstance_destroy(uint32 unit, uint32 msti);
rtk_stp_isMstpInstanceExist_get(uint32 unit, uint32 msti, uint32 *pMsti_exist);
rtk_stp_mstpInstanceMode_get(uint32 unit, rtk_stp_mstiMode_t *pMsti_mode);
rtk_stp_mstpInstanceMode_set(uint32 unit, rtk_stp_mstiMode_t msti_mode);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

7 Traffic Isolation

Typical applications for traffic isolation might be in a Co-location Facility or IaaS network (Infrastructure as a Service Cloud), where you might have several customers using the same subnet, but communications between the customers is not desirable as it would circumvent their firewalls. Another common use for traffic isolation might be in a hotel situation, where each hotel room has internet access, all are on the same subnet, but communications between the rooms is not desired.

Traffic isolation separates all the traffic (unicast, multicast, and broadcast) in layer 2 to provide traffic interference and bandwidth utilization. The system provides two type isolation modules, Port-based Isolation and VLAN-based Isolation. The details are as below.

7.1 Port-based Isolation

When hosts are connected to router through switch ports, they can utilize Port Isolation to force the communication between hosts through the router. Port isolation is a mechanism to provide a CPU configurable destination port mask for an ingress port. For each ingress port, the device provides a port isolation port mask configuration defined in register PORT_ISO_CTRL as Table 62: PORT_ISO_CTRL RegisterTable 62. Bit value 1 of the port mask means the port can communicate with each other. Bit value 0 is that the port can't forward any packet. No matter the received packets are known/unknown Unicast, known/unknown Multicast, Broadcast packets. All of the packets are limited by the configuration of Port Isolation port mask in forwarding.

Table 62: PORT_ISO_CTRL Register

Field Name	Bits	Description
P_ISO_MBR_0	29	Port mask (port 28 ~ port 0) that specify the ports to be communicated.

For example as Figure 1, hosts A and B are connected to ports 0 and 3 separately while router is connected to port 7. The traffic between A and B must be forwarded by router. Therefore, user can configure the 7th bit of Port Isolation port mask of ports 0 and 3 to be 1 while other bits should be configured to 0, and Port Isolation port mask of port 7 still remains all ports. Then, direct communication between downlink ports, 0 and 3, is forbidden and only traffic between uplink port and downlink port can be forwarded。

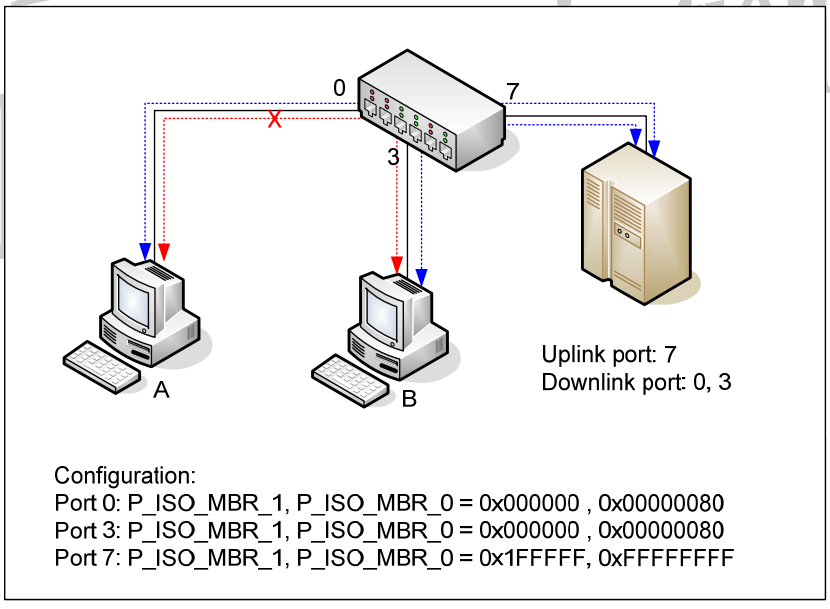


Figure 1: Example of Uplink Port and Downlink Port for Port Isolation

Realtek CONFIDENTIAL CAMEO Communications, Inc.

API REFERENCE

```
rtk_port_isolation_get(uint32 unit, rtk_port_t port, rtk_portmask_t *pPortmask);
rtk_port_isolation_set(uint32 unit, rtk_port_t port, rtk_portmask_t portmask);
rtk_port_isolation_add(uint32 unit, rtk_port_t port, rtk_port_t iso_port);
rtk_port_isolation_del(uint32 unit, rtk_port_t port, rtk_port_t iso_port);
```

7.2 VLAN-based Isolation

Sometimes, uplink ports communicate with uplink port and downlink ports, but downlink ports don't be allowed to communicate with each other in VLAN domain. Customer can use VLAN-based Isolation to achieve the application. The device provides 16-set VLAN-based Isolation configurations in PORT_ISO_VB_ISO_PM_CTRL register as Table 64. In each configuration, customers can define a VID register and using VALID register to enable or disable the entry. Besides, system applies register PORT_ISO_VB_CTRL.VLAN_TYPE to select VID of received packet to comparing register VID. Look at following two examples:

Example 1: Received packet with (outer VID 100) and (inner VID 200). VLAN_PORT_FWD defined in VLAN module is configured in inner VLAN, and VLAN translation is disabled.
In VLAN_TYPE = 0, comparing VID = 200.
In VLAN_TYPE = 1, comparing VID = 100.
In VLAN_TYPE = 2, comparing VID = 200.

Example 2: Same as example 1, but VLAN translation is enabled that (outer VID 100 is translated to 150) and (inner VID 200 is translated to 250).
In VLAN_TYPE = 0, comparing VID = 250.
In VLAN_TYPE = 1, comparing VID = 150.
In VLAN_TYPE = 2, comparing VID = 250.

Table 63: PORT_ISO_VB_CTRL Register

Field Name	Bits	Description
VLAN_TYPE	2	VLAN ID type used for comparing. 2'b00: inner VLAN 2'b01: outer VLAN 2'b10: forwarding VLAN 2'b11: reserved

Table 64: PORT_ISO_VB_ISO_PM_CTRL Register

Field Name	Bits	Description
VB_ISO_MBR	29	VLAN-based isolated port-mask (port 28 to 0). Ports configured as 1 in this port mask could communicate with all ports belonging to the same specified VLAN. Ports configured as 0 could only communicate with ports configured as 1 belonging to the same specified VLAN.
VID	12	VLAN ID.
VALID	1	Dedicate the status of VLAN-based isolation entry. 1'b0: entry invalid 1'b1: entry valid

As port-based isolation, a 29-bit port mask is supported to defined communication port list for each entry. The ports with bit value 0 can't communicate with other ports having same bit value 0. However, they can communicate with the other port with bit value 1. If bit value of ports is set to 1, the port can communicate to all ports in the VLAN. Sometimes, bit value of uplink port is set to 1 and bit value of downlink port is set to 0 in network environment. For example as , ports 0 and 1 are uplink ports and other ports are downlink ports. Only configure (VB_ISO_MBR_0) of VLAN-based Isolation entry to be (0x3), then uplink ports can communicate to all ports, but downlink ports only

Realtek CONFIDENTIAL CAMEO Communications, Inc.

communicate to uplink port.

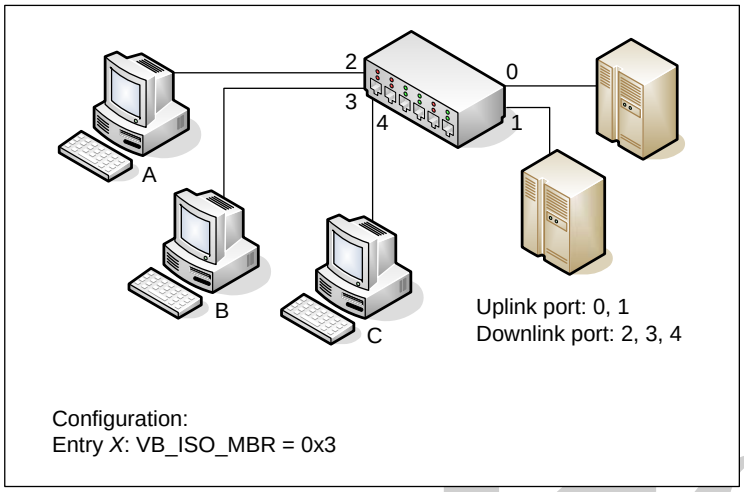


Figure 2: Example of Uplink Port and Downlink Port for VLAN-based Isolation

API REFERENCE

```
rtk_port_vlanBasedIsolationEntry_get(uint32 unit, uint32 index, rtk_port_vlanIsolationEntry_t* pEntry);
rtk_port_vlanBasedIsolationEntry_set(uint32 unit, uint32 index, rtk_port_vlanIsolationEntry_t* pEntry);
rtk_port_vlanBasedIsolation_vlanSource_get(uint32 unit, rtk_port_vlanIsolationSrc_t *pVlanSrc);
rtk_port_vlanBasedIsolation_vlanSource_set(uint32 unit, rtk_port_vlanIsolationSrc_t vlanSrc);
```

PROGRAMMING EXAMPLE

Adding VLAN-based Isolation Entry and VLAN-based Isolation VID Type Configuration

```
/* Configure VLAN-based isolation entry value */
rtk_port_vlanIsolationEntry_t entry;
memset(&entry, 0, sizeof( rtk_port_vlanIsolationEntry_t )) ;
entry.enable = ENABLED;
entry.vid = 100;
entry.portmask.bits[0] = 0x3;

/* Add VLAN-based isolation entry */
rtk_port_vlanBasedIsolationEntry_set(unit, 0, &entry);


/* Configure inner VID to be VLAN-based isolation comparing VID */
vlanSrc = VLAN_ISOLATION_SRC_INNER;
rtk_port_vlanBasedIsolation_vlanSource_set(unit, vlanSrc) ;
```

Realtek CONFIDENTIAL

8 Reserved Multicast Address (RMA)

The device supports RMA with DMAC from 0x01-80-C2-00-00-00 to 0x01-80-C2-00-00-2F and some RMA with special ether type.

8.1 Learning

Some protocols, such as STP and LACP which use port MAC as the source MAC of the control frame, it is useless to learn the port MAC. Thus, the device per RMA can specify whether to learn the source MAC address of a RMA frame. The learning behavior of STP state is higher than which of RMA_SMAC_LRN_CTRL register.

Table 65: RMA_SMAC_LRN_CTRL Register

Field Name	Bits	Description
LRN	1	Learning source MAC address for RMA packet. 1'b0: Not learn 1'b1: Learn

API REFERENCE

```
typedef    enum rtk_trap_rmaGroup_frameType_e
{
    RMA_GROUP_TYPE_SLOW_PROTOCOL_OAM = 0,
    RMA_GROUP_TYPE_SLOW_PROTOCOL_OTHER,
    RMA_GROUP_TYPE_03,
    RMA_GROUP_TYPE_0E_EXCEPT_PTP_LLDP,
    RMA_GROUP_TYPE_10,
    RMA_GROUP_TYPE_GMRP,
    RMA_GROUP_TYPE_GVRP,
    RMA_GROUP_TYPE_MSRP,
    RMA_GROUP_TYPE_0X,
    RMA_GROUP_TYPE_1X,
    RMA_GROUP_TYPE_2X,
    RMA_GROUP_TYPE_END
} rtk_trap_rmaGroup_frameType_t;

rtk_trap_rmaGroupLearningEnable_get
(uint32 unit, rtk_trap_rmaGroup_frameType_t rmaGroup_frameType, rtk_enable_t *pEnable)
rtk_trap_rmaGroupLearningEnable_set
(uint32 unit, rtk_trap_rmaGroup_frameType_t    rmaGroup_frameType, rtk_enable_t enable)
```

8.2 RMA Action

The device supports actions for handling behavior of RMA.

The forward action is to lookup L2 table first. If lookup hit, the packet will be forwarded to the portmask of hit entry. Otherwise, the packet will be flooding to the portmask of lookup miss which is retrieved from VLAN profile setting when L2 lookup miss action is forward, or drop packet when L2 lookup miss action is drop.

The flood action will bypass egress VLAN filtering. The action is to do L2 table lookup first. If lookup hit, the packet will be forwarding to the portmask of hit entry. Else the packet will be flooding to the portmask of lookup miss which is retrieved from VLAN profile setting when L2 lookup miss action is forward, or drop packet when L2 lookup miss action is drop. For BPDU, the flood portmask is retrieved from RMA_BPDU_FLD_PMSK register.

Table 66: RMA_BPDU_FLD_PMSK Register

Field Name	Bits	Description
PMSK	53	Flooding portmask for BPDU when BPDU action is flood to all ports.

API REFERENCE

*rtk_trap_bpduFloodPortmask_get(uint32 unit, rtk_portmask_t *pflood_portmask)*
*rtk_trap_bpduFloodPortmask_set(uint32 unit, rtk_portmask_t *pflood_portmask)*

RMA and normal L2 multicast can have different forwarding behavior. The device supports to control RMA bypass L2 multicast DA lookup miss action. Hence when RMA is enabled to bypass L2 multicast DA lookup miss action, the RMA action will take effect regardless of L2 multicast DA lookup miss action.

Table 67: RMA_CTRL_1 Register

Field Name	Bits	Description
RMA_BYPASS_LM_ACT	1	Bypass L2 multicast DA lookup miss action for all RMA frame. 1'b0: Disable 1'b1: Enable

API REFERENCE

*rtk_trap_rmaLookupMissActionEnable_get(uint32 unit, rtk_enable_t *pEnable)*
rtk_trap_rmaLookupMissActionEnable_set(uint32 unit, rtk_enable_t enable)

8.3 BPDU

The MAC address of BPDU is 01-80-C2-00-00-00. The device supports per port register to control the behavior for BPDU.

Table 68: RMA_PORT_BPDU_CTRL Register

Field Name	Bits	Description
ACT	2	BPDU behavior configuration. 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood.

The device also supports BPDU to control if learning source MAC address.

Table 69: RMA_MGN_LRN_CTRL Register

Field Name	Bits	Description
BPDU_LRN	1	Learning behavior for BPDU. 1'b0: Not learn 1'b1: Learn

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

rtk_trap_portMgmtFrameAction_get
*(uint32 unit, rtk_port_t port, rtk_trap_mgmtType_t frameType, rtk_action_t *pAction)*
rtk_trap_portMgmtFrameAction_set
(uint32 unit, rtk_port_t port, rtk_trap_mgmtType_t frameType, rtk_action_t action)

rtk_trap_mgmtFrameLearningEnable_get
*(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_enable_t *pEnable)*
rtk_trap_mgmtFrameLearningEnable_set
(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_enable_t enable)

8.4 PTP and LLDP

The ether type of PTP is 0x88F7 and which of LLDP is 0x88CC. The device supports per port registers to control the behavior for PTP and LLDP.

Table 70: RMA_PORT_PTP_CTRL Register

Field Name	Bits	Description
ACT	2	PTP behavior configuration. 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood

Table 71: RMA_PORT_LLDP_CTRL Register

Field Name	Bits	Description
ACT	2	LLDP behavior configuration. 2'b00: Table lookup 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood

The device also supports PTP and LLDP to control if learning source MAC address.

Table 72: RMA_MGN_LRN_CTRL Register

Field Name	Bits	Description
PTP_LRN	1	Learning behavior for PTP. 1'b0: Not learn 1'b1: Learn
LLDP_LRN	1	Learning behavior for LLDP. 1'b0: Not learn 1'b1: Learn

API REFERENCE

rtk_trap_portMgmtFrameAction_get
*(uint32 unit, rtk_port_t port, rtk_trap_mgmtType_t frameType, rtk_action_t *pAction)*
rtk_trap_portMgmtFrameAction_set
(uint32 unit, rtk_port_t port, rtk_trap_mgmtType_t frameType, rtk_action_t action)

rtk_trap_mgmtFrameLearningEnable_get
*(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_enable_t *pEnable)*
rtk_trap_mgmtFrameLearningEnable_set

Realtek CONFIDENTIAL CAMEO Communications, Inc.

(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_enable_t enable)

8.5 Other RMA

Besides BPDU, PTP, and LLDP, the device also supports the following RMA in Table 9. Packet with DMAC: 0x01-80-C2-00-00-02, ether type: 0x8809, subtype: 0x03 is taken as OAM packet. If multiple RMA hit at the same time, the priority is: OAM_ACT > RMA_02_ACT > RMA_03_ACT > RMA_0E_ACT > RMA_10_ACT > RMA_20_ACT > RMA_21_ACT > RMA_22_ACT > RMA_0X_ACT > RMA_1X_ACT > RMA_2X_ACT.

Table 73: RMA_CTRL_0 Register

Field Name	Bits	Description
OAM_ACT	2	RMA forwarding behavior configuration. 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_02_ACT	2	DMAC is 01-80-C2-00-00-02 excluding OAM packet 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_03_ACT	2	DMAC is 01-80-C2-00-00-03 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_0E_ACT	2	DMAC is 01-80-C2-00-00-0E excluding PTP and LLDP 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_10_ACT	2	DMAC is 01-80-C2-00-00-10 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_20_ACT	2	DMAC is 01-80-C2-00-00-20 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_21_ACT	2	DMAC is 01-80-C2-00-00-21 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_22_ACT	2	DMAC is 01-80-C2-00-00-22 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_0X_ACT	2	DMAC is 01-80-C2-00-00-01~01-80-C2-00-00-0F, excluding those listed above excluding listed above 2'b00: Forward

Realtek CONFIDENTIAL CAMEO Communications, Inc.

		2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_1X_ACT	2	DMAC is 01-80-C2-00-00-11~01-80-C2-00-00-1F, excluding those listed above excluding listed above 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
RMA_2X_ACT	2	DMAC is 01-80-C2-00-00-21~01-80-C2-00-00-2F, excluding those listed above excluding listed above 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood

The PTP and LLDP configurations have higher priority than the RMA configuration in this section.

API REFERENCE

rtk_trap_rmaGroupAction_get
*(uint32 unit, rtk_trap_rmaGroup_frameType_t rmaGroup_frameType, rtk_trap_rma_action_t *pRma_action)*
rtk_trap_rmaGroupAction_set
(uint32 unit, rtk_trap_rmaGroup_frameType_t rmaGroup_frameType, rtk_trap_rma_action_t rma_action)

8.6 STP Bypass

The device supports to bypass STP behavior regardless the port state of STP is blocking or learning. The STP bypass action only works if the action of RMA is trap to CPU. If the packet is STP bypassed, the source MAC address learning is decided by RMA_SMAC_LRN_CTRL register and STP state.

Table 74: RMA_CTRL_1 Register

Field Name	Bits	Description
RMA_0X_BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree for RMA 01-80-C2-00-00-02 through 01-80-C2-00-00-0F exclude 01-80-C2-00-00-02. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
RMA_03_BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree for RMA 01-80-C2-00-00-03. 0'b0: Drop by STP 0'b1: Bypass (works only if the ACT of LLDP is trapping to CPU)
LLDP_BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree for LLDP. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
PTP_BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree for PTP. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
SLOW_PROTO_BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree for slow protocol (01-80-C2-00-00-02). 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)

API REFERENCE

CONFIDENTIAL
CAMEO Communications, Inc

```
typedef enum rtk_trap_bypassStpType_e
{
    BYPASS_STP_TYPE_USER_DEF_0 = 0,
    BYPASS_STP_TYPE_USER_DEF_1,
    BYPASS_STP_TYPE_USER_DEF_2,
    BYPASS_STP_TYPE_USER_DEF_3,
    BYPASS_STP_TYPE_USER_DEF_4,
    BYPASS_STP_TYPE_USER_DEF_5,
    BYPASS_STP_TYPE_SLOW_PROTO,
    BYPASS_STP_TYPE_RMA_03,
    BYPASS_STP_TYPE_RMA_0X,
    BYPASS_STP_TYPE_EAPOL,
    BYPASS_STP_TYPE_PTP,
    BYPASS_STP_TYPE_LLDP,
    BYPASS_STP_TYPE_END
} rtk_trap_bypassStpType_t;

rtk_trap_bypassStp_get(uint32 unit, rtk_trap_bypassStpType_t frameType, rtk_enable_t *pEnable)
rtk_trap_bypassStp_set(uint32 unit, rtk_trap_bypassStpType_t frameType, rtk_enable_t enable)
```

8.7 VLAN Bypass

The device supports to bypass VLAN drop which decides by VLAN ingress filter, VLAN error, VLAN accept frame type, or CFI = 1. The bypass action only works if the action of RMA is trap to CPU. If the packet is STP bypassed, the source MAC address learning is decided by RMA_SMAC_LRN_CTRL register and STP state.

Table 75: RMA_CTRL_1 Register

Field Name	Bits	Description
RMA_PTP_BYPASS_VLAN	1	Bypass VLAN drop for PTP. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
RMA_LLDP_BYPASS_VLAN	1	Bypass VLAN drop for LLDP.
RMA_BPDU_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-00.
RMA_OAM_BYPASS_VLAN	1	Bypass VLAN drop for OAM.
RMA_SLOWPRO_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-02 excluding OAM.
RMA_03_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-03.
RMA_0E_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-0E.
RMA_GMRP_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-20.
RMA_GVRP_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-21.
RMA_0X_BYPASS_VLAN	1	Bypass VLAN drop for RMA is 01-80-C2-00-00-0x. Excluding RMA listed above.

API REFERENCE

```
rtk_trap_bypassVlan_get(uint32 unit, rtk_trap_bypassVlanType_t frameType, rtk_enable_t *pEnable)
rtk_trap_bypassVlan_set(uint32 unit, rtk_trap_bypassVlanType_t frameType, rtk_enable_t enable)
```

8.8 User Defined RMA

The device supports 6 user defined RMA settings. Using them you can configure specific MAC address as RMA. The fields of each user defined RMA setting are shown as Table 12 and Table 13. The whole 48 bits can be

configured in user defined RMA 0 & 1, but only LSB 8bits can be configured in user defined RMA 2 & 3 & 4 & 5, the MSB 40bits are taken as 0x01-80-C2-00-00.

Table 76: RMA_USR_DEF_CTRL_SET 0 & 1 Register

Field Name	Bits	Description
ADDR	48	Specific MAC address as RMA. [XX:XX:XX:XX:XX:XX]
BYPASS_VLAN	1	Bypass VLAN drop. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
LRN	1	Learning behavior. 1'b0: Not learn 1'b1: Learn
BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
ACT	2	Behavior configuration. 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
EN	1	1'b0: Disable 1'b1: Enable

Table 77: RMA_USR_DEF_CTRL_SET 2 & 3 & 4 & 5 Register

Field Name	Bits	Description
ADDR	8	Specific MAC address as RMA. [01:80:C2:00:00:XX]
BYPASS_VLAN	1	Bypass VLAN drop. 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
LRN	1	Learning behavior. 1'b0: Not learn 1'b1: Learn
BYPASS_STP	1	Bypass "Blocking" and "Learning" port state of spanning tree 1'b0: Disable 1'b1: Enable (only works if the ACT is trap to CPU)
ACT	2	Behavior configuration. 2'b00: Forward 2'b01: Drop 2'b10: Trap to CPU 2'b11: Flood
EN	1	1'b0: Disable 1'b1: Enable

The behavior control priority is user defined RMA 0 > RMA1 > RMA2 > RMA3 > RMA4 > RMA5. The behavior control of user defined RMA is higher than which of BPDU, PTP, LLDP, and other RMA.

API REFERENCE

*rtk_trap_userDefineRma_get(uint32 unit, uint32 userDefine_idx, rtk_trap_userDefinedRma_t *pUserDefinedRma)*
*rtk_trap_userDefineRma_set(uint32 unit, uint32 userDefine_idx, rtk_trap_userDefinedRma_t *pUserDefinedRma)*

*rtk_trap_bypassVlan_get(uint32 unit, rtk_trap_bypassVlanType_t frameType, rtk_enable_t *pEnable)*
rtk_trap_bypassVlan_set(uint32 unit, rtk_trap_bypassVlanType_t frameType, rtk_enable_t enable)

*rtk_trap_userDefineRmaLearningEnable_get(uint32 unit, uint32 userDefine_idx, rtk_enable_t *pEnable)*

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

rtk_trap_userDefineRmaLearningEnable_set(uint32 unit, uint32 userDefine_idx, rtk_enable_t enable)

*rtk_trap_bypassStp_get(uint32 unit, rtk_trap_bypassStpType_t frameType, rtk_enable_t *pEnable)*
rtk_trap_bypassStp_set(uint32 unit, rtk_trap_bypassStpType_t frameType, rtk_enable_t enable)

*rtk_trap_userDefineRmaAction_get(uint32 unit, uint32 userDefine_idx, rtk_trap_rma_action_t *pAction)*
rtk_trap_userDefineRmaAction_set(uint32 unit, uint32 userDefine_idx, rtk_trap_rma_action_t action)

*rtk_trap_userDefineRmaEnable_get(uint32 unit, uint32 userDefine_idx, rtk_enable_t *pEnable)*
rtk_trap_userDefineRmaEnable_set(uint32 unit, uint32 userDefine_idx, rtk_enable_t enable)

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

9 Trunk

Trunk mechanism can aggregate several physical ports to a logical port for higher bandwidth. The device provides at most 8 groups of trunk configuration. Each trunk group can aggregate at most 8 ports. For trunk ports traffic balancing, a hash function is applied and the hash parameters can be configured by user. There are 2 sets of hash algorithm configurations, each trunk group can bind to a set of configuration. The device also provide traffic separation mechanism to choose the link maximum id member port dedicated for known multicast traffic or flooding traffic.

Port id could be physical port id or logical port id (trunk id). Mirror, ingress and egress bandwidth control module base on physical port not logic port, however, almost all of the other modules, such as storm filter, VLAN, L2 table and so on, port id means logical port.

9.1 Trunk Member

TRK_MBR register field decides the trunk member port-mask. At most 8 ports can be configured as member port. Only 8 member ports would be chose as TX port if member ports are configured more than 8 ports.

API REFERENCE

```
rtk_trunk_port_get(uint32 unit, uint32 trk_gid, rtk_portmask_t *pTrunk_member_portmask);
rtk_trunk_port_set(uint32 unit, uint32 trk_gid, rtk_portmask_t *pTrunk_member_portmask);
```

9.2 Load Balancing

Traffic to trunk port would be distributed to member ports by hash function. The device takes packet header contents as hash key. The keys are configurable by 7 bits HASH_MSK register field (per set configuration for 4 sets) which contains:

- SPA(bit 0): source physical port
- SMAC(bit 1): source MAC address
- DMAC(bit 2): destination MAC address
- SIP(bit 3): source IP
- DIP(bit 4): destination IP
- SPORT(bit 5): source layer 4 port
- DPORT(bit 6): destination layer 4 port

The hash key can be any combination of candidates, such as set HASH_MSK to 0x7 choosing SPA, SMAC, DMAC as hash key.

The hash function does XOR operation for every 4 bits of hash keys and produces a 4 bits hash result. Each hash key can be shifted several bits before XOR operation to avoid traffic unbalanced because of the way of folding the keys. For example, if SPA hash key shift 2 bits, the key folding is:

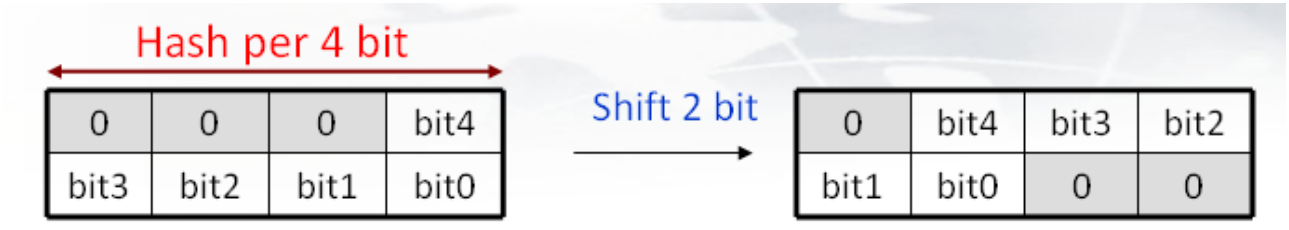


Figure 14: Example of SPA Hash Key Shift

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

The register fields to configure shift bits are SPA_SHIFT_BITS, SMAC_SHIFT_BITS, DMAC_SHIFT_BITS, SIP_SHIFT_BITS, DIP_SHIFT_BITS, SPORT_SHIFT_BITS, and DPORT_SHIFT_BITS. These shifts are per set configuration.

The 4 bits hash result (range from 0 ~ 15) will be divided by the number of TX candidate member ports, and the remainder decides the packet output port. The candidate ports are link up member ports that are not dedicated for separated traffic (about separated traffic please refer to). Output port is the (remainder+1)th link member port counting from least ID. For example, if hash result is 10 and there are 3 TX candidate trunk member ports (1,3,5), the remainder is 1 thus 2nd port (port 3) is output port.

Register field HASH_MSK_IDX (per trunk configuration) binds a trunk group to a hash algorithm set. It is 2 bits length field since there are 4 sets.

API REFERENCE

```
rtk_trunk_distributionAlgorithmBind_get(uint32 unit, uint32 trk_gid, uint32 *pAlgo_idx);
rtk_trunk_distributionAlgorithmBind_set(uint32 unit, uint32 trk_gid, uint32 algo_idx);
rtk_trunk_distributionAlgorithmParam_get(uint32 unit, uint32 algo_idx, uint32 *pAlgo_bitmask);
rtk_trunk_distributionAlgorithmParam_set(uint32 unit, uint32 algo_idx, uint32 algo_bitmask);
rtk_trunk_distributionAlgorithmShift_get(uint32 unit, uint32 algo_idx, rtk_trunk_distAlgoShift_t *pShift);
rtk_trunk_distributionAlgorithmShift_set(uint32 unit, uint32 algo_idx, rtk_trunk_distAlgoShift_t *pShift);
```

9.3 Traffic Separation

Traffic is hashed to choose output port, thus the available bandwidth of a traffic is not stationary. It has chance that a streaming traffic is flow controlled because it is transmitted using the same port as a low priority high speed traffic. Another situation is that a user may want L2 lookup miss traffic be separated from other traffic since they have more chance to be malevolence. The device provides a traffic separation mechanism to separate known multicast traffic or L2 lookup miss traffic to a specific port. By configuring SEPARATE_TRAFFIC register field (per trunk group configuration), the device could disable traffic separation, separate known multicast traffic, separate L2 lookup miss traffic, or separate both known multicast traffic and L2 lookup miss traffic.

The largest ID link up member port is the dedicated port to output separated traffic. The port would not be a hash candidate port if traffic separation is enabled. For example, if link up trunk member ports are port 1,3,5,7, and traffic separation is configured to separate known multicast traffic, port 7 would be dedicated to output known multicast traffic, and other traffic would output through port 1,3,5.

Traffic separation mechanism would automatically stop if there is 0 or only 1 link up trunk member port. It would work only when there is more than 1 link up member ports.

Table 78: TRK_SEP_TRAFFIC_CTRL Register

Field Name	Bits	Description
SEP_TRAFFIC	1	Separate known multicast / flooding(L2 look up missed) traffic for a trunk port. When this option is enabled, send known multicast packet/flooding packet to MSB port of a trunk instead of following hashing result in the link aggregation group N. If this option is 1 or 2, the MSB port of a trunk would not join the hash selection but just dedicated for multicast traffic or flooding traffic. 0b00: disable traffic separation

CONFIDENTIAL
CAMEO Communications, Inc.

	0b01: active MSB port is dedicated to known multicast traffic 0b10: active MSB port is dedicated to flooding traffic 0b11: active MSB port is dedicated to known multicast and flooding traffic

API REFERENCE

```
rtk_trunk_trafficSeparate_get(uint32 unit, uint32 trk_gid, rtk_trunk_separateType_t *pSeparateType);
rtk_trunk_trafficSeparate_set(uint32 unit, uint32 trk_gid, rtk_trunk_separateType_t separateType);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

10 Mirror

Mirror is a useful mechanism to monitor traffic. By configuring interest patern of traffic (such as RX/TX port, or classification rules), a copy of mirrored traffic would be forwarded to mirroring port for further analysis. The device support 4 mirroring set configuration. Each mirroring set can configure a mirroring port that a copy of mirrored packets would be sent to. The mirrored packets can be specified by ingress portmask, egress portmask, or flow based (by configuring ACL). To mirror traffic across swithes, the device supports RSPAN feature.

Since mirror is a traffic monitor feature, the mirroring traffic should not trigger flow control to affect traffic normal forwording. The device would automatically stop mirroring if the mirroring port buffer reaches flow control threshold. Mirroring would be automatically back to work when the mirroring port has resource to TX mirrored traffic.

10.1 Ingress/Egress Mirror

Ingress mirror is to mirror the original ingress packets and egress mirror is to mirror the modified egress packets. For example, a VLAN un-tag packet received from port 1 outgoes to port 2 with VLAN tagged. If port 1 is ingress mirrored, a copy of the packet without VLAN tag would be sent to mirroring port. If port 2 is egress mirrored, a copy with VLAN tag would be sent to mirroring port. For ingress mirror, both good and bad packets such as CRC error, under-size, over-size, alignment error packets would be mirrored. For egress mirror, only packets not dropped by any module can be mirrored. For example, the packets dropped by egress bandwidth control would not be egress mirrored.

The device supports 4 mirroring set, enable status of each mirroring set is decided by MIR_EN register field. MIR_DST_P is a 5 bits port ID that decides the mirroring port if MIR_EN is enabled. Each mirroring set can configure SPM_0 to specify the ports to do ingress mirror. For example, to mirror packets received from port 1-4 can be achieved by setting SPM_0 to 0xf, and set MIR_EN to 1 for a mirroring set.

Egress mirror portmask is specified by DPM_0. If any outgoing ports of packets hit the portmask, the packet satisfies the egress mirror condition. For example, configure DPM_0 to 0x10 (mirror port 5), and a multicast packet outgoing to port 1,3,5 would be mirrored since it goes to port 5.

If user wants to specify both ingress portmask and egress portmask for mirror condition. MIR_OP can be used to decide the packets hit SPM "AND" DPM or hit SPM "OR" DPM would be mirrored. If MIR_OP is 0 (OR), the packet coming port hitting SPM or outgoing port hitting DPM would be mirrored. If MIR_OP is 1 (AND), only the packet hits both SPM and DPM would be mirrored, thus if SPM or DPM is all zero, the mirror condition would be never satisfied.

For a packet hit both SPM and DPM that configured in the same mirroring set, the device would follow MIR_SEL to do either ingress mirror or egress mirror thus only a copy of mirrored packet would be sent to mirroring port. If MIR_SEL is 0, the egress modified packet would be mirrored; if MIR_SEL is 1, the original ingress packet would be mirrored.

For a user that does not want to outgo normal forwarded packet to mirroring port, DST_P_ISO can be configured to 1 so only mirrored packets can pass to mirroring port. If DST_P_ISO is 0, the mirroring port may receive normal forwarded traffic and mirrored traffic. If a packet is both normal forwarded and mirrored to mirroring port, two packets would outgo to mirroring port: one is normal forwarded, another is mirrored.

If mirrored port and mirroring port is not including in the same VLAN, CROSS_VLAN can be configured to 1 so the mirrored packet can cross VLAN. If CROSS_VLAN is 0, the mirrored packet will not mirrored for the ports are in different VLAN.

If a mirror set is both used by ACL mirror and port-based mirror, FLOW_BASED_ONLY can be configured to 1 so the mirror set will only valid for ACL mirror using, it will not do port-based mirror ever. If CROSS_VLAN is 0, the mirror set can be both used by ACL mirror and port-based mirror.

If a Packet is determined to be both mirrored and normal forwarded to mirroring port, DUPLICATE_FLTR can be

Realtek CONFIDENTIAL CAMEO Communications, Inc.

configured to 1 so both mirrored and normal forward packet will be transmitted to mirroring port. If DUPLICATE_FLTR is 0, normal forwarding packet will be transmitted, mirrored packet will be filtered.

If Packet transmitted from mirroring port is determined to be mirrored, SELF_FILTER can be configured to 1 so the mirrored packet will be filtered. If SELF_FILTER is 0, the mirrored packet could be mirrored to mirroring port.

Each mirror set has a MIR_QID_EN and MIR_QID set, if the MIR_QID_EN is configured to 1, when the mirror set is hit, the mirrored packet will be sent to the TX QID specified by MIR_QID. If MIR_QID_EN is 0, the mirrored packet will be sent by normal ways.

It can specify flow-based mirror by configuring FLOW_BASED_PMSK_IGNORE to 1 to ignore SPM/DPM/MIR_OP setting of mirror set. While set to 0, ACL mirror will still using SPM/DPM/MIR_OP for mirror hit decision.

Each mirror set has a MIR_MODE column, if MIR_MODE is configured to 1, mirrored traffic follow mirroring port configuration, like normal forwarding to mirroring port, packet modify will follow mirroring port setting. If MIR_MODE is 0, ingress mirror is original packet, egress mirror is modified packet by ALE, follow egress port configuration.

Table 79: Mirror Entry Related Registers

Register Name	Field Name	Bits	Description
MIR_CTRL	MIR_EN	1	Enable this mirroring set. 1'b0: disable 1'b1: enable
	MIR_SEL	1	Decide which direction of packet would be mirrored if both TX/RX mirror are hit. 1'b0: TX 1'b1: RX
	MIR_OP	1	Decides the traffic should hit SPM OR DPM or hit SPM AND DPM to be mirrored. 1'b0: or 1'b1: and
	DST_P_ISO	1	Isolate the mirroring port from normal forwarding traffic. 1'b0: The mirroring port accepts mirrored traffic and normal forwarding traffic. 1'b1: The mirroring port only accepts mirrored traffic.
	MIR_DST_P	5	Mirroring port.
	CROSS_VLAN	1	Enable cross VLAN of mirrored set. 1'b0: disable 1'b1: enable
	FLOW_BASED_ONLY	1	Set the mirror set for flow based mirror only. 1'b0: for both flow-based and port-based mirror 1'b1: entry valid only for flow-based mirror
	DUPLICATE_FLTR	1	Packet is determined to be both mirrored and normal forwarded to mirroring port, it is transmitted to mirroring port 1'b0: twice 1'b1: once. Duplicated packet is filtered, only keep normal forwarding packet.
	SELF_FILTER	1	Packet transmitted from mirroring port is determined to be mirrored 0b0: it could be mirrored to mirroring port 0b1: the mirrored packet is filterd
	MIR_QID	3	Mirrored traffic enter output queue id at mirroring port
	MIR_QID_EN	1	Specify mirrored traffic output queue id 0b0: disable 0b1: enable

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	FLOW_BASED_PMSK_IGNORE	1	Specify flow-based mirror ignore SPM/DPM/ MIR_OP setting of mirror set 0b0: do not ignore SPM/DPM/ MIR_OP 0b1: ignore SPM/DPM/MIR_OP setting
MIR_SPM_CTRL	SPM_0	29	Bitmap to select RX mirrored port (port 0 to 29). 1'b0: disable 1'b1: enable
MIR_DPM_CTRL	DPM_0	29	Bitmap to select TX mirrored port (port 0 to 29). 1'b0: disable 1'b1: enable
MIR_MODE_CTRL	MIR_MODE	1	Mirror mode 0b0: RX mirror is original packet, TX mirror is modified packet by ALE, follow Tx port configuration 0b1: mirrored traffic follow mirroring port configuration, like normal forwarding to mirroring port

API REFERENCE

```
rtk_mirror_group_init(uint32 unit, rtk_mirror_entry_t *pMirrorEntry);
rtk_mirror_group_get(uint32 unit, uint32 mirror_id,rtk_mirror_entry_t *pMirrorEntry);
rtk_mirror_group_set(uint32 unit,uint32 mirror_id, rtk_mirror_entry_t *pMirrorEntry);
```

NOTE: It is suggested that it should use rtk_mirror_group_init to initialize mirror entry structure before doing directly mirror settings.

10.2 Flow Based Mirror

By flow based mirror, a user can mirror traffic of a specific patern such as mirroring IPv4 traffic only. It is done by configuring ingress or egress ACL to mirror action. By setting action to mirror and configure specific mirror entry ID (MIRROR_INDEX), the packet that its patern hit the ACL classification rule would be mirrored to the mirroring port of the specific entry. Detail of configuring ACL classifications and actions please refer to ACL developer guide.

The ingress ACL mirror can mirror ingress and egress traffic. MIRROR_ACT in ingress ACL action table is used to configure the mirrored traffic should be original (ingress traffic) or modified (egress traffic).

If a packet hit multiple ACL mirror rules, only the rule with lowest ID would take effect. For example, if ACL 100 mirror SA=00:00:00:00:00:01 packet to mirror entry 3; ACL 200 mirror DA=ff:ff:ff:ff:ff:ff packet to mirror entry 2; ACL 300 mirror IPv4 packet to mirror entry 1. If a packet hits all the paterns, it would be mirrored to mirroring port of entry 3 since ACL 100 has lowest ID. However, a packet could hit both ingress and egress ACL rules that it can be mirrored to 2 entries that ingress and egress ACL indicate if the entries ID are different.

Table 80: Flow Based Mirror Related ACL action Fields

Field Name	Bits	Description
MIRROR_INDEX	2	Mirror set index
MIRROR_ACT	1	Mirror original or modified packet 0: Original 1: Modified

10.3 RSPAN

Remote switched port analysis (RSPAN) mechanism is to mirror traffic cross network. Switches in RSPAN architecture could play roles as a source switch, an intermediate switch, or a destination switch as Figure 1.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

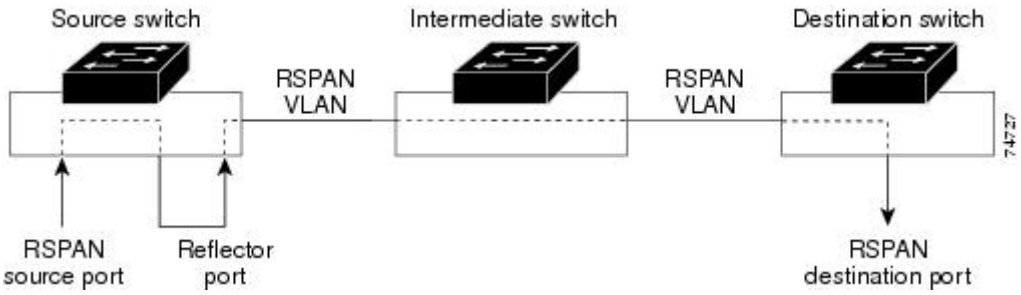


Figure 3: Ingress VLAN Tag Processing Flow

The source switch collects mirrored traffic (behaves like local mirror), and send a copy of mirrored traffic to reflector port. An RSPAN tags would be added to mirrored traffic when it sends out from reflector port. The system provides per mirroring set RSPAN_TX_TAG_ADD register field to control if add RSPAN tag for the mirroring traffic. If RSPAN_TX_TAG_ADD is 1, the mirroring traffic outgo to the mirroring port (playing role of reflector port) would add RSPAN tag. The TPID, priority, CFI, VID field of RSPAN tag are controlled by per mirroring set register RSPAN_TAG_TPID, RSPAN_TAG_PRI, RSPAN_TAG_CFI, and RSPAN_TAG_VID.

The RSPAN traffic forwarding of intermediate switch is based on VLAN flooding. Since RSPAN traffic path may be conflict with real data path, it should be learnt. VLAN profile configuration can be utilized to disable L2 MAC learning for RSPAN VLAN. Detail of configuring VLAN profile please refer to VLAN developer guide. For the case that is necessary to specify RSPAN traffic incoming ports and outgoing ports, VLAN based isolation can be utilized. Detail of configuring VLAN based isolation please refer to port developer guide.

The process of RSPAN destination switch is to identify RSPAN tag, forward it to destination port, and remove the RSPAN tag. To support RSPAN destination session, a user has to enable RSPAN_RX_TAG_EN and RSPAN_RX_TAG_RM for a mirroring set. With RSPAN_RX_TAG_EN enabled, the packets received from any ports of the switch that match the TPID and RSPAN_TAG_VID would be mirrored to mirroring port of the mirroring set. It is suggested that keep DPM and SPM of the mirroring set empty, or the outgoing traffic of mirroring port would contain local mirrored traffic and RSPAN traffic. With RSPAN_RX_TAG_RM enabled, the RSPAN tag of RSPAN traffic would be removed when it outgoes to mirroring port so traffic analyzer could receive the original mirrored data.

Table 81: Mirror Entry Related Registers

Register Name	Field Name	Bits	Description
MIR_RSPAN_TX_CTRL	RSPAN_TX_TAG_ADD	1	Add RSPAN VLAN tag for mirrored packets that do not carry RSPAN VLAN tag for mirroring set n. This bit is ignored by packets which already have RSPAN VLAN tag. 0b0: disable 0b1: enable
MIR_RSPAN_RX_TAG_RM_CTRL	RSPAN_RX_TAG_RM	1	If RSPAN_RX_TAG_EN is 1, this bit decides if the RSPAN VLAN tag of RSPAN packet should be removed. This bit is ignored by packets that do not have RSPAN VLAN tag. 0b0: disable 0b1: enable
MIR_RSPAN_RX_TAG_EN_CTRL	RSPAN_RX_TAG_EN	1	Enable the RX RSPAN VLAN tag identification ability. If it is identified as a RSPAN traffic, it would be mirrored to MIR_DST_P. 0b0: disable 0b1: enable
MIR_RSPAN_VLAN_CTRL	RSPAN_TAG_TPID	16	TPID of RSPAN TPID value. This is used to identify RX RSPAN traffic and to decide the TPID of TX added RSPAN VLAN tag.
	RSPAN_TAG_PRI	3	Priority of RSPAN tag. This is used to decide PRI of the TX added RSPAN VLAN tag.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	RSPAN_TAG_CFI	1	CFI of RSPAN tag. This is used to decide CFI of the TX added RSPAN VLAN tag.
	RSPAN_TAG_VID	12	VID of RSPAN tag. This is used to identify RX RSPAN traffic and to decide VID of the TX added RSPAN VLAN tag.

API REFERENCE

*rtk_mirror_rspanIgrMode_get(uint32 unit, uint32 mirror_id, rtk_mirror_rspanIgrMode_t *pIgrMode);*
rtk_mirror_rspanIgrMode_set(uint32 unit, uint32 mirror_id, rtk_mirror_rspanIgrMode_t igrMode);
*rtk_mirror_rspanEgrMode_get(uint32 unit, uint32 mirror_id, rtk_mirror_rspanEgrMode_t *pEgrMode)*
rtk_mirror_rspanEgrMode_set(uint32 unit, uint32 mirror_id, rtk_mirror_rspanEgrMode_t egrMode)
*rtk_mirror_rspanTag_get(uint32 unit, uint32 mirror_id, rtk_mirror_rspanTag_t *pTag)*
*rtk_mirror_rspanTag_set(uint32 unit, uint32 mirror_id, rtk_mirror_rspanTag_t *pTag)*

10.4 Sampling

Each mirror set has SAMPLE_RATE column to do a sampling of mirrored packets. If SAMPLE_RATE is configured to 0, it equals to normal mirror, do not need to care about sampling rate. Take 100 for example, if SAMPLE_RATE is configured to 100, the 100 mirrored packet will be ignore, when the 101 mirrored packet will be mirrored to mirroring port.

Table 82: MIR_SAMPLE_RATE_CTRL

Field Name	Bits	Description
SAMPLE_RATE	16	The sampling rate of mirrored packets. 0x0: just mirror, do not need to care about sampling rate. n: sample the mirrored packets every n+1 packets

API REFERENCE

*rtk_mirror_ sflowMirrorSampleRate_get(uint32 unit, uint32 mirror_id, uint32 *pRate);*
rtk_mirror_ sflowMirrorSampleRate_set(uint32 unit, uint32 mirror_id, uint32 rate);

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

11 Ingress Bandwidth Control

To limit port base ingress traffic rate, system provide ingress bandwidth control function for every port. When ingress bandwidth exceeds the relevant configured-bandwidth, system could be configured to drop packet. Ingress bandwidth control module could be configured to leak some special protocol traffic, such as BPDU, RMA, IGMP, ARP and Realtek control packet(ethertype = 0x8899).

11.1 Bandwidth Configuration

For each port, the device provides the register IGR_BWCTRL_LB_CTRL. P_LB_EN in Table 83configurations to enable or disable the ingress bandwidth control module. The ingress bandwidth range is 16Kbps~1Gbps configured by IGR_BWCTRL_PORT_RATE_CTRL. P_RATE register in Table 83. The units of both are 16Kbps.

Table 83: IGR_BWCTRL_LB_CTRL Register

Field Name	Bits	Description
P_LB_EN	1	Enable ingress bandwidth control function of the port 1'b0: Disable 1'b1: Enable

Table 84: IGR_BWCTRL_PORT_RATE_CTRL Register

Field Name	Bits	Description
P_RATE	18	Ingress bandwidth control port rate. Unit = 16kbps 0: Blocking; Others: Controlled Rate = RATE[15:0] * 16Kbps Note: LB rate = 1 means 16Kbps, for giga MAC, LB rate >= 0xF424 means 1Gbps, the unused two bits are reserved as dummy register For 2.5G MAC, LB rate >= 0x2625A means 2.5 Gbps.

For easily monitoring, system provides per-port register IGR_BWCTRL_PORT_EXCEED_FLG.EXCEED_FLG in Table 85 to log the exceed status whether the traffic rate beyond the rate configured on the port. Customer could read it to get the log data and write 1 to clear it.

Table 85: IGR_BWCTRL_EXCEED_FLG Register

Field Name	Bits	Description
P_EXCEED_FLG	1	Exceed status of the port. Write 1 to clear. 1'b0: not exceed 1'b1: exceed

API REFERENCE

```
rtk_rate_igrBandwidthCtrlEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_rate_igrBandwidthCtrlEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
rtk_rate_igrBandwidthCtrlRate_get(uint32 unit, rtk_port_t port, uint32 *pRate);
rtk_rate_igrBandwidthCtrlRate_set(uint32 unit, rtk_port_t port, uint32 rate);
rtk_rate_portIgrBandwidthCtrlExceed_get(uint32 unit,rtk_port_t port,uint32 *pIsExceed);
rtk_rate_portIgrBandwidthCtrlExceed_reset(uint32 unit, rtk_port_t port);
```

PROGRAMMING EXAMPLE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Ingress Bandwidth Control Setting and Enabled Ingress Bandwidth Flow Control Configurations

```
/* Enable bandwidth control and rate configured to 10Mbps */
port = 2;
rate = 625;    /* 625*16Kbps = 10Mbps */
rtk_rate_igrBandwidthCtrlEnable_set(unit, port, ENABLED);
rtk_rate_igrBandwidthCtrlRate_set(unit, port, rate);
```

11.2 Protocol Leaky

Ingress bandwidth module provides an option to leak some special protocol flow. It is a global option. To leak a flow means the flow could bypass ingress bandwidth module and it's bandwidth would not be counted to the configured-bandwidth.

The device provides a register IGR_BWCTRL_CTRL in Table 86 for optionally limit following traffic:

- ARPREQ_ADMIT for ARP request and Neighbor Solicitation for ICMPv6
- RMA_ADMIT for RMA packet
- BPDU_ADMIT for BPDU packet
- RTKPKT_ADMIT for the packet with ether type 0x8899 (RRCP、RLPP、RLDP…)
- IGMP_ADMIT for IGMP packet

The traffic could be limited or unlimited by input bandwidth control. If these packets are admitted, they do NOT be counted in ingress bandwidth traffic.

Table 86: IGR_BWCTRL_CTRL Register

Field Name	Bits	Description
ARPREQ_ADMIT	1	Admit ARP Request and Neighbor Solicitation for ICMPv6 (Type:135) packet bypassing ingress bandwidth control. 1'b0: disable 1'b1: enable
RMA_ADMIT	1	Admit RMA (exclude BPDU, include PTP & LLDP) packet bypassing ingress bandwidth control. 1'b0: disable 1'b1: enable
BPDU_ADMIT	1	Admit BPDU packet bypassing ingress bandwidth control. 1'b0: disable 1'b1: enable
RTKPKT_ADMIT	1	Admit Realtek control packet (ethtype=0x8899) bypassing ingress bandwidth control. 1'b0: disable 1'b1: enable
IGMP_ADMIT	1	Admit IGMP packet bypassing ingress bandwidth control. Note: This includes IPv4 IGMP and IPv6 MLDv1/v2. 1'b0: disable 1'b1: enable

API REFERENCE

```
rtk_rate_igrBandwidthCtrlBypass_get(uint32 unit, rtk_rate_igr_bypassType_t bypassType, rtk_enable_t *pEnable);
rtk_rate_igrBandwidthCtrlBypass_set(uint32 unit, rtk_rate_igr_bypassType_t bypassType, rtk_enable_t enable);
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

PROGRAMMING EXAMPLE

Enabled IGMP Packet Bypass Ingress Bandwidth Control Configuration

```
/* Enable IGMP bypass feature */
bypassType = IGR_BYPASS_IGMP;
rtk_rate_igrBandwidthCtrlBypass_set (unit, bypassType, ENABLED);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

12 Storm Control

Storm control works to per port limit traffic receiving rate of specific traffic types thus avoid switch bandwidth or CPU resource massively occupied by malfunction or malicious link partner. The system supports basic storm control to limit rate of unknown unicast traffic, unicast traffic (contains known and unknown unicast), unknown multicast traffic, multicast (contains known and unknown multicast), and broadcast traffic. To avoid control packets be filtered by storm control, registers are provided for protocol packets, such as BPDU, IGMP, and ARP request (containing neighbor solicitation) traffic, to bypass storm control.

12.1 Basic Storm Control

The system per port supports storm control for unicast, multicast, and broadcast traffic. The traffic that exceed the limited rate would be dropped thus the upbound receiving rate is controlled. The limited rates are specified by UC_RATE (for unicast traffic), MC_RATE(for multicast traffic), and BC_RATE(for broadcast traffic). The rate could be packet or byte based decided by MODE field which is a global register field. If MODE is packet based (value=0), the unit of UC_RATE/MC_RATE/BC_RATE is packet per second; if MODE is byte based (value=1), the unit of UC_RATE/MC_RATE/BC_RATE would be 16 Kbps.

When MODE is byte based, if inter frame gap is counted in traffic rate is a global option by configurating INC_IFG (value=1 means count inter frame gap in traffic rate).

The storm control is a leaky bucket mechanism, the configured rate could be considered as leaky rate. When ingress traffic reaches the bucket threshold, the packets would be dropped. The thresholds are global register fields which are UC_BURST, MC_BURST, and BC_BURST. The unit of thresholds is also decided by MODE: if MODE is packet based, the unit would be one packet; if MODE is byte based, the unit would be one byte.

There are options for unicast and multicast traffic to decide if limit both known and unknown traffic or only limit unknown traffic:

- Unicast traffic (DMAC bit 40 = 0)
  - Limit only unknown unicast traffic
    - By configure UC_TYPE to 0
  - Limit both known and unknown unicast traffic
    - By configure UC_TYPE to 1
- Multicast traffic (DMAC bit 40 = 1 and DMAC is not FF:FF:FF:FF:FF:FF)
  - Limit only unknown multicast traffic
    - By configure MC_TYPE to 0
  - Limit both known and unknown multicast traffic
    - By configure MC_TYPE to 1
- Broadcast traffic (DMAC is FF:FF:FF:FF:FF:FF)
  - Always limit both known and unknown broadcast traffic

For example, if user wants to limit unknown multicast and broadcast traffic received from port 1 to 1000 packets per second and with 10 packets bucket threshold. It could be achieved by configuring MODE = 0 (packet based), MC_TYPE of port 1 = 0 (unknown multicast), MC_RATE = 1000, BC_RATE = 1000, MC_BURST_PPS = 10, and BC_BURST_PPS = 10.

Table 87: Storm Control Global Register

Field Name	Bits	Description
INC_IFG	1	Storm control rate include IFG(inter frame gap) and preamble. 1'b0: exclude 1'b1: include
MODE	1	Storm control is based on packet count or byte count. 1'b0: pps 1'b1: bps
UC_BURST	16	Unicast burst size of storm control counter for bps mode.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

		Unit: byte
MC_BURST	16	Multicast burst size of storm control counter for bps mode Unit: byte
BC_BURST	16	Broadcast burst size of storm control counter for bps mode Unit: byte
UC_BURST_PPS	16	Unicast burst size of storm control counter for pps mode. Unit: packet
MC_BURST_PPS	16	Multicast burst size of storm control counter for pps mode Unit: packet
BC_BURST_PPS	16	Broadcast burst size of storm control counter for pps mode Unit: packet

Table 88: Storm Control Per Port Register

Field Name	Bits	Description
UC_LB_EN	1	Unknown unicast storm filtering leaky bucket control. 0x0: disable 0x1: enable Note: if leaky bucket is disable, there is no rate limit ability and interal leaky bucket token counter will be cleared by asic.
UC_TYPE	1	Unicast stream type selection 1'b0: unknown unicast storm 1'b1: both known and unknown unicast storm
UC_RATE	20	Unknown Unicast Storm Filtering Control Rate for Port n. In packet count(pps), the unit is 1pps. For giga port and 10G port, the valid bit is [19:0] in pps: 20'h0: blocking 20'hFFFFF: unlimited In byte count(bps), the unit is 16Kbps. For giga port, the valid bit is [15:0] in bps: 20'h0: blocking 20'hFFFF: unlimited For 10G port, the valid bit is [19:0] in bps: 20'h0: blocking 20'hFFFFF: unlimited
MC_LB_EN	1	Multicast storm filtering leaky bucket control. 0x0: disable 0x1: enable Note: if leaky bucket is disable, there is no rate limit ability and interal leaky bucket token counter will be cleared by asic.
MC_TYPE	1	Multicast Storm type selection. 1'b0: only unknown multicast storm 1'b1: both known and unknown multicast stream
MC_RATE	20	Multicast Storm filtering Control Rate for Port n. (Please refer description of UC_RATE.)
BC_LB_EN	1	Broadcast storm filtering leaky bucket control. 0x0: disable 0x1: enable Note: if leaky bucket is disable, there is no rate limit ability and interal leaky bucket token counter will be cleared by asic.
BC_RATE	20	Broadcast Storm Filtering Control Rate for Port n. (Please refer description of UC_RATE.)

Realtek CONFIDENTIAL CAMEO Communications, Inc.

API REFERENCE

```
rtk_rate_stormControlRateMode_get(uint32 unit,rtk_rate_storm_rateMode_t *pRate_mode);
rtk_rate_stormControlRateMode_set(uint32 unit,rtk_rate_storm_rateMode_t *rate_mode);
rtk_rate_stormControlEnable_get(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type,
rtk_enable_t *pEnable);
rtk_rate_stormControlEnable_set(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type,
rtk_enable_t enable);
rtk_rate_stormControlRate_get(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type, uint32
*pRate);
rtk_rate_stormControlRate_set(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type, uint32
rate);
rtk_rate_stormControlTypeSel_get(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type,
rtk_rate_storm_sel_t *pStorm_sel);
rtk_rate_stormControlTypeSel_set(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type,
rtk_rate_storm_sel_t storm_sel);
rtk_rate_stormControlBurstSize_get (uint32 unit, rtk_rate_storm_group_t storm_type, uint32 *pBurst_size);
rtk_rate_stormControlBurstSize_set (uint32 unit, rtk_rate_storm_group_t storm_type, uint32 burst_size);
```

12.2 Storm Control Bypass Mechanism

User may want to avoid specific L2 control packets being filtered by storm control mechanism. The system provides global bypass control registers:

ARPREQ_ADMIT: if configured value = 1, the ARP request and neighbor solicitation for ICMPv6 packets would bypass storm control.

RMA_ADMIT: if configured value = 1, the packets with RMA DMAC (exclude BPDU, include PTP and LLDP) would bypass storm control.

BPDU_ADMIT: if configured value = 1, the BPDU packets would bypass storm control.

RTKPKT_ADMIT: if configured value = 1, the Realtek control packets (ethertype=0x8899) would bypass storm control.

IGMP_ADMIT: if configured value = 1, the IPv4 IGMP and IPv6 MLDv1/v2 packets would bypass storm control.

Table 89: Bypass Storm Control Register

Field Name	Bits	Description
ARPREQ_ADMIT	1	Admit ARP Request (and Neighbor Solicitation for ICMPv6 (Ethertype = 0x86DD , NextHeader = 0x3A, Type:135) packet bypassing storm control. 1'b0: disable 1'b1: enable
RMA_ADMIT	1	Admit RMA (exclude BPDU, include PTP & LLDP) packet bypassing storm control. 1'b0: disable 1'b1: enable
BPDU_ADMIT	1	Admit BPDU packet bypassing storm control. 1'b0: disable 1'b1: enable
RTKPKT_ADMIT	1	Admit Realtek control packet (ethtype=0x8899) bypassing storm control. 1'b0: disable 1'b1: enable

Realtek CONFIDENTIAL CAMEO Communications, Inc.

IGMP_ADMIT	1	Admit IGMP packet bypassing storm control. Note: This includes IPv4 IGMP and IPv6 MLDv1/v2. 1'b0: disable 1'b1: enable

API REFERENCE

*rtk_rate_stormControlBypass_get(uint32 unit, rtk_rate_storm_bypassType_t bypassType, rtk_enable_t *pEnable);*
*rtk_rate_stormControlBypass_get(uint32 unit, rtk_rate_storm_bypassType_t bypassType, rtk_enable_t *pEnable);*

12.3 Exceed Flag

If storm ever occurs and the upper bound threshold was reached, system would set exceed flag register to 1. The exceed flag is per port and per traffic type 1 bit length register. The flags are "write 1 to clear" thus it allows multiple users to read.

It should be noticed that known and unknown unicast traffic type shares UC_EXCEED_FLG and known and unknown multicast traffic shares MC_EXCEED_FLG. Per port limited storm traffic is known or unknown could be found by reading UC_TYPE or MC_TYPE register which described in 12.1.

Table 90: Storm Control Exceed Flag Per Port Register

Field Name	Bits	Description
BC_EXCEED_FLG	1	Broadcast storm control meter exceed status. Write 1 to clear. 1'b0: not exceed 1'b1: exceed
MC_EXCEED_FLG	1	Multicast storm control meter exceed status. Write 1 to clear. 1'b0: not exceed 1'b1: exceed
UC_EXCEED_FLG	1	Unicast storm control meter exceed status. Write 1 to clear. 1'b0: not exceed 1'b1: exceed

API REFERENCE

*rtk_rate_stormControlExceed_get(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type, uint32 *pIsExceed);*
rtk_rate_stormControlExceed_reset(uint32 unit, rtk_port_t port, rtk_rate_storm_group_t storm_type);

Realtek CONFIDENTIAL CAMEO Communications, Inc.

13 Attack Prevention

The system supports hardware Attack Checker to filter the malicious packets. Attacker attempts to make a machine or network resource unavailable to its intended users. The checker can detect these packets and filter them to protect the receiver. Or trap these packets to CPU to do analysis.

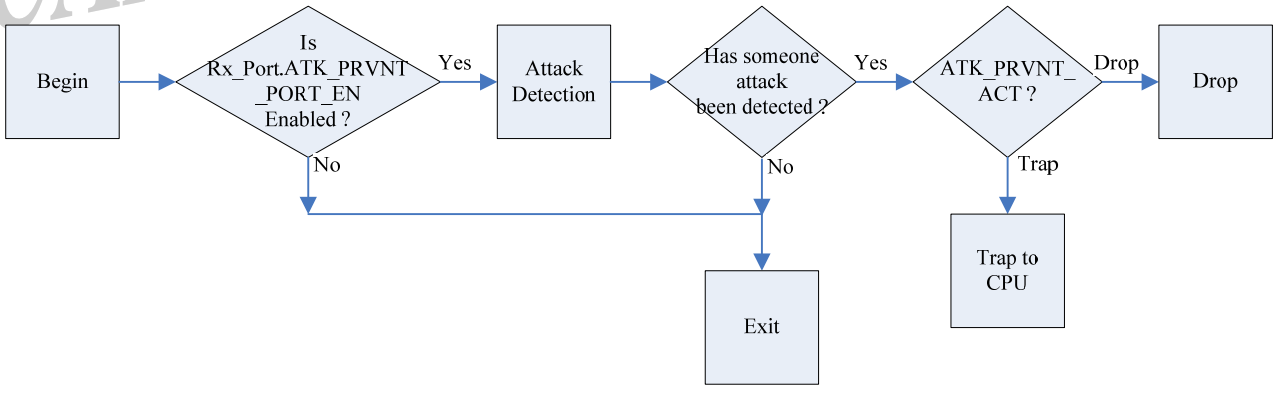
13.1 Attack Detection

The device supports the following types of attack prevention.

Table 91: Attack Types

Type	Description	Action
DA_EQUAL_SA	Source MAC address is equal to Destination MAC Address	Drop/Trap
LAND	IPv4/IPv6 source address is equal to destination address	Drop/Trap
UDP_BLAT	UDP packet with source port = destination port	Drop/Trap
TCP_BLAT	TCP packet with source port = destination port	Drop/Trap
POD	Ping-of-Death, (IP payload length + fragment offset) > 65535.	Drop/Trap
IPV6_FRAG_LEN_MIN	Fragment size of IPv6 datagram < IPV6.FRAG_LEN_MIN (Range: 0~65535)	Drop/Trap
ICMP_FRAG_PKT	ICMP(v4/v6) protocol data are carried in a fragmented IP(v4/v6) datagram.	Drop/Trap
ICMPV4_PING_MAX	(Total length field – header length) > ICMP.PKT_LEN_MAX (Range: 0~65535)	Drop/Trap
ICMPV6_PING_MAX	Payload length > ICMP.PKT_LEN_MAX (Range: 0~65535)	Drop/Trap
SMURF	ICMP(v4/v6) echo request packet with broadcast address	Drop/Trap
TCP_HDR_LEN_MIN	TCP header length < TCP.HDR_LEN_MIN (Range: 0~31)	Drop/Trap
SYN_SPORT_LESS_1024	TCP Control flag SYN=1 & ACK=0 & TCP SPORT<1024	Drop/Trap
NULL_SCAN	TCP Seq_Num = 0, All control flag are zeroes.	Drop/Trap
XMAS	TCP Seq_Num = 0, Control flag (FIN, URG, PSH are set)	Drop/Trap
SYN_FIN	TCP Control flag (SYN, FIN are set)	Drop/Trap
SYN_RST	TCP Control flag (SYN, RST are set)	Drop/Trap
TCP_FRAG_OFF_MIN	TCP packet with IP fragment offset = 1 (means 8Bytes)	Drop/Trap

Packet goes over the functional blocks in below sequence.



Realtek CONFIDENTIAL CAMEO Communication Inc.

Figure 15: Attack Prevention Packet Flow

Table 92: ATK_PRVNT_PORT_EN Register

Field Name	Bits	Description
EN	1	Enable per-port attack prevention.

Table 93: ATK_PRVNT_CTRL Register

Field Name	Bits	Description
DA_EQUAL_SA	1	Filter packets with MACDA = MACSA.
LAND	1	Filter IPv4/IPv6 packets whose SIP = DIP.
UDP_BLAT	1	Filter IPv4/IPv6 UDP packets whose SPORT = DPORT.
TCP_BLAT	1	Filter IPv4/IPv6 TCP packets whose SPORT = DPORT.
POD	1	Filter IPv4/IPv6 packets that length is larger than 64K bytes through fragments. (Ping of Death)
IPV6_FRAG_LEN_MIN	1	Check for minimum size of IPv6 fragments. (If packet length is larger than ATK_PRVNT_IPV6_CTRL.FRAG_LEN_MIN)
ICMP_FRAG_PKT	1	Filter fragmented ICMP packets (IPv4 & IPv6).
ICMPV4_PING_MAX	1	Filter ICMPv4 ping packets with payload size greater than the programmable value in ATK_PRVNT_ICMP_CTRL.PKT_LEN_MAX.
ICMPV6_PING_MAX	1	Filter ICMPv6 ping packets with payload size greater than the programmable value in ATK_PRVNT_ICMP_CTRL.PKT_LEN_MAX.
SMURF	1	Filter ICMP packets of ping request to a broadcast DIP (x.x.x.255) and the SIP is spoofed (victim).
TCP_HDR_LEN_MIN	1	Check first TCP fragmented packet that don't have the full TCP header. (i.e., data offset field in TCP header should have the value larger than or equal to the configured value in ATK_PRVNT_TCP_CTRL.HDR_LEN_MIN.
SYN_SPORT_LESS_1024	1	Filter TCP packets with SYN bit set and ACK bit not set and sport less than 1024.
NULL_SCAN	1	Filter TCP packets with sequence number is zero and the control-bits are zeroes.
XMAS	1	Filter TCP packets with sequence number is zero and the FIN,URG, and PSH bits are set (XMAS Scan attack).
SYN_FIN	1	Filter TCP packets with SYN and FIN set (SYN-FIN Scan attack).
SYN_RST	1	Filter TCP packets with SYN and RST set (SYN-RST Scan attack).
TCP_FRAG_OFF_MIN	1	Filter TCP non-initial fragments carrying TCP header.

Table 94: ATK_PRVNT_ACT Register

Field Name	Bits	Description
DA_EQUAL_SA	1	Determine the action for the DA=SA attack packets. 0b0: drop 0b1: trap
LAND	1	Determine the action for the LAND attack packets. 0b0: drop 0b1: trap
UDP_BLAT	1	Determine the action for the UDP blat attack packets. 0b0: drop 0b1: trap
TCP_BLAT	1	Determine the action for the TCP blat attack packets. 0b0: drop 0b1: trap

Realtek CONFIDENTIAL CAMEO Communications, Inc.

POD	1	Determine the action for the POD attack packets. 0b0: drop 0b1: trap
IPV6_FRAG_LEN_MIN	1	Determine the action for the IPv6 minimum fragmented packets. 0b0: drop 0b1: trap
ICMP_FRAG_PKT	1	Determine the action for the ICMP fragmented packets. 0b0: drop 0b1: trap
ICMPV4_PING_MAX	1	Determine the action for the ICMPv4 ping attack packets. 0b0: drop 0b1: trap
ICMPV6_PING_MAX	1	Determine the action for the ICMPv6 ping attack packets. 0b0: drop 0b1: trap
SMURF	1	Determine the action for the SMURF attack packets. 0b0: drop 0b1: trap
TCP_HDR_LEN_MIN	1	Determine the action for the TCP header too small packets. 0b0: drop 0b1: trap
SYN_SPORT_LESS_1024	1	Determine the action for the SYN_SPORT_LESS_1024 packets. 0b0: drop 0b1: trap
NULL_SCAN	1	Determine the action for the NULLSCAN packets. 0b0: drop 0b1: trap
XMAS	1	Determine the action for the XMAS packets. 0b0: drop 0b1: trap
SYN_FIN	1	Determine the action for the SYNFIN packets. 0b0: drop 0b1: trap
SYN_RST	1	Determine the action for the SYNRST packets. 0b0: drop 0b1: trap
TCP_FRAG_OFF_MIN	1	Determine the action for the TCP_FRAG_OFF_MIN packets. 0b0: drop 0b1: trap

Table 95: ATK_PRVNT_IPV6_CTRL Register

Field Name	Bits	Description
FRAG_LEN_MIN	16	Value to use when checking for minimum size of IPV6 fragments. (Checking is enabled by setting APCR.IPV6_MIN_FRAG_LEN).

Table 96: ATK_PRVNT_ICMP_CTRL Register

Field Name	Bits	Description
PKT_LEN_MAX	16	Maximum length ICMP packet allowed before dropping.

Table 97: ATK_PRVNT_TCP_CTRL Register

Field Name	Bits	Description
HDR_LEN_MIN	5	Minimum TCP header length allowed. (minimum 0 bytes, maximum 31 bytes)

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table 98: ATK_PRVNT_SMURF_CTRL Register

Field Name	Bits	Description
NETMASK	32	Bitmask of Network Mask to check the Smurf.

API REFERENCE

```
rtk_sec_portAttackPreventEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_sec_portAttackPreventEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
rtk_sec_attackPreventAction_get(uint32 unit, rtk_sec_attackType_t attack_type, rtk_action_t *pAction);
rtk_sec_attackPreventAction_set(uint32 unit, rtk_sec_attackType_t attack_type, rtk_action_t action);

rtk_sec_minIPv6FragLen_get(uint32 unit, uint32 *pLength);
rtk_sec_minIPv6FragLen_set(uint32 unit, uint32 length);
rtk_sec_maxPingLen_get(uint32 unit, uint32 *pLength);
rtk_sec_maxPingLen_set(uint32 unit, uint32 length);
rtk_sec_minTCPHdrLen_get(uint32 unit, uint32 *pLength);
rtk_sec_minTCPHdrLen_set(uint32 unit, uint32 length);
rtk_sec_smurfNetmaskLen_get(uint32 unit, uint32 *pLength);
rtk_sec_smurfNetmaskLen_set(uint32 unit, uint32 length);
```

13.2 Validation Check

The device supports to check the validation of specified types of packet.

13.2.1 Invalid ARP Packet

The device supports the following types of attack prevention.

Table 99: Invalid ARP Packet

Type	Description	Action
ARP_INVLD	Invalid IPv6 ARP packet, that is matched the one of the following conditions: 1. For request packet: (1) MAC SA != ARP Sender MAC Address (2) ARP Sender IP = all-zero or multicast or broadcast IP address. 2. For Reply Packet: (1) MAC SA != ARP Sender MAC Address. (2) ARP Target MAC Address = all-zero or multicast address. (3) ARP Target MAC Address != MAC DA. (4) ARP Sender IP = all-zero or multicast or broadcast IP address. (5) ARP Target IP = all-zero or multicast or broadcast IP address.	Forward/Drop/Trap

Table 100: ATK_PRVNT_ARP_INVLD_PORT_ACT Register

Field Name	Bits	Description
ACT	1	Action for ARP invalid packets.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	0b00: Forward 0b01: Drop 0b10: Trap 0b11: Reserved

API REFERENCE

```
rtk_sec_portAttackPrevent_get(
    uint32                  unit,
    rtk_port_t              port,
    rtk_sec_attackType_t    attack_type,
    rtk_action_t            *pAction);
rtk_sec_portAttackPrevent_set(
    uint32                  unit,
    rtk_port_t              port,
    rtk_sec_attackType_t    attack_type,
    rtk_action_t            action);
```

13.2.2 Gratuitous ARP Packet

The device supports the following types of attack prevention.

Table 101: Gratuitous ARP Packet

Type	Description	Action
GRATUITOUS_ARP	Gratuitous ARP packet: For ARP request packet: (Target_HW_ADDR = all-zeroes or broadcast) AND (Sender_PROTO_ADDR = Target_PROTO_ADDR). For ARP reply packet: (Sender_HW_ADDR = Target_HW_ADDR) AND (Sender_PROTO_ADDR = Target_PROTO_ADDR)	Forward/Drop/Trap

Table 102: ATK_PRVNT_PORT_GARP_ACT Register

Field Name	Bits	Description
GARP_ACT	1	Action for Gratuitous ARP packets. 0b00: Forward 0b01: Drop 0b10: Trap 0b11: Copy

API REFERENCE

```
rtk_sec_portAttackPrevent_get(
    uint32                  unit,
    rtk_port_t              port,
    rtk_sec_attackType_t    attack_type,
    rtk_action_t            *pAction);
rtk_sec_portAttackPrevent_set(
    uint32                  unit,
    rtk_port_t              port,
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
    rtk_sec_attackType_t      attack_type,
    rtk_action_t              action);
```

13.3 Notes

1. The default configuration is not to filter any types of attack packets.
2. The MAC address of the packets dropped by Attack Prevention function will not be learnt.
3. The drop priority of Attack Prevention is higher than the trap priority of ingress ACL.
4. The trap priority of Attack Prevention is lower than the drop priority of ingress ACL.

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

14 Priority Decision

Quality of Service (QoS) is a technology for managing network traffic, and provides guarantee bandwidth on ability traffic to network applications or protocols. In IEEE 802.1Q Standard, eight priorities (0~7) are defined and are used to map traffic classes. The device provides eight traffic classes, and eight priorities which can be one-to-one mapping to traffic classes.

14.1 Priority Assignment

Traffic can be classified into traffic classes by different properties or requirements. In the device, five priority assignments are provides. Each priority assignment stands for a special property. Packet can be sent into managed traffic class by the priority assignments for different requirement. The priority assignments supported by the device are:

- Port-Inner-based
- Port-Outer-based
- DSCP-based
- Inner-tag-based
- Outer-tag-based

The five priority assignments would decide a unique 3-bit internal priority through Priority Decision Table for each forwarding packet. The internal priority is mapped to a traffic class using a global Internal Priority to QID Mapping Table. The flow can be referred to Figure 4.

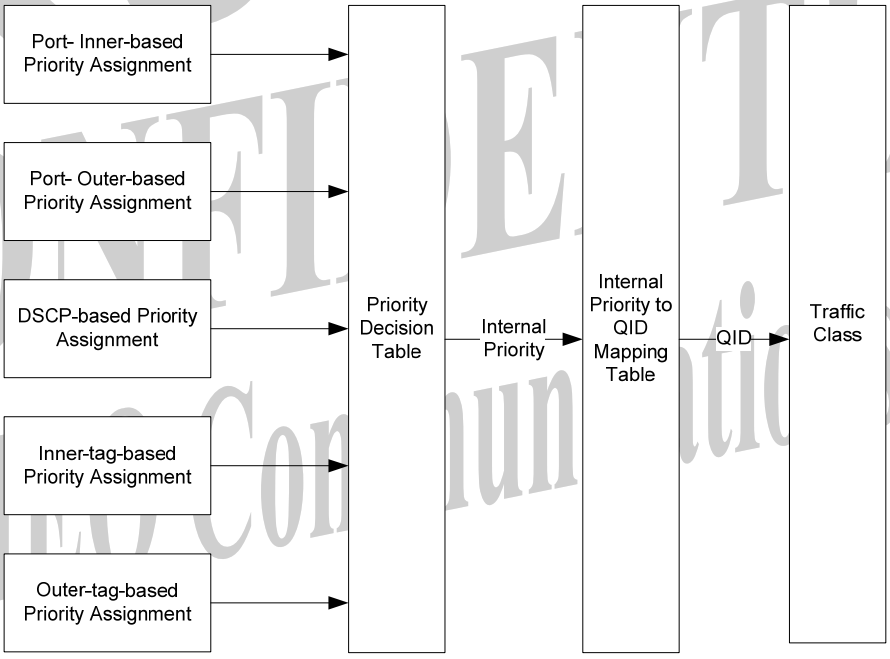


Figure 4: Priority Assignment Flow Chart

14.1.1 Port-based Inner Priority Assignment

The device supports 3-bit Port-Inner-based configuration (PRI_SEL_PORT_PRI register in Table 103) for each RX port.

Table 103: PRI_SEL_PORT_PRI Register

Field Name	Bits	Description

Realtek CONFIDENTIAL CAMEO Communications, Inc.

INTPRI_PORT	3	Port-based Inner priority.

API REFERENCE

```
rtk_qos_portInnerPri _get (uint32 unit, rtk_port_t port, rtk_pri_t *pPri) ;
rtk_qos_portInnerPri _set (uint32 unit, rtk_port_t port, rtk_pri_t pri) ;
```

14.1.2 Port-based Outer Priority Assignment

The device supports 3-bit Port-Outer-based configuration (PRI_SEL_PORT_OTAG_PRI register in Table 104) for each RX port.

Table 104: PRI_SEL_PORT_OTAG_PRI Register

Field Name	Bits	Description
INTPRI_PORT	3	Port-based outer priority.

API REFERENCE

```
rtk_qos_portOuterPri _get (uint32 unit, rtk_port_t port, rtk_pri_t *pPri) ;
rtk_qos_portOuterPri _set (uint32 unit, rtk_port_t port, rtk_pri_t pri) ;
```

14.1.3 DSCP-based Priority Assignment

DSCP-based priority is only applicable to IPv4 or IPv6 packets. For other packets, DSCP-based priority is Null. Traffic Class filed in IPv6 header is an 8-bit field, and it's equal to the type of services (TOS) byte in IPv4 header (please refer Figure 5 and Figure 6). DSCP is in left 6-bit of TOS field, and it's transferred by global DSCP Remapping Table configurations (PRI_SEL_DSCP_REMAP register in Table 105) to get DSCP-based priority.

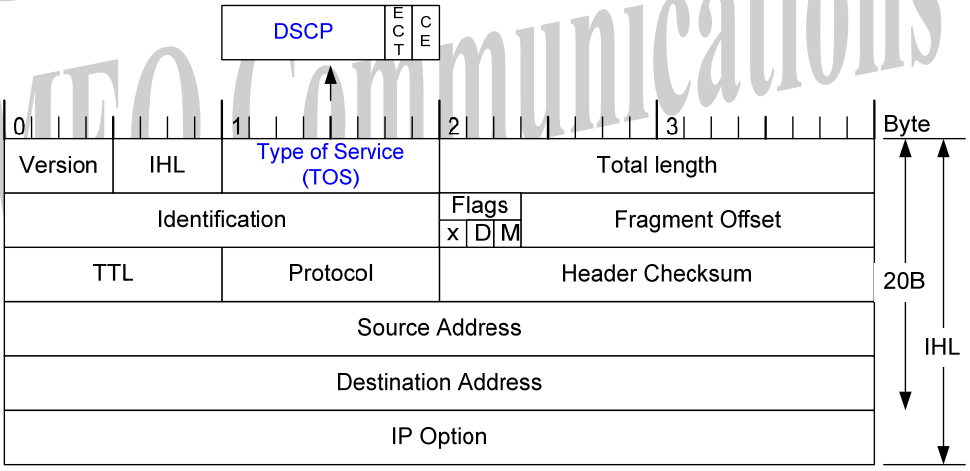


Figure 5: IPv4 Header Format

Realtek CONFIDENTIAL CAMEO Communications, Inc.

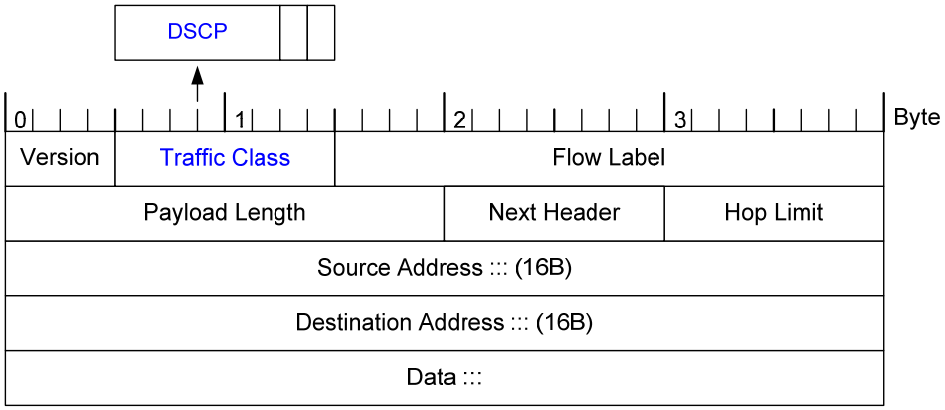


Figure 6: IPv6 Header Format

Besides DSCP remapping table, the device supports an invalid DSCP configuration (PRI_DSCP_INVLD_CTRL0.DSCP in Table 106Table 106Table 106Table 106Table 106, PRI_DSCP_INVLD_CTRL0.DSCP_MSK register in Table 106). If DSCP hits the configuration, DSCP-based priority is treated as Null.

Table 105: PRI_SEL_DSCP_REMAP Register

Field Name	Bits	Description
INTPRI_DSCP	3	Remap DSCP to internal priority.

Table 106: PRI_DSCP_INVLD_CTRL0 Register

Field Name	Bits	Description
DSCP_MSK	6	DSCP value mask.
DSCP	6	DSCP value to be compared. When IP Packet DSCP & DSCP_MSK equal to this field, it means match.

Table 107: PRI_DSCP_INVLD_CTRL1 Register

Field Name	Bits	Description
INVLD_DSCP_PMSK	29	Invalidate special DSCP value port mask. 1 means the port should make DSCP-based priority invalid for ip packet whose DSCP match the pre-defined value.

API REFERENCE

```
rtk_qos_dscpPriRemap_get(uint32 unit, uint32 dscp, rtk_pri_t *pInt_pri);
rtk_qos_dscpPriRemap_set(uint32 unit, uint32 dscp, rtk_pri_t int_pri);
rtk_qos_invldDscpMask_get(uint32 unit, uint32 *pDscpMask);
rtk_qos_invldDscpMask_set(uint32 unit, uint32 dscpMask);
rtk_qos_invldDscpVal_get(uint32 unit, uint32 *pDscp);
rtk_qos_invldDscpVal_set(uint32 unit, uint32 dscp);
rtk_qos_portInvldDscpEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_qos_portInvldDscpEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
```

14.1.4 Inner-tag-based Priority Assignment

If packet matches the C → C' conversion(packet has inner-tag, hit C2SC configuration, C2SC_VID_SEL=0,

Realtek CONFIDENTIAL CAMEO Communications Inc.

C2SC_PRI_ASSIGN=1), then the priority C2SC_NEWPRI assigned by C2SC configuration will be used to get the rempped priority, otherwise, the priority of inner-tag is used. If packet doesn't have inner-tag, the Inner-tag-based priority is Null.

API REFERENCE

```
rtk_qos_1pPriRemap_get(uint32 unit, rtk_pri_t dot1p_pri, rtk_pri_t *pInt_pri);
rtk_qos_1pPriRemap_set(uint32 unit, rtk_pri_t dot1p_pri, rtk_pri_t int_pri);
```

14.1.5 Outer-tag-based Priority Assignment

If packet has outer-tag, outer-tag priority is translated to obtain outer-tag-based priority by global Outer-tag Priority Remapping Table. It's different from inner-tag-based remapping configuration that there are two remapping configurations (PRI_SEL_OPRI_DEI0_REMAP register in Table 108 and PRI_SEL_OPRI_DEI0_REMAP register in Table 109) for different DEI value. Otherwise, the outer-tag-based priority is Null for outer-un-tag packets,

Table 108: PRI_SEL_OPRI_DEI0_REMAP Register

Field Name	Bits	Description
INTPRI_OPRI	3	Remap outer-tag priority with DEI 0 to internal priority.

Table 109: PRI_SEL_OPRI_DEI1_REMAP Register

Field Name	Bits	Description
INTPRI_OPRI	3	Remap outer-tag priority with DEI 1 to internal priority.

API REFERENCE

```
rtk_qos_outer1pPriRemap_get(uint32 unit, rtk_pri_t dot1p_pri, uint32 dei, rtk_pri_t *pInt_pri);
rtk_qos_outer1pPriRemap_set(uint32 unit, rtk_pri_t dot1p_pri, uint32 dei, rtk_pri_t int_pri);
```

14.2 Priority Selection Table

A packet may have seven different priorities from port-inner-based, port-outer-based, DSCP-based, Inner-tag-based, Outer-tag-based assignments. One of the five priorities is selected to be unique internal priority for incoming packet according to one of Priority Selection Table. The device supports global 4 Priority Selection Table configurations (PRI_SEL_TBL_CTRL register in Table 110) and per-ingress-port table index configuration (PRI_SEL_PORT_TBL_IDX_CTRL register in Table 111).

Weight value 0~7 can be configured to each priority assignment in Priority Selection Table. Weight 0 stands for ignoring the priority assignment. If all priority assignment weights are be configured to 0, the internal priority produced by Priority Selection Table is 0. Valid weight values are 1~7, and 7 is highest. Highest priority is selected to be the internal priority in priority assignments having same weight values for incoming packet.

Table 110: PRI_SEL_TBL_CTRL Register

Field Name	Bits	Description
OTAG_WT	3	Priority selection weight of Outer-tag-based priority. 3'b000: ignore the priority source 3'b001 ~ 3'b111: weight value 1 ~ 7. 7 is highest weight
ITAG_WT	3	Priority selection weight of Inner-tag-based priority.
DSCP_WT	3	Priority selection weight of DSCP-based priority.

Realtek CONFIDENTIAL Communications, Inc.

PORT_OTAG_WT	3	Priority selection weight of Port-Outer-based-priority.
PORT_ITAG_WT	3	Priority selection weight of Port-Inner-based-priority.

Table 111: PRI_SEL_PORT_TBL_IDX_CTRL Register

Field Name	Bits	Description
TBL_IDX	2	The index of priority selection table for specified port.

Take Figure 7 to be an example. Weights of Port-Inner-based and Port-Outer-based are 0, and the device ignores theirs priority results. Outer-tag-based have highest weight value 7, but its priority results are Null(without Otag). Continue to check other priority assignments. DSCP-based, Inner-tag-based have same weight 6. The highest priority 2 is selected from them to be internal priority.

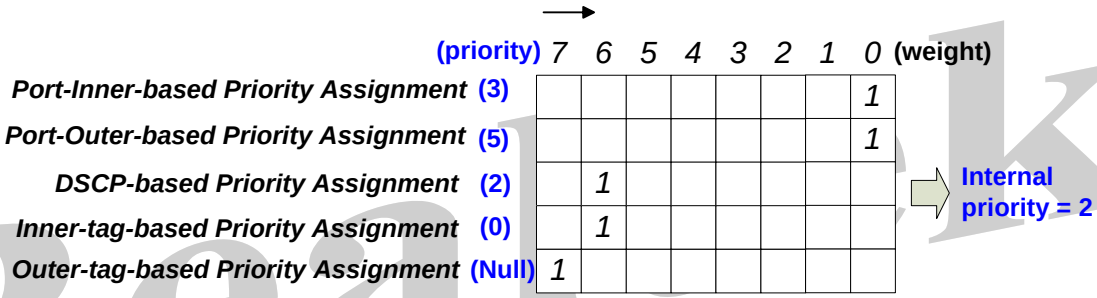


Figure 7: Example for Priority Selection Weight Rule

API REFERENCE

```
rtk_qos_priSelGroup_get(uint32 unit, uint32 grp_idx, rtk_qos_priSelWeight_t *pWeightOfPriSel);
rtk_qos_priSelGroup_set(uint32 unit, uint32 grp_idx, rtk_qos_priSelWeight_t *pWeightOfPriSel);
rtk_qos_portPriSelGroup_get(uint32 unit, rtk_port_t port, uint32 *pPriSelGrp_idx);
rtk_qos_portPriSelGroup_set(uint32 unit, rtk_port_t port, uint32 priSelGrp_idx);
```

PROGRAMMING EXAMPLE

Priority Selection Table Configuration

```
/* Priority selection table configuration */
pri_sel_tbl_idx = 2;

/* Set weight value to each priority assignment */
pri_sel_weight. weight_of_portBasedIpri = 1;
pri_sel_weightl. weight_of_portBasedOpri = 0;
pri_sel_weight.weight_of_dscp = 6;
pri_sel_weight.weight_of_innerTag = 6;
pri_sel_weight.weight_of_outerTag = 7;

/* Assign weight values to selection table */
rtk_qos_priSelGroup_set(unit, pri_sel_tbl_idx, &pri_sel_weight);

/* Set port 0 to use the configured priority selection table */
rtk_qos_portPriSelGroup_set(unit, 0, pri_sel_tbl_idx);
```

Realtek CONFIDENTIAL CAMEO Communications, Inc.

14.3 Internal Prioirty for Packets Fowarded to CPU

Packets forwarded to CPU could be divied into trap/copy or normal forwarding to CPU packet. Trap or copy to CPU has diffent reason which is recorded in CPU Rx Tag REASON field, please refer to the chapter 18.1. For each reason, user could assign the default internal priority with register QM_PKT2CPU_INTPRI_n(n = 0~2). Register QM_PKT2CPU_INTPRI_MAP is for normal forwarding to CPU packet.

Table 112: QM_PKT2CPU_INTPRI_0 Register

Field Name	Bits	Description
CFI	3	Internal priority to the CFI packet that forworded to CPU
IGR_VLAN_FLTR	3	Internal priority to the ingress VLAN filtering packet that forworded to CPU
INVLD	3	Internal priority to the other reason's packet that forworded to CPU. Note: Other reasons as below: 1. NEW_SA (New Source Address) 2. L2_PMV (L2 entry port moving) 3. ATK_HIT (attack prevention) 4. ACL_HIT but ACL not assign priority

Table 113: QM_PKT2CPU_INTPRI_1 Register

Field Name	Bits	Description
RLDP_RLPP	3	Realtek RLDP or RLPP packet
PORT_MAC_CONSTRT	3	Internal priority to the exceeding port-based mac constraint packet that forworded to CPU
VLAN_MAC_CONSTRT	3	Internal priority to the exceeding VLAN-based mac constraint packet that forworded to CPU
SYS_MAC_CONSTRT	3	Internal priority to the exceeding system-based mac constraint packet that forworded to CPU
RMA	3	Internal priority to the RMA packet that forworded to CPU

Table 114: QM_PKT2CPU_INTPRI_2 Register

Field Name	Bits	Description
CPU_MAC	3	Internal priority to the CPU's MAC packet that forworded to CPU
UNKNOWN_DA	3	Internal priority to the unknown DA packet that forworded to CPU
SPECIAL_COPY	3	Internal priority to the special copy packet(ARP/IPV6 Neighbor Discovery) that forworded to CPU
SPECIAL_TRAP	3	Internal priority to the special trap packet(IGMP/MLD/EAPOL) that forworded to CPU
ROUT_EXCEPT	3	Internal priority to the routing exception or next-hop entry age out packet that forworded to CPU

Table 115: QM_PKT2CPU_INTPRI_MAP Register

Field Name	Bits	Description
PRI7	3	Internal priority to the packets with priority 7 that normal forworded to CPU
PRI6	3	Internal priority to the packets with priority 6 that normal forworded to CPU
PRI5	3	Internal priority to the packets with priority 5 that normal forworded to CPU
PRI4	3	Internal priority to the packets with priority 4 that normal forworded to

Realtek CONFIDENTIAL CAMEO Communications Inc.

		CPU
PRI3	3	Internal priority to the packets with priority 3 that normal forworded to CPU
PRI2	3	Internal priority to the packets with priority 2 that normal forworded to CPU
PRI1	3	Internal priority to the packets with priority 1 that normal forworded to CPU
PRI0	3	Internal priority to the packets with priority 0 that normal forworded to CPU

API REFERENCE

```
rtk_trap_mgmtFramePri_get(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_pri_t *pPriority);
rtk_trap_mgmtFramePri_set(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_pri_t priority);
rtk_qos_pkt2CpuPriRemap_get(uint32 unit, rtk_pri_t intPri, rtk_pri_t *pNewPri)
rtk_qos_pkt2CpuPriRemap_set(uint32 unit, rtk_pri_t intPri, rtk_pri_t newPri)
```

PROGRAMMING EXAMPLE

to CPU port packet Priority Configuration

```
/* assign BPDU packet priority which is forwarded to CPU */
priority = 3;
rtk_trap_mgmtFramePri_set(unit, MGMT_TYPE_BPDU, priority);

/* assign LACP packet priority which is forwarded to CPU */
priority = 5;
rtk_trap_mgmtFramePri_set(unit, MGMT_TYPE_LACP, priority)

/* remap normal forwarding to cpu packet priority*/
intpri = 0;
newpri = 0;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);

intpri = 1;
newpri = 0;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);

intpri = 2;
newpri = 0;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);

intpri = 3;
newpri = 0;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);


intpri = 4;
newpri = 1;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);

intpri = 5;
newpri = 1;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);
```

```
intpri = 6;
newpri = 1;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);

intpri = 7;
newpri = 1;
rtk_qos_pkt2CpuPriRemap_set(unit, intpri, newpri);
```

14.4 Internal Priority to QID Mapping

The device supports 8 egress queues to each port (including CPU port) . However, the egress queue is treated as traffic class. Each priority can be configured to map a traffic class by global QM_INTPRI2QID_CTRL register as Table 116.

Table 116: QM_INTPRI2QID_CTRL Register

Field Name	Bits	Description
INTPRI_QID	3	Internal priority mapping to egress queue ID 3'b000: queue ID 0 3'b001: queue ID 1 3'b010: queue ID 2 3'b011: queue ID 3 3'b100: queue ID 4 3'b101: queue ID 5 3'b110: queue ID 6 3'b111: queue ID 7

API REFERENCE

```
rtk_qos_priMap_get(uint32 unit, uint32 queue_num, rtk_qos_pri2queue_t *pPri2qid);
rtk_qos_priMap_set(uint32 unit, uint32 queue_num, rtk_qos_pri2queue_t *pPri2qid);
```

PROGRAMMING EXAMPLE

Configure System Traffic Class Number and Internal Priority to QID Mapping

```
/* Traffic class configuration */
queue_num = 8;

/* Priority to QID mapping configurations */
pri2qid.pri2queue[0] = 0;
pri2qid.pri2queue[1] = 0;
pri2qid.pri2queue[2] = 1;
pri2qid.pri2queue[3] = 1;
pri2qid.pri2queue[4] = 2;
pri2qid.pri2queue[5] = 2;
pri2qid.pri2queue[6] = 3;
pri2qid.pri2queue[7] = 3;
```

Realtek CONFIDENTIAL CAMEO Communications Inc.

```
/* Configure internal priority to QID mapping */
rtk_qos_priMap_set(unit, queue_num, &pri2qid);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

15 Egress Shaping

Packet scheduler architecture is supported to each egress port. It can used to manage egress port bandwidth, traffic class bandwidth, select algorithm type in byte-count or packet-count and ratio configurations between traffic classes, etc. In traffic class bandwidth level, the device provides fixed bandwidth feature to reserved private bandwidth of a specified traffic class.

15.1 Scheduler Architecture

Egress bandwidth shaping is implemented in per-egress port scheduling module as Figure 8. The egress port rate is configured by SCHED_LB_CTRL.P_EGR_LB_EN and it's implemented in a leaky bucket (LB). For each traffic class queue, the scheduling supports two level LB. One is average packet rate (APR) LB used to control queue bandwidth. The other is used in weighted faired queue (WFQ) algorithm.

The device supports two algorithms: WFQ in byte-based and weighted round-robin (WRR) in packet-based. The algorithm is configured by SCHED_P_TYPE_CTRL.SCHED_TYPE register. Besides, the device provides WEIGHT_Qn register to configure weighted value for each traffic class queue. In WFQ mode, system shares port rate (EGR_RATE) to all WFQ LBs according to queue weighted value. In WRR, inside transmitting packet counter is instead of WFQ LB.

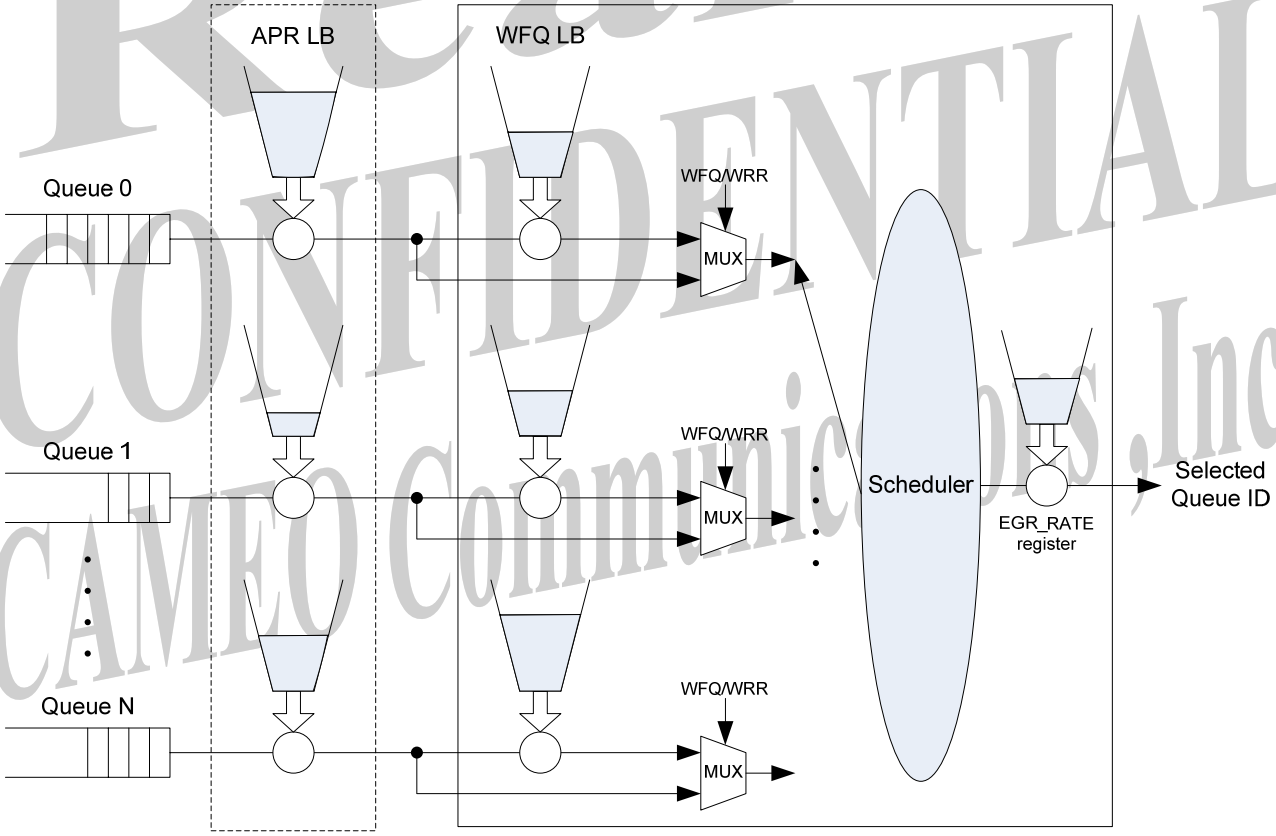


Figure 8: Per-egress-port Packet Scheduler

In the LB as Figure 9, there are high and low thresholds. High threshold is used in transmitting packets checking, and low threshold is used in subtract token checking. If token in LB is less than high threshold, packets can be transmitted. When packets pass through, token which size is same as the passing packet is added into the bucket. On the side, LB token is subtracted by setting rate until the token is less than low threshold. The device supports

Realtek CONFIDENTIAL CAMEO Communications, Inc.

global SCHED_LB_THR.APR_LB_SIZE register and SCHED_LB_THR.WFQ_LB_SIZE register to set high thresholds of APR LB and WFQ LB. However, the low thresholds of APR LB and WFQ LB is fixed in 0 Byte and it can't be modulated.

Besides, the device supports a global SCHED_CTRL.INC_IFG register to compute inter-frame gap (IFG) or not. In including IFG configuration, 20Bytes token is also added to each LB when packet transmits. In excluding IFG configuration, only packet size token is added to LB.

System supports three-level scheduler. The priority increase from Level 1 to level 3. The high level can preempt the low level's bandwidth. Level 3 are Pure Strict Priority queues, Level 2 are Strict Priority+WFQ/WRR queues, and Level 1 are WFQ/WRR queues.

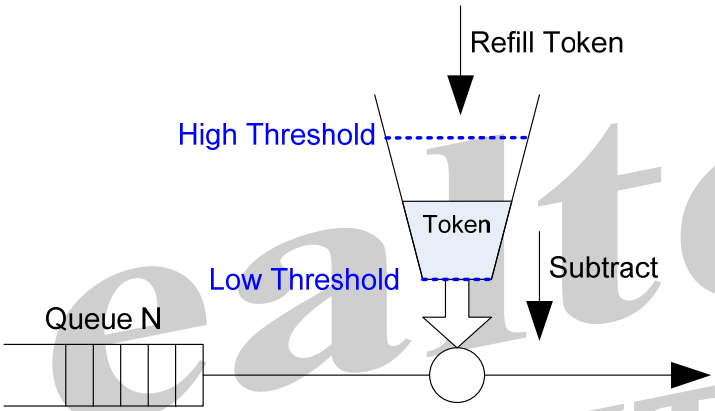


Figure 9: Leaky Bucket Architecture

Table 117: SCHED_P_TYPE_CTRL Register

Field Name	Bits	Description
SCHED_TYPE	1	Scheduling type selection WFQ or WRR for Port n. 0b0: Weighted Fair-Queuing (WFQ) 0b1: Weighted Round-Robin (WRR)

Table 118: SCHED_ CTRL Register

Field Name	Bits	Description
INC_IFG	1	Ingress/Egress port bandwidth control includes/excludes the IFG(20Bytes). 1'b0: exclude 1'b1: include

Table 119: SCHED_LB_THR Register

Field Name	Bits	Description
APR_LB_SIZE	16	High threshold(bucket size) of APR LB in bytes for all ports. Unit: Byte
WFQ_LB_SIZE	16	High threshold(bucket size) of WFQ LB in bytes for all ports Unit: Byte

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_qos_schedulingAlgorithm_get(uint32 unit, rtk_port_t port, rtk_qos_scheduling_type_t *pScheduling_type);
rtk_qos_schedulingAlgorithm_set(uint32 unit, rtk_port_t port, rtk_qos_scheduling_type_t scheduling_type);
rtk_rate_egrBandwidthCtrlIncludeIfg_get(uint32 unit, rtk_enable_t *pIfg_include);
rtk_rate_egrBandwidthCtrlIncludeIfg_set(uint32 unit, rtk_enable_t ifg_include);
```

PROGRAMMING EXAMPLE

Scheduling Algorithm Configuration

```
/* Scheduling algorithm configuration */
sche_algo = WFQ;

/* Set scheduling algorithm of port 3 */
rtk_qos_schedulingAlgorithm_set(unit, 3, sche_algo);
```

15.2 Egress Port Bandwidth Management

For each port, the device supports egress bandwidth configuration (SCHED_P_EGR_RATE_CTRL. P_EGR_RATE). Not only bandwidth configuration for egress port, but also weight configuration (SCHED_Q_CTRL.Q_WEIGHT) is supported for egress traffic class. In WRR, the ratio of transmitting packets number can be controlled by the weight configuration. In WFQ, each queue can get the expected bandwidth by the weight configuration. For example, a port configured in WFQ has 500Mbps egress bandwidth and 4 egress queues Q0, Q1, Q2 and Q3, and wire speed packets are sent to it. The weights of 4 queues are 1:2:3:4. The transmitting rates of Q0, Q1, Q2 and Q3 are 50Mbps, 100Mbps, 150Mbps and 200Mbps, respectively.

Table 120: SCHED_LB_CTRL Register

Field Name	Bits	Description
P_EGR_LB_EN	1	Egress bandwidth control ability for output port . 0x0: disable 0x1: enable Note: if leaky bucket is disabled, there is no rate limit ability and interal leaky bucket token counter will be cleared by asic. if WFQ is selected, it also should be enabled.

Table 121: SCHED_P_EGR_RATE_CTRL Register

Field Name	Bits	Description
P_EGR_RATE	18	Total egress bandwidth of WFQ or WRR for port n. Unit: 16 Kbps 0: blocking Note: LB rate = 1 means 16Kbps(16000bps), port 24/26 is 2.5G port, other port is 1G port, for 1G port, LB rate >= 0xF424 means 1Gbps, for 2.5G port, LB rate >= 0x2625A means 2.5 Gbps.For port 28, if SCHED_CTRL. P28_LB_MODE is set as 1, the unit will be 1 pps not 16Kbps.

Table 122: SCHED_Q_CTRL Register

Field Name	Bits	Description
Q_WEIGHT	7	Weight value assign of port n queue m for WFQ/WRR Note: value 1~127 implies weight 1~127，value 0 combined with PnQm_STRICT_EN implies pure strict priority

Realtek CONFIDENTIAL CAMEO Communications, Inc.

	(PnQm_STRICT_EN , PnQm_WEIGHT) (1, 0) : Pure SP queue (1, 1~127) : SP + WFQ/WRR queue (0, 0~127) : WFQ/WRR queue

API REFERENCE

```
rtk_qos_schedulingQueue_get(uint32 unit, rtk_port_t port, rtk_qos_queue_weights_t *pQweights);
rtk_qos_schedulingQueue_set(uint32 unit, rtk_port_t port, rtk_qos_queue_weights_t *pQweights);
rtk_qos_queueStrictEnable_get(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t *pEnable);
rtk_qos_queueStrictEnable_set(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t enable);
rtk_rate_egrBandwidthCtrlEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_rate_egrBandwidthCtrlEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
rtk_rate_egrBandwidthCtrlRate_get(uint32 unit, rtk_port_t port, uint32 *pRate);
rtk_rate_egrBandwidthCtrlRate_set(uint32 unit, rtk_port_t port, uint32 rate);
```

PROGRAMMING EXAMPLE

Configure Traffic Class Weight and Egress Bandwidth to the Specified Port

```
/* Queue weight configuration */
queue_weight.weights[0] = 1; /* queue 0 = weight 1 */
queue_weight.weights[1] = 2; /* queue 1 = weight 2 */
queue_weight.weights[2] = 3; /* queue 2 = weight 3 */
queue_weight.weights[3] = 4; /* queue 3 = weight 4 */
queue_weight.weights[4] = 5; /* queue 4 = weight 5 */
queue_weight.weights[5] = 6; /* queue 5 = weight 6 */
queue_weight.weights[6] = 0; /* queue 6 = pure strict priority */
queue_weight.weights[7] = 0; /* queue 7 = pure strict priority */

/* Bandwidth configuration */
rate = 100; /* unit = 16Kbps, rate = 100*16Kbps */

/* Set weight value to port 1 */
port = 1;
rtk_qos_schedulingQueue_set(unit, port, &queue_weight);

/* Set bandwidth configuration to port 1 */
rtk_rate_egrBandwidthCtrlEnable_set(unit, port, ENABLED);
rtk_rate_egrBandwidthCtrlRate_set(unit, port, rate);

/*Set queue6 queue7 pure strict priority schedule*/
rtk_qos_queueStrictEnable_set(nit, 1, 6, ENABLED);
rtk_qos_queueStrictEnable_set(nit, 1, 7, ENABLED);
```

15.3 Egress Port Bandwidth Management in CPU Port

Not only byte per second (BPS) mode but also packet per second (PPS) mode are supported to CPU port due to packet unit is often used in CPU process. System provides SCHED_CTRL.P28_LB_MODE register in Table 123 to select bandwidth process mode for CPU port. The configuration applies to both egress port rate and egress queue rate of CPU port. In BPS mode, it can refer normal port configuration and normal queue configuration. In PPS mode, the definitions of egress port rate and egress queue rate are in step of 1 pps instead of 16 Kbps. When PPS mode is used for bandwidth management, SCHED_TYPE of port 28 must be set as WRR instead of WFQ, else it does not work.

Table 123: SCHED_CTRL Register

Field Name	Bits	Description
P28_LB_MODE	1	Port 28 egress port and queue bandwidth control based on packet count or byte count 0b0: bps 0b1: pps

API REFERENCE

```
rtk_rate_egrBandwidthCtrlCPURateMode_get(uint32 unit, rtk_rate_rateMode_t *pRate_mode);
rtk_rate_egrBandwidthCtrlCPURateMode_set(uint32 unit, rtk_rate_rateMode_t rate_mode);
```

PROGRAMMING EXAMPLE

Configure Egress Bandwidth Counting Mode of CPU Port

```
/* Bandwidth control counting mode */
rate_mode = RATE_MODE_PKT; /* unit : packet */

/* Set egress bandwidth counting mode of CPU port */
rtk_rate_egrBandwidthCtrlCPURateMode_set(unit, rate_mode);
```

15.4 Egress Queue Bandwidth Management

Not only egress port bandwidth configuration but also egress queue bandwidth configuration (SCHED_LB_CTRL.Qn_EGR_LB_APR_EN) is supported by the device. Please pay attention, the egress queue bandwidth configuration is different to the fixed bandwidth function of WFQ LB described in section 15.5. The egress queue bandwidth configuration is implemented in APR LB.

Suppose egress queue bandwidth of Q3 is set to 140Mbps in the same example described in section 15.2. The transmitting rates of Q0, Q1, Q2 and Q3 will be 60Mbps, 120Mbps, 180Mbps and 140Mbps, respectively. However, if egress queue bandwidth of Q3 is configured to over 200Mbps, the transmitting rates of 4 queues are keeping in 50Mbps, 100Mbps, 150Mbps and 200Mbps due to Q3 is limited by weight value 4 in WFQ module.

Table 124: SCHED_LB_CTRL Register

Field Name	Bits	Description
Qn_EGR_LB_APR_EN	1	APR LB control for output queue 0. 0x0: disable 0x1: enable Note: if leaky bucket is disable, there is no rate limit ability and interal leaky bucket token counter will be cleared by asic.

Table 125: SCHED_Q_EGR_RATE_CTRL Register

Field Name	Bits	Description
Q_EGR_RATE	18	Average Packet Rate of APR LB in times of 16 Kbps for port n output queue m. Unit: 16 Kbps 0: blocking Note: LB rate = 1 means 16Kbps(16000bps), port 24/26 is 2.5G port, other port is 1G port, for 1G port, LB rate >= 0xF424 means 1Gbps, for

CONFIDENTIAL
CAMEO Communications, Inc.

	2.5G port, LB rate >= 0x2625A means 2.5 Gbps.For port 28, if SCHED_CTRL. P28_LB_MODE is set as 1,the unit will be 1 pps not 16Kbps.

API REFERENCE

```
rtk_rate_egrQueueBwCtrlEnable_get(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t *pEnable);
rtk_rate_egrQueueBwCtrlEnable_set(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t enable);
rtk_rate_egrQueueBwCtrlRate_get(uint32 unit, rtk_port_t port, rtk_qid_t queue, uint32 *pRate);
rtk_rate_egrQueueBwCtrlRate_set(uint32 unit, rtk_port_t port, rtk_qid_t queue, uint32 rate);
```

PROGRAMMING EXAMPLE

Configure Traffic Class Bandwidth to the Specified Port

```
/* Bandwidth configuration */
rate = 100; /* unit = 16Kbps, rate = 100*16Kbps */

/* Set bandwidth configuration to queue 7of port 1 */
rtk_rate_egrQueueBwCtrlEnable_set(unit, 1, 7, ENABLED);
rtk_rate_egrQueueBwCtrlRate_set(unit, 1, 7, rate);
```

15.5 Egress Queue Fixed Bandwidth Mechanism

The device supports reserved bandwidth function to each traffic class. The capability is used to retain private bandwidth for the specified traffic class, and it can be configured by SCHED_Q_CTRL.Q_BORW_TKN. If a traffic class which Q_BORW_TKN is 0b1 doesn't transmit packets, its unused bandwidth is distributed to other traffic classes. Take the same example described in section 15.2. No traffic is transmitted by Q3 and Q3_BORW_TKN is 1, the rates of Q0, Q1 and Q2 are nearly 83Mbps, 166Mbps and 249Mbps, respectively. Contrariwise, the traffic class' Q3_BORW_TKN is 0, the unused bandwidth is reserved. The rates of Q0, Q1, Q2 and Q3 of the same example are 50Mbps, 100Mbps, 150Mbps and 0bps, respectively.

Table 126: SCHED_Q_CTRL Register

Field Name	Bits	Description
Q_BORW_TKN	1	Borrow WFQ remaining tokens of queue m to other queues in port n 0b0: doesnt borrow tokens 0b1: borrow tokens

API REFERENCE

```
rtk_rate_egrQueueFixedBandwidthEnable_get(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t *pEnable);
rtk_rate_eqrQueueFixedBandwidthEnable_set(uint32 unit, rtk_port_t port, rtk_qid_t queue, rtk_enable_t enable);
```

PROGRAMMING EXAMPLE

Configure Fix Bandwidth Function of the Specified Traffic Class

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
/* Configure fix bandwidth capability to queue 3 of port 1 */
rtk_rate_eqrQueueFixedBandwidthEnable_set(unit, 1, 3, ENABLED);
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

16 Egress Remarking

Network devices usually apply some QoS attributes to decide traffic forwarding speed. Sometimes, traffic transmitting speed is reduced or increased by the attributes remarking. The section describes QoS remarking module. In the module, packet with QoS attributes as below may be remarked or preserved before packets are actually going to be sent out:

- Inner-tag priority
- Outer-tag priority
- IPv4 and IPv6 DSCP

System provides individual enable/disable per-egress-port configuration as Table 62 for each QoS attributes. In Inner-tag priority and outer-tag priority, since packet hits one of inner-tag or outer-tag TPID configurations described in VLAN module, the packet is treated as having inner-tag priority or outer-tag priority.

Table 127: RMK_PORT_RMK_EN_CTRL Register

Field Name	Bits	Description
IPRI_RMK_EN	1	Enable/Disable inner-priority remarking for a port. 1'b0: disable 1'b1: enable
OPRI_RMK_EN	1	Enable/Disable outer-tag priority remarking for a port. 1'b0: disable 1'b1: enable
DSCP_RMK_EN	1	Enable/Disable DSCP remarking for a port. 1'b0: disable 1'b1: enable

API REFERENCE

```
rtk_qos_1pRemarkEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_qos_1pRemarkEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
rtk_qos_out1pRemarkEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_qos_out1pRemarkEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
rtk_qos_dscpRemarkEnable_get(uint32 unit, rtk_port_t port, rtk_enable_t *pEnable);
rtk_qos_dscpRemarkEnable_set(uint32 unit, rtk_port_t port, rtk_enable_t enable);
```

16.1 Inner-tag Priority Remarking

The inner-tag priority remarking module is not only used to remark inner-tag priority but also used to decide an inner-tag priority to inner-untag packet. The two sections are discussed as following.

16.1.1 Inner-tag Priority Remarking Behavior

System provides a global register RMK_CTRL.IPRI_RMK_SRC in Table 128 for selecting inner-tag priority remarking source. Besides, there is a global inner-tag remarking table (RMK_IPRI_CTRL in Table 129) for priority remarking setting. There are overall 2 inner-tag priority remarking sources: internal-priority and original inner-tag priority. Two remarking sources, internal-priority and original inner-tag priority, share the global inner-tag priority remarking table to configure remarking value. The table index can refer Figure 10.

Table 128: RMK_CTRL Register

Field Name	Bits	Description
IPRI_RMK_SRC	1	Select inner-priority remarking source. If original packet doesn't have inner-tag, system takes the register value, IPRI_RMK_DFLT_PRI, to be its inner-priority.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

	1'b0: internal priority 1'b1: original inner-priority

Table 129: RMK_IPRI_CTRL Register

Field Name	Bits	Description
IPRI	3	New inner-tag priority for an internal-priority, or original inner-priority according to IPRI_RMK_SRC setting.

Table Index	Remarked Value
Inner-priority 7/Internal-priority 7	Inner-priority
Inner-priority 6/Internal-priority 6	Inner-priority
Inner-priority 5/Internal-priority 5	Inner-priority
Inner-priority 4/Internal-priority 4	Inner-priority
Inner-priority 3/Internal-priority 3	Inner-priority
Inner-priority 2/Internal-priority 2	Inner-priority
Inner-priority 1/Internal-priority 1	Inner-priority
Inner-priority 0/Internal-priority 0	Inner-priority

Figure 10: Table Index Chart of Inner-tag Priority Remarking Table

API REFERENCE

```
rtk_qos_1pRemarkSrcSel_get(uint32 unit, rtk_qos_1pRmkSrc_t *pType);
rtk_qos_1pRemarkSrcSel_set(uint32 unit, rtk_qos_1pRmkSrc_t type);
rtk_qos_1pRemark_get(uint32 unit, rtk_pri_t int_pri, rtk_pri_t *pDot1p_pri);
rtk_qos_1pRemark_set(uint32 unit, rtk_pri_t int_pri, rtk_pri_t dot1p_pri);
```

PROGRAMMING EXAMPLE

Inner-tag Priority Remarking Source and Remarking Value Configurations

```
/* Set Internal-priority to be inner-tag priority remarking source */
rmk_src = DOT_1P_RMK_SRC_INT_PRI;
rtk_qos_1pRemarkSrcSel_set(unit, rmk_src);

/* Configure Internal-priority to inner-tag priority remarking table value */
for (i=0; i<7; i++) {
    rtk_qos_1pRemark_set(unit, i, 3);
}
```

16.1.2 Default Inner-tag Priority Assignment

Device provides global 2-bit register RMK_CTRL.IPRI_DFLT_SRC defined in Table 130 to decide TX inner-tag priority for inner-untag packet. Totally there are three methods, using global inner-tag priority value, copy RX port-based priority, and copy internal-priority, for customize setting. In using global inner-tag priority value, system takes global register RMK_CTRL.IPRI_DFLT_PRI to be TX inner-tag priority.

Table 130: RMK_CTRL Register

Field Name	Bits	Description
IPRI_DFLT_SRC	1	Default inner-priority configuration for inner-un-tag packet. 1'b0: use IPRI_DFLT_PRI 1'b1: copy RX port-based inner tag priority
IPRI_DFLT_PRI	3	Inner-tag priority default value.
IPRI_DFLT_INTPRI	1	Inner-priority default use internal priority as default priority source, ignore *IPRI_DFLT_SRC* configuration. 1'b0: Inner default priority follow IPRI_DFLT_SRC setting 1'b1: Inner default priority use internal priority

In another case, inner-tag priority remarking of egress port is enabled and remarking source configured with original inner-tag priority. Since the packet is inner-untag, system also follows IPRI_DFLT_INTPRI and IPRI_DFLT_SRC configurations to decide TX inner-tag priority.

API REFERENCE

```
rtk_qos_1pDfltPriSrcSel_get(uint32 unit, rtk_qos_1pDfltPriSrc_t *pType);
rtk_qos_1pDfltPriSrcSel_set(uint32 unit, rtk_qos_1pDfltPriSrc_t type);
rtk_qos_1pDfltPri_get(uint32 unit, rtk_pri_t *pDot1p_pri);
rtk_qos_1pDfltPri_set(uint32 unit, rtk_pri_t dot1p_pri);
```

16.2 Outer-tag Priority Remarking

In outer-tag priority remarking module, system provides two sections: remarking outer-tag priority and deciding outer-tag priority for outer-untag packet. Following are the details.

16.2.1 Outer-tag Priority Remarking Behavior

Outer-tag priority remarking is similarly inner-tag priority remarking. System also provides a global register RMK_CTRL.OPRI_RMK_SRC in Table 131 for selecting outer-tag priority remarking source: internal-priority and original outer-priority. Three remarking sources internal-priority and original outer-priority use global outer-tag priority remarking table defined in register RMK_OPRI_CTRL as Table 132 to configure remarking value. Figure 11 is the table index charts of the table.

Table 131: RMK_PORT_OPRI_SRC_CTRL Register

Field Name	Bits	Description
OPRI_RMK_SRC	1	Select outer-priority remarking source. 1'b0: internal priority 1'b1: original outer-priority

Table 132: RMK_OPRI_CTRL Register

Field Name	Bits	Description
OPRI	3	New outer-tag priority for an internal-priority, or original inner-priority according to OPRI_RMK_SRC setting.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

Table Index	Remarked Value
Outer-priority 7/Internal-priority 7	Outer-priority
Outer-priority 6/Internal-priority 6	Outer-priority
Outer-priority 5/Internal-priority 5	Outer-priority
Outer-priority 4/Internal-priority 4	Outer-priority
Outer-priority 3/Internal-priority 3	Outer-priority
Outer-priority 2/Internal-priority 2	Outer-priority
Outer-priority 1/Internal-priority 1	Outer-priority
Outer-priority 0/Internal-priority 0	Outer-priority

Figure 11: Table Index Chart of Outer-tag Priority Remarking Table

API REFERENCE

```
rtk_qos_portOuter1pRemarkSrcSel_get(uint32 unit, rtk_port_t port, rtk_qos_outer1pRmkSrc_t *pType);
rtk_qos_portOuter1pRemarkSrcSel_set(uint32 unit, rtk_port_t port, rtk_qos_outer1pRmkSrc_t type);
rtk_qos_out1pRemark_get(uint32 unit, rtk_pri_t int_pri, rtk_pri_t *pDot1p_pri) ;
rtk_qos_out1pRemark_set(uint32 unit, rtk_pri_t int_pri, rtk_pri_t dot1p_pri) ;
```

PROGRAMMING EXAMPLE

Outer-tag Priority Remarking Source and Remarking Value Configurations

```
/* Set internal-priority to be outer-tag priority remarking source */
rmk_src = OUTER_1P_RMK_SRC_INT_PRI;
rtk_qos_portOuter1pRemarkSrcSel_set(unit, rmk_src);

/* Configure internal-priority to outer-tag priority remarking table value */
for (i=0; i<8; i++) {
    rtk_qos_out1pRemark_set(unit, i, 7);
}
```

16.2.2 Default Outer-tag Priority Assignment

For outer-untag packet, the device also provides outer-tag priority configuration as supporting inner-untag packet. But it's different that the configuration is per-port supporting. The device follows global RMK_CTRL.OPRI_DFLT_CFG register as Table 133 to take configuration of ingress port or egress port. Besides, per-port configuration in RMK_PORT_OPRI_SRC_CTRL register as Table 134 has four selections: copy inner-tag priority, copy RX port-based priority, copy internal priority, and using global default outer-priority value. The last selection, using global default outer-priority value, is configured by register RMK_CTRL.OPRI_DFLT_PRI.

Similarly, if outer-tag priority remarking of egress port is enabled and remarking source is configured with original outer-tag priority, system takes the outer-priority produced by outer-tag priority module described above to be TX outer-priority for outer-untag.

Table 133: RMK_CTRL Register

Field Name	Bits	Description
OPRI_DFLT_PRI	3	Outer-tag priority default value.
OPRI_DFLT_CFG	1	Select configurations, OPRI_DFLT_SRC, of ingress port or egress port

	to decide outer-tag priority. 1'b0: use RX port configuration 1'b1: use TX port configuration

Table 134: RMK_PORT_OPRI_SRC_CTRL Register

Field Name	Bits	Description
OPRI_DFLT_SRC	2	Select default outer-priority source. 2'b00: copy from inner-priority. If original packet without inner-priority, system takes RX port-based outer priority. 2'b01: copy from port-based outer tag priority. 2'b10: use OPRI_DFLT_PRI. 2'b11: copy from internal priority.

API REFERENCE

```
rtk_qos_outer1pDfltPri_get(uint32 unit, rtk_pri_t *pDot1p_pri);
rtk_qos_outer1pDfltPri_set(uint32 unit, rtk_pri_t dot1p_pri);
rtk_qos_outer1pDfltPriCfgSrcSel_get(uint32 unit, rtk_qos_outer1pDfltCfgSrc_t *pDflt_sel);
rtk_qos_outer1pDfltPriCfgSrcSel_set(uint32 unit, rtk_qos_outer1pDfltCfgSrc_t dflt_sel);
rtk_qos_portOuter1pDfltPriSrcSel_get(uint32 unit, rtk_port_t port, rtk_qos_outer1pDfltSrc_t *pType);
rtk_qos_portOuter1pDfltPriSrcSel_set(uint32 unit, rtk_port_t port, rtk_qos_outer1pDfltSrc_t type);
```

PROGRAMMING EXAMPLE

Default Outer-tag Priority Source and Value Configurations

```
/* Set ingress port to be default outer-tag priority referring port */
dflt_port_sel = OUTER_1P_DFLT_CFG_SRC_INGRESS;
rtk_qos_outer1pDfltPriCfgSrcSel_set(unit, dflt_port_sel);

/* Configure default outer-tag priority value */
rtk_qos_outer1pDfltPri_set(unit, 3);
```

16.3 DSCP Remarking

DSCP is the filed in header of IP packet, and it has different levels of service for network traffic. As inner-tag priority remarking and outer-tag priority remarking, DSCP remarking has a global registers RMK_CTRL.DSCP_RMK_SRC in Table 135 for decide remarking source: internal-priority and original DSCP.

Table 135: RMK_CTRL Register

Field Name	Bits	Description
DSCP_RMK_SRC	1	Select DSCP remarking source. 1'b0: internal priority 1'b1: original DSCP

Table 136: RMK_DSCP_CTRL Register

Field Name	Bits	Description
DSCP	3	New DSCP value for a specified DSCP value, inner-priority, outer-priority or internal-priority.

Similarly, system provides a global register RMK_DSCP_CTRL in Table 136 to be DSCP remarking table for two of

DSCP remarking sources. Please pay attention that the first 8 fields of the table index, DSCP 0~DSCP 7, can be considered as internal priority 0~7 as Figure 12 in different remarking source.

Table Index	Remarked DSCP
DSCP 63	DSCP Value
DSCP 62	DSCP Value
……	……
DSCP 9	DSCP Value
DSCP 8	DSCP Value
DSCP 7/Internal-priority 7	DSCP Value
DSCP 6/Internal-priority 6	DSCP Value
DSCP 5/Internal-priority 5	DSCP Value
DSCP 4/Internal-priority 4	DSCP Value
DSCP 3/Internal-priority 3	DSCP Value
DSCP 2/Internal-priority 2	DSCP Value
DSCP 1/Internal-priority 1	DSCP Value
DSCP 0/Internal-priority 0	DSCP Value

Figure 12: Table Index Chart of DSCP Remarking Table

The device can remark DSCP value in IPv4 and IPv6 packet due to Traffic Class filed in IPv6 header is an 8-bit field equal to the type of services (TOS) byte in IPv4 header. Figure 13 and Figure 14 are the header formats of IPv4 and IPv6. If DSCP remarking enabled, the 6-bit in the byte would be modified no matter what the packet is IPv4 or IPv6.

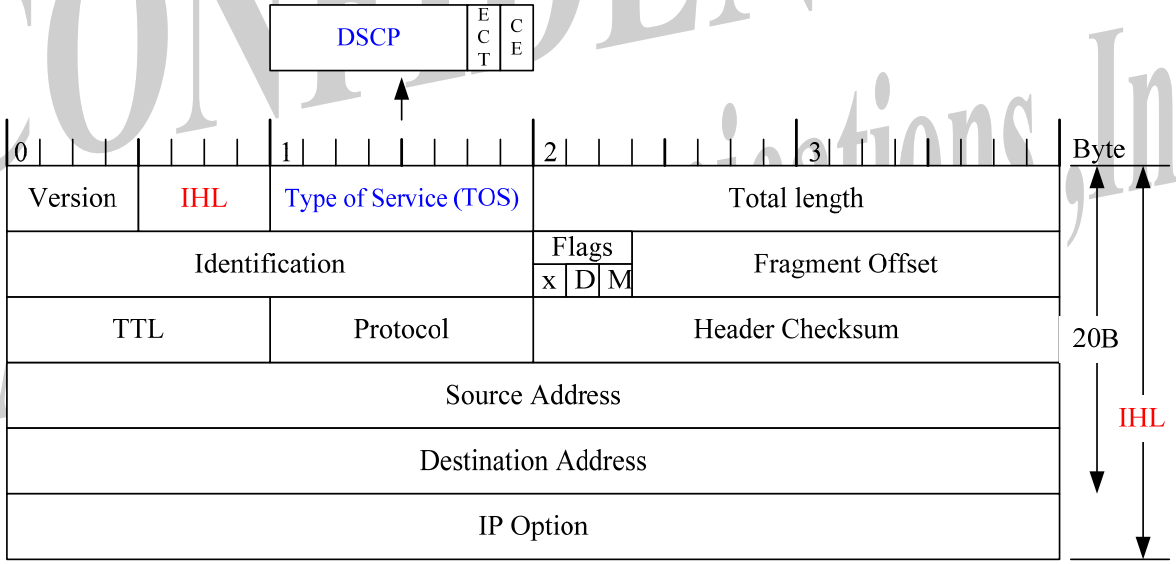


Figure 13: ToS field in IPv4 header format

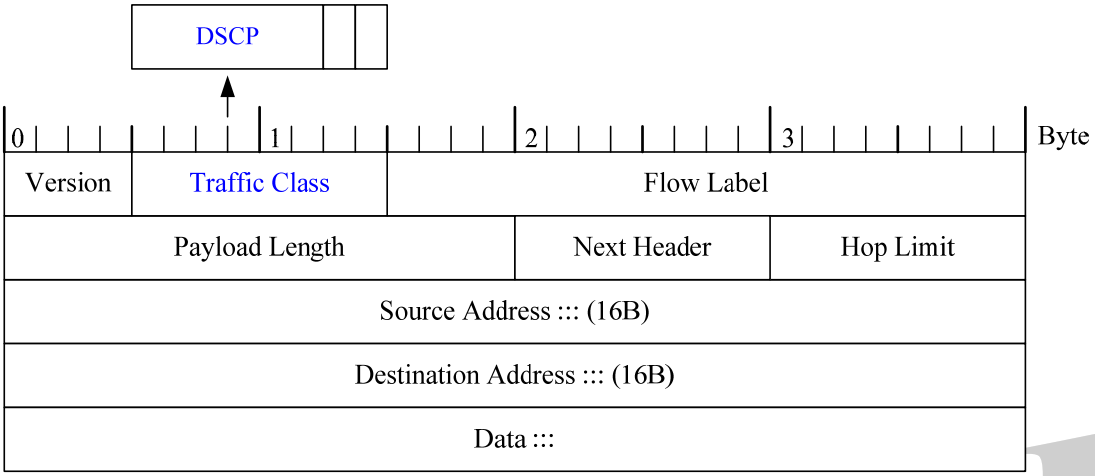


Figure 14: Traffic Class field in IPv6 header format

API REFERENCE

```
rtk_qos_dscpRemarkSrcSel_get(uint32 unit, rtk_qos_dscpRmkSrc_t *pType);
rtk_qos_dscpRemarkSrcSel_set(uint32 unit, rtk_qos_dscpRmkSrc_t type);
rtk_qos_dscpRemark_get(uint32 unit, rtk_pri_t int_pri, uint32 *pDscp);
rtk_qos_dscpRemark_set(uint32 unit, rtk_pri_t int_pri, uint32 dscp);
```

PROGRAMMING EXAMPLE

DSCP Remarking Remarking Source and DSCP Remarking Value Configuration

```
/* Set outer-tag to be tag target of DEI remarking for port 0 */
rmkSrc = DSCP_RMK_SRC_DSCP;
rtk_qos_dscpRemarkSrcSel_set(unit, rmkSrc);

/* Configure DSCP remarking table value */
for (i=0; i<64; i++) {
    rtk_qos_dscpRemark_set(unit, i, 63);
}
```

Realtek CONFIDENTIAL CAMEO Communications

17 NIC (Network Interface Controller)

NIC connected with CPU MAC (MAC 28) is used for receiving/transmitting packets from/to CPU. NIC is a DMA engine which is in charge of moving packet between CPU (Memory) and Switch Core.

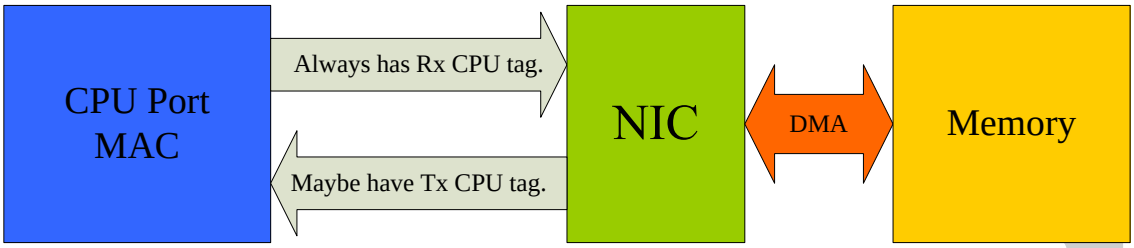


Figure 16: NIC and CPU MAC functional blocks

17.1 Packet Descriptor

The descriptor (drv_nic_pkt_t) below in SDK NIC driver is used to describe a packet:

```
FilePath: system/include/drv/nic/common.h

typedef struct drv_nic_pkt_s
{
    uint8   *head;          /* pointer to the head of the packet data buffer */
    uint8   *data;          /* pointer to the base address of the packet */
    uint8   *tail;          /* pointer to the end address of the packet */
    uint8   *end;           /* pointer to the end of the packet data buffer */
    uint32 length;          /* packet length when the packet is valid (not a empty data buffer) */
    void    *buf_id;        /* pointer to the user-defined packet descriptor */

    uint8   as_txtag;     /* 0: without tx-tag, 1: with tx-tag */
    union
    {
        /* Reception information */
        struct
        {
            ... (ignored, refer to CPU tag) ...
        } rx_tag;

        /* Transmission information */
        struct
        {
            ... (ignored, refer to CPU tag) ...
        } tx_tag;
    };

    struct drv_nic_pkt_s *next;             /* pointer to next packet struct if exists */
} drv_nic_pkt_t;
```

Below figure demonstrates how a jumbo packet is described:

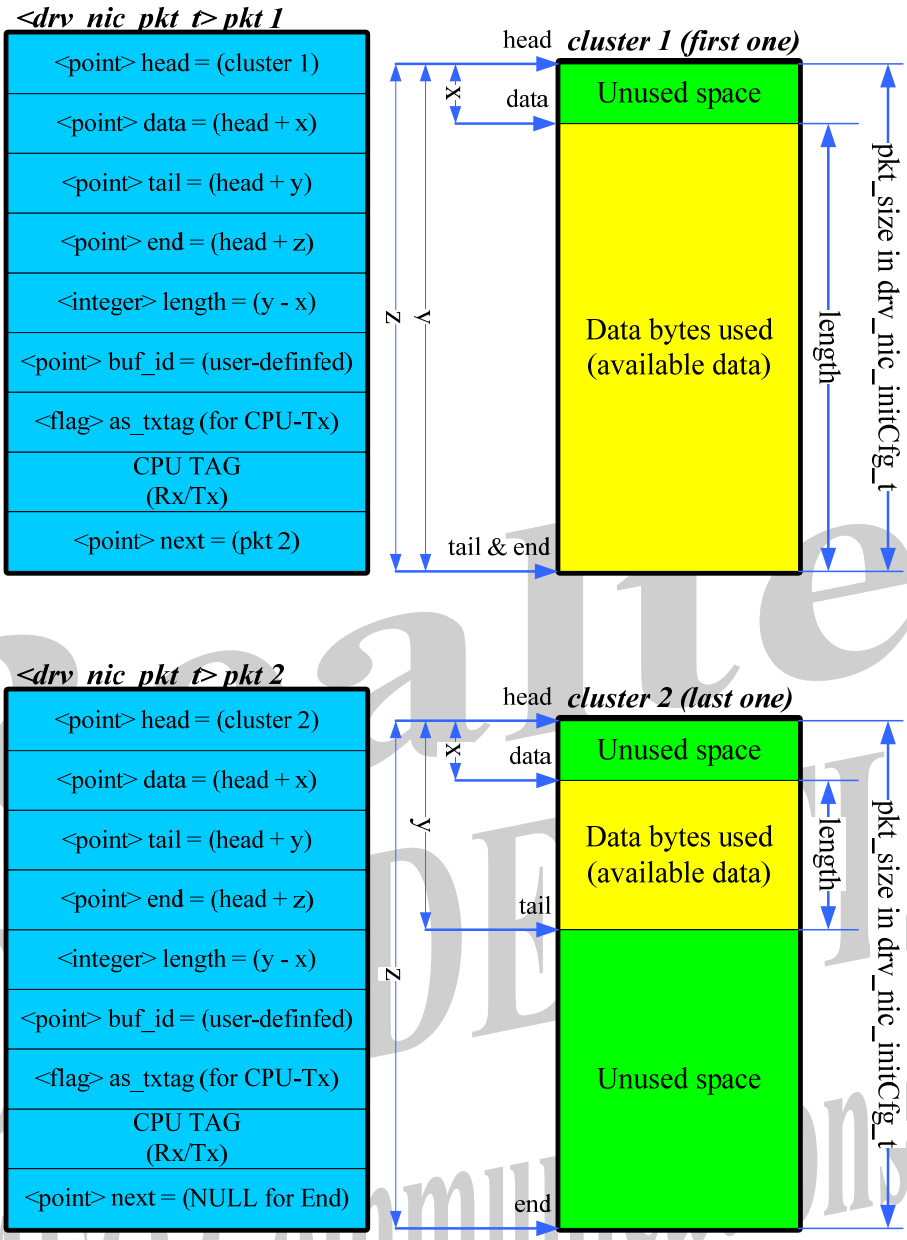


Figure 17: Example of Multiple Clusters Packet

17.2 Packet Reception (CPU-Rx)

The device provides 8 Rx rings mapped to the eight egress queue of the CPU MAC. When the packet is forwarded to CPU, the NIC performs a DMA Write to move the packet from the CPU MAC to the CPU memory.

API REFERENCE

FilePath: system/include/drv/nic/common.h

```
typedef enum drv_nic_rx_e
{
    NIC_RX_NOT_HANDLED = 0,     /* The packet is not handled, continue processing */
    NIC_RX_HANDLED,             /* The packet is handled and caller takes control of packet data */
    NIC_RX_HANDLED_OWNED,       /* The packet is handled and processing is completed */
} drv_nic_rx_t;

typedef int32 (*drv_nic_pkt_alloc_f)(uint32 unit, int32 size, uint32 flags, drv_nic_pkt_t **ppPacket);
typedef int32 (*drv_nic_pkt_free_f)(uint32 unit, drv_nic_pkt_t *pPacket);
typedef drv_nic_rx_t (*drv_nic_rx_cb_f)(uint32 unit, drv_nic_pkt_t *pPacket, void *pCookie);

typedef struct drv_nic_initCfg_s
{
    int32                   pkt_size;       /* Maximum bytes to support in a packet */
    drv_nic_pkt_alloc_f     pkt_alloc;      /* Packet structure and packet data allocation routine */
    drv_nic_pkt_free_f      pkt_free;       /* Packet structure and packet data deallocation routine */
} drv_nic_initCfg_t;

drv_nic_init(uint32 unit, drv_nic_initCfg_t *pInitCfg);
drv_nic_rx_register(uint32 unit, uint8 priority, drv_nic_rx_cb_f fRxCb, void *pCookie, uint32 flags);

drv_nic_rx_unregister(uint32 unit, uint8 priority, drv_nic_rx_cb_f fRxCb);
drv_nic_rx_start(uint32 unit);
drv_nic_rx_stop(uint32 unit);
```

During NIC driver initialization '*drv_nic_init*', it calls the user-defined function `*pkt_alloc*` in `*drv_nic_initCfg_t*` with `*pkt_size*` to prepare the buffer called Cluster to receive packet.

When a packet has been written to the Cluster (Memory), NIC notifies CPU to receive the packet with an interrupt signal (RX_DONE). Then the NIC driver callback the Rx (handler) callback function `*fRxCb*` that registered with `drv_nic_rx_register` API function according to priority. The parameter `*pCookie*` returned as an input parameter and user can use it to pass the application data. The parameter `*flags*` is optional which is reserved for future use.

The Rx handler can return one of the following values to indicate the result of handling.

1. NIC_RX_NOT_HANDLED – the packet is not handled. This return value indicates NIC driver to invoke the next Rx handler.
2. NIC_RX_HANDLED – the packet is handled by the Rx handler. This return value indicates the packet should be freed by the caller (NIC driver) by invoking the `*pkt_free*` in `*drv_nic_initCfg_t*`.
3. NIC_RX_HANDLED_OWNED – the packet is handled by the Rx handler. This return value indicates Caller (NIC driver) should not free the packet, and the packet free will be done by user.

Developer can invoke `*drv_nic_rx_start*`/`*drv_nic_rx_stop*` API to start/stop the Rx DMA process.

Table 137: DMA_IF_CTRL.RX Register

Field Name	Bits	Description
RX_EN	1	Enable Rx DMA process 1'b0: disable 1'b1: enable

According to the introduction above, the basic procedures to receive packets should be:

1. Invoke *drv_nic_init* API to initialize NIC if it has not been invoked yet.
2. Invoke *drv_nic_rx_register* API to register a Rx packet handler.
3. Invoke *drv_nic_rx_start* API to enable the Rx DMA process.

The Rx packet handler can retrieve related information of the received packet with the CPU Rx-Tag. Please refer to the CPU Tag Developer Guide for the detail.

CONFIDENTIAL
CAMEO Communications, Inc.

17.3 Packet Transmission (CPU-Tx)

The device provides 2 Tx rings (high/low priority) to transmit packet through CPU MAC to front ports.When CPU transmits a packet, the NIC performs a DMA Read to move the packet from CPU memory to CPU MAC.

API REFERENCE

```
typedef void (*drv_nic_tx_cb_f)(uint32 unit, drv_nic_pkt_t *pPacket, void *pCookie);

drv_nic_init(uint32 unit, drv_nic_initCfg_t *pInitCfg);
drv_nic_pkt_alloc(uint32 unit, int32 size, uint32 flags, drv_nic_pkt_t **ppPacket);
drv_nic_pkt_free(uint32 unit, drv_nic_pkt_t *pPacket);
drv_nic_pkt_tx(uint32 unit, drv_nic_pkt_t *pPacket, drv_nic_tx_cb_f fTxCb, void *pCookie);
```

The SDK driver select the Tx queue according to the priority in CPU-Tx tag, it use the high queue if the priority value (0~7) is over 3.

CPU can invoke API `*drv_nic_pkt_tx*` to transmit a packet. If the caller want to be notified after packet is transmitted successfully, it can use the parameter `*fTxCb*` - Tx callback function. The NIC driver invokes the `*fTxCb*` function with `*pCookie*` (application data returned with callback, it can be null) parameter to notify the caller if `*fTxCb*` is not NULL, then the packet should be freed by user. If `*fTxCb*` is NULL, the NIC driver automatically invokes the `*pkt_free*` callback function to free the packet.

Table 138: DMA_IF_CTRL.TX Register

Field Name	Bits	Description
TX_EN	1	Enable Tx DMA process 1'b0: disable 1'b1: enable
TX_FETCH	1	Tx descriptor fetch notify (Set this bit to trigger the packet send) Note: Auto clear to 0 after trigger.

The basic procedures to transmit packet should be:

1. Invoke *drv_nic_init* API to initialize NIC if it has not been invoked yet.
2. Invoke *drv_nic_pkt_alloc* API to allocate an empty packet buffer for use.
3. Write raw data into the packet buffer (cluster).
4. Invoke *drv_nic_pkt_tx* API to send this packet.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

18 CPU Tag

CPU Tag is a 12-Byte payload which is inserted between Source MAC Address and EtherType field. CPU Tag has different meaning for CPU Rx and Tx. It is used to indicate the packet processing information when CPU Rx and used to instruct the switch processing behavior when CPU Tx.

CPU Rx Tag is inserted by CPU MAC when the switch forwards packet to CPU. CPU Tx Tag is inserted by NIC when CPU transmits packet to switch and it is removed before transmitting to front ports.

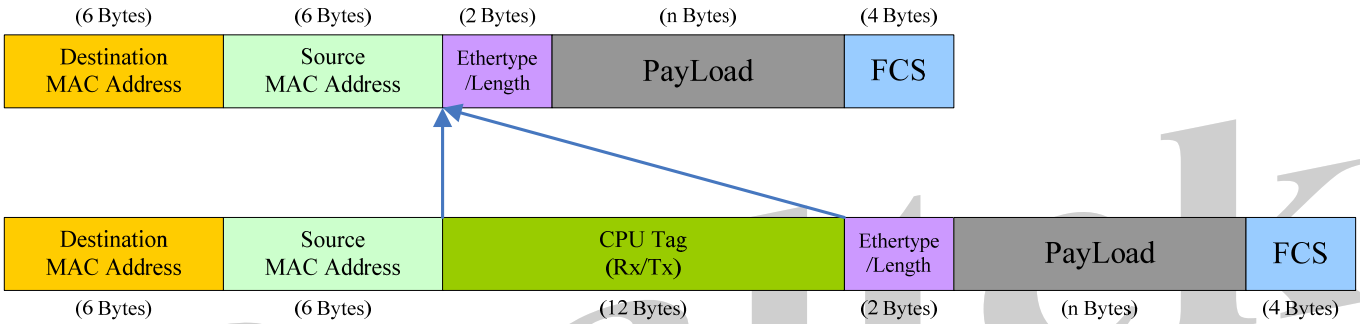


Figure 18: CPU Tag Position

18.1 CPU Rx Tag

The CPU Rx Tag provides information regarding the receiving packet and its payload is shown as below:

Table 139: Field Description of CPU Rx Tag

Field Name	Bits	Description
SPN	5	Ingress port which the packet came from. (0~28)
MIR_HIT	4	Mirroring hit status for 4-sets of mirror configuration. 4`b[3:0] = { set 3, set 2, set 1, set 0 }
ACL_IDX	12	Indicate the hit ACL entry index. Valid if ACL_HIT=1.
ACL_HIT	1	Indicate whether ACL hit.
OTAGIF	1	Indicate whether the packet contains outer tag.
ITAGIF	1	Indicate whether the packet contains inner tag.
FORWARD_VID	12	Forwarding VLAN ID
QID	3	Indicate the egress queue the packet came from.
ATK_HIT	1	To indicate that attack prevention has detected this attack
ATK_TYPE	5	Indicate attack type: 5'b00000: DA = SA 5'b00001: LAND attack (DIP = SIP) 5'b00010: UDP Blat attack (DPORT = SPORT) 5'b00011: TCP Blat attack (DPORT = SPORT) 5'b00100: POD (ping of death) attack 5'b00101: IPv6 fragmented that size is less than the minimum. 5'b00110: ICMP fragmented packet. 5'b00111: ICMPv4 ping that size is over the maximum. 5'b01000: ICMPv6 ping that size is over the maximum. 5'b01001: Smurf (ICMP ping request that DIP is broadcast IP). 5'b01010: TCP header is not complete. 5'b01011: TCP SYN/!ACK packet with sport < 1024. 5'b01100: TCP NULL scan. 5'b01101: TCP XMAS attack (seq=0, FIN,URG,PSH are set)

Realtek CONFIDENTIAL CAMEO Communications, Inc.

		5'b01110: TCP SYN/FIN packet (SYN-FIN scan attack). 5'b01111: TCP SYN/RST packet (SYN-RST scan attack). 5'b10000: TCP Fragment Offset = 1 5'b10001: ARP invalid 5'b10010: Gratuitous ARP 5'b10011~5'b11111: Reserved
MAC_CST	1	Indicate this packet meets the MAC constraint condition.
NEW_SA	1	The Source MAC address is a new address and it has been learnt.
L2_PMV	1	The Source MAC causes the port moving of L2 entry.
OVERSIZE	1	The packet is truncated by NIC (means it's not a completed packet).
REASON	5	0: None (the field is invalid) 1: RLDP or RLPP packet 2: RMA 3: VLAN ingress filter 4: Inner/Outer tag CFI/DEI =1 5: CPU MAC 6: Special trap(IGMP/MLD/EAPOL) 7: Special copy(ARP/IPV6 Neighbor Discovery) 8: Routing exception 9: DA lookup miss(L2UC/L2MC/IPMC/IPv6MC) 10: MAC_CST_SYS(system-based mac constraint) 11: MAC_CST_VLAN(vlan-based mac constraint) 12: MAC_CST_PORT (port-based mac constraint) 13: L2_CRC_ERR 14: IPv6_EXTHDR 15: Normal forwarding (due to L2 table lookup)

When REASON=0, it means the packet is trapped by the other reason. and should refer to *MIR_HIT*, *ACL_HIT*, *ATK_HIT* , *NEW_SA*, and *L2_PMV* fields.

18.2 CPU Tx Tag

The CPU Tx Tag provides control information to instruct switch how to process the packet and it is shown as below:

Table 140: Field Description of CPU Tx Tag

Field Name	Bits	Description
DPM_TYPE	1	Indicate the type of destination port mask: 1'b0: Logical (physical egress port is calculated by trunk module) 1'b1: Physical
ACL_ACT	1	Indicate whether the ingress/egress ACL should take effect.
BP_FLTR_1	1	Indicate whether to bypass filtering of following modules: (1) Egress Port Isolation (2) Egress Mirror Isolation
BP_FLTR_2	1	Indicate whether to bypass filtering of following modules: (1) Egress spanning-tree port state (except disabled state) (2) Egress VLAN filtering
AS_PRI	1	Priority assignment. 1'b0: the priority is decided by lookup process. 1'b1: assign priority directly.
PRI	3	Assigned priority value (0~7) Note: this field takes effect only when AS_PRI=1.
L2LEARNING	1	L2 learning behavior. 1'b0: don't learn.

Realtek CONFIDENTIAL CAMEO Communications, Inc.

		1'b1: learn, but still limited by L2 learning constraint.
ALE_AS_TAGSTS	1	Determines the Outer/Inner TAG processing behavior. 1'b0: don't touch VLAN tag. 1'b1: normal processing.
FORWARD_VID_SEL	1	Forwarding VID selection: 1'b0: use inner tag 1'b1: use outer tag
AS_DPM	1	Assign egress ports. 1'b0: follow address table lookup. 1'b1: assign destination ports.
DPM	29	Destination Port Mask (Refer to AS_DPM and DPM_TYPE fields). Note: this field takes effect only when AS_DPM=1.

BP_FLTR_1 and BP_FLTR_2 are useful for some control protocols, such as UDLD(Unidirectional Link Detection), which underlie the logical link layer and would like to ignore the filtering modules.

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

19 802.1X

In many environments, the access to the service offered by LAN is permitted only after authorized for the security consideration. 802.1X is a port-based network access control protocol that provides a mean of authenticating and authorizing devices attached to the LAN port, and of preventing access to that port if the authentication process fails. The following figure illustrates the infrastructure of 802.1X:

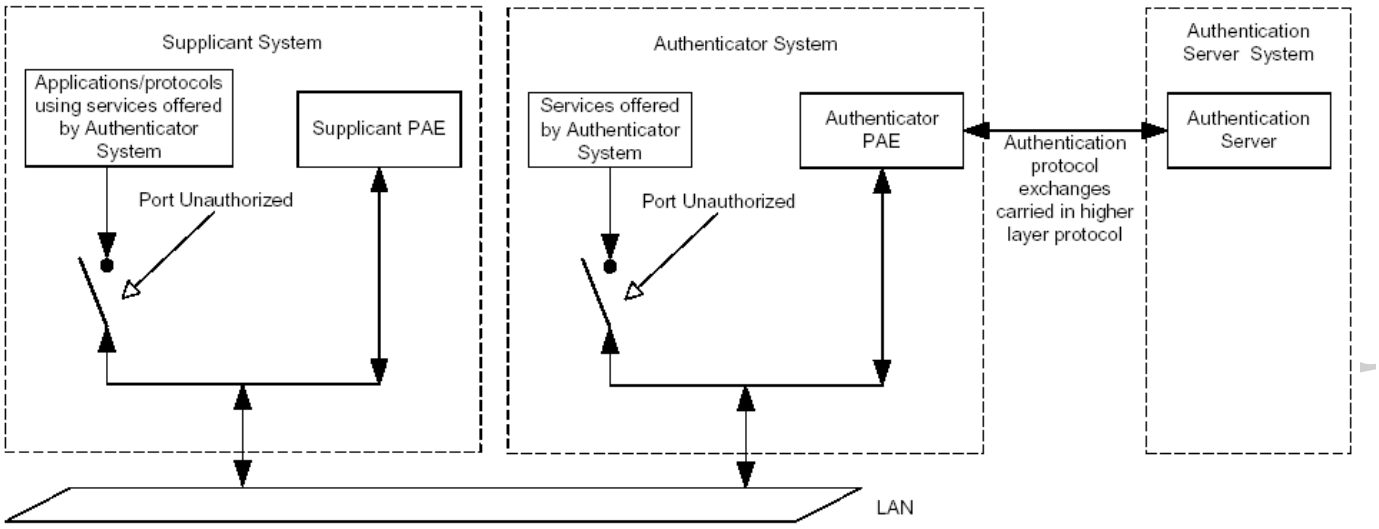


Figure 6-5—Authenticator, Supplicant, and Authentication Server roles

In the infrastructure, there are three roles – authenticator, supplicant, and authentication server. In this document, the authenticator is our switch which works as a communication agent between supplicant (such as PC) and authentication server (may be a security database). The supplicant provides credentials to the authentication server through authenticator. If the authentication server agrees this supplicant, it will send "ACCEPT" message to the switch, otherwise it will send "REJECT". On receiving the above decision from authentication server, the switch will open/close its LAN service to the supplicant.

For controlling the network access, we have a premise that an unauthorized host must not "register" its source MAC in switch's L2 table. Therefore, by controlling the L2 learning and forwarding behavior, a host may be allowed or denied to access the LAN service.

This document will suggest means for trapping the above communication packets and how to provide/close LAN network service making use of ASIC's facilities. The state machine transformation and encapsulation/de-capsulation of packet is defined in 802.1X protocol and out of the scope of this document. For detail behavior of this protocol, please refer to IEEE 802.1X standard.

CONFIDENTIAL
CAMEO Communications, Inc.

19.1 Port-based authentication

In port-based authentication, a port on the switch is permitted to use network service only after the successful authentication. Due to its port-based essence, all hosts connect to this port are view as authorized.

19.1.1 Receiving Control Packet

In port-based authentication, the communication packets between supplicant and switch are called EAPOL and those between switch and authentication server are called EAP on RADIUS. If 802.1X is enabled on a port, we should trap the EAPOL and RADIUS packets to CPU.

To trap EAPOL, we can make use of SPCL_TRAP_EAPOL_CTRL.

Table 141: SPCL_TRAP_EAPOL_CTRL Register

Field Name	Bits	Description
ACT	1	Indicate whether trap EAPOL packet whose Ethertype = 0x888E. 0b0: forward 0b1: trap

Note that whenever this bit set to 1, the EAPOL packets will be trap to CPU even they are received on an 802.1X disabled port. If you would like to trap only on enabled port, you could achieve by edit an ACL rule and set the portmask on those enabled ports. For editing an ACL rule, please refer to "*8390_Developer_Guide_ACL*".

Because EAP on RADIUS packet is a typical UDP packet, therefore we can trap it by checking the UDP port.

API REFERENCE

```
typedef enum rtk_trap_mgmtType_e
{
    MGMT_TYPE_RIP = 0,              /* Applicable to 8328 */
    MGMT_TYPE_ICMP,                 /* Applicable to 8328 */
    MGMT_TYPE_ICMPV6,               /* Applicable to 8328 */
    MGMT_TYPE_ARP,
    MGMT_TYPE_MLD,
    MGMT_TYPE_IGMP,
    MGMT_TYPE_BGP,                  /* Applicable to 8328 */
    MGMT_TYPE_OSPFV2,               /* Applicable to 8328 */
    MGMT_TYPE_OSPFV3,               /* Applicable to 8328 */
    MGMT_TYPE_SNMP,                 /* Applicable to 8328 */
    MGMT_TYPE_SSH,                  /* Applicable to 8328 */
    MGMT_TYPE_FTP,                  /* Applicable to 8328 */
    MGMT_TYPE_TFTP,                 /* Applicable to 8328 */
    MGMT_TYPE_TELNET,               /* Applicable to 8328 */
    MGMT_TYPE_HTTP,                 /* Applicable to 8328 */
    MGMT_TYPE_HTTPS,                /* Applicable to 8328 */
    MGMT_TYPE_DHCPV6,               /* Applicable to 8328 */
    MGMT_TYPE_DHCP,                 /* Applicable to 8328 */
    MGMT_TYPE_DOT1X,                /* Applicable to 8328 */
    MGMT_TYPE_BPDU,
    MGMT_TYPE_PTP,                  /* Applicable to 8390 */
    MGMT_TYPE_LLDP,                 /* Applicable to 8390 */
    MGMT_TYPE_EAPOL,                /* Applicable to 8390, 8380 */
    MGMT_TYPE_IPV6ND,               /* Applicable to 8390 */
    MGMT_TYPE_SELFMAC,              /* Applicable to 8390, 8380 */
    MGMT_TYPE_OTHER,                /* Applicable to 8390, 8380 */
    MGMT_TYPE_OAM,                  /* Applicable to 8390 */
    MGMT_TYPE_CFM,                  /* Applicable to 8390 */
    MGMT_TYPE_IGR_VLAN_FLTR,        /* Applicable to 8390, 8380 */
    MGMT_TYPE_VLAN_ERR,             /* Applicable to 8390 */
    MGMT_TYPE_CFI,                  /* Applicable to 8390, 8380 */
    MGMT_TYPE_RMA_USR_DEF_1,        /* Applicable to 8390 */
    MGMT_TYPE_RMA_USR_DEF_2,        /* Applicable to 8390 */
    MGMT_TYPE_LACP,                 /* Applicable to 8390 */
    MGMT_TYPE_IPV4_HDR_ERR,         /* Applicable to 8390 */
    MGMT_TYPE_IPV4_TTL_EXCEED,      /* Applicable to 8390 */
    MGMT_TYPE_IPV4_OPT,             /* Applicable to 8390 */
    MGMT_TYPE_IPV6_HDR_ERR,         /* Applicable to 8390 */
    MGMT_TYPE_IPV6_HL_EXCEED,       /* Applicable to 8390 */
    MGMT_TYPE_IPV6_HOPBYHOP,        /* Applicable to 8390 */
```

```
    MGMT_TYPE_GW_MAC_ERR,           /* Applicable to 8390 */
    MGMT_TYPE_UNKNOWN_DA,           /* Applicable to 8390, 8380*/
    MGMT_TYPE_RLDP_RLPP,          /* Applicable to 8380*/
    MGMT_TYPE_RMA,                /* Applicable to 8390, 8380*/
    MGMT_TYPE_SPECIAL_COPY,       /* Applicable to 8380*/
    MGMT_TYPE_SPECIAL_TRAP,       /* Applicable to 8380*/
    MGMT_TYPE_ROUT_EXCEPT,        /* Applicable to 8380*/
    MGMT_TYPE_MAC_CST_SYS,        /* Applicable to 8380*/
    MGMT_TYPE_MAC_CST_PORT,       /* Applicable to 8380*/
    MGMT_TYPE_MAC_CST_VLAN,       /* Applicable to 8380*/
    MGMT_TYPE_IPV6_HOP_POS_ERR, /* Applicable to 8390, 8380*/
    MGMT_TYPE_IPV6_HDR_UNKWN,     /* Applicable to 8390, 8380*/
    MGMT_TYPE_L2_CRC_ERR,         /* Applicable to 8390, 8380*/
    MGMT_TYPE_IP4_CHKSUM_ERR,     /* Applicable to 8390*/
    MGMT_TYPE_INVALID_SA,         /* Applicable to 8390*/
    MGMT_TYPE_END
} rtk_trap_mgmtType_t;

rtk_trap_mgmtFrameAction_get (uint32 unit, rtk_trap_mgmtType_t frameType, rtk_action_t *pAction);
rtk_trap_mgmtFrameAction_set(uint32 unit, rtk_trap_mgmtType_t frameType, rtk_action_t action);
```

19.1.2 Control on Network Access

To prevent supplicant from access the network service illegally, we can use the L2_PORT_SALRN and L2_PORT_NEW_SA_FWD. When a port is 802.1X enabled, we take the following steps to control the network access on this port:

1. Flush all MACs learnt on this port
2. Set the L2 table SA learning behavior to "Not Learn" for a packet whose SA is new to system
3. Set the forwarding behavior to "Drop" for a packet whose SA is new to system

By flushing all MACs learnt on this port, all incoming packets are treats as new SA and thus be dropped before this port is authorized. After successful authentication, we change the learning mode of this port to "ASIC Auto Learn" and "forward". Afterward, all ingress traffic is forwarded normally.

Table 142: L2_PORT_SALRN Register

Field Name	Bits	Description
SALRN	2	L2 table SA learning behavior 00b: ASIC Auto Learn 01b: learn as a suspend entry 10b: not Learn 11b: reserved

Table 143: L2_PORT_NEW_SA_FWD Register

Field Name	Bits	Description
NEW_SA_FWD	2	Packet forwarding behavior when receives a packet with new SMAC. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
typedef enum rtk_l2_newMacLrnMode_e
{
    HARDWARE_LEARNING = 0,
    SOFTWARE_LEARNING,
    NOT_LEARNING,
    LEARNING_MODE_END,
} rtk_l2_newMacLrnMode_t;

rtk_l2_newMacOp_get(
    uint32                  unit,
    rtk_port_t              port,
    rtk_l2_newMacLrnMode_t  *pLrnMode,
    rtk_action_t            *pFwdAction);

rtk_l2_newMacOp_set(
    uint32                  unit,
    rtk_port_t              port,
    rtk_l2_newMacLrnMode_t  lrnMode,
    rtk_action_t            fwdAction);
```

19.1.3 OperControlledDirections

The OperControlledDirections is a parameter defined in the IEEE 802.1X standard which defines whether an authorized port can talk to an unauthorized port:

1. **OperControlledDirections = Both**. An authorized port can NOT talk to an unauthorized port.
2. **OperControlledDirections = In**. An authorized port can talk to an unauthorized port.

In normal switch forwarding process, the DMAC of a packet may be or not in L2 table. If the DMAC is already in L2 table, it implies the destination host has been authorized (or the egress port is 802.1X disabled). In another condition that DMAC is unknown, this packet will be flood to all port (or on VLAN in an 802.1Q VLANs environment). Due to the principle that a host attached to an unauthorized port must not register its source MAC address in L2 table, we can accomplish this functionality by properly setting the L2 lookup miss flooding portmask:

$FLD\_PMSK = \{disable, port\text{-}auth, \{port\text{-}unAuth \cap opdir[port] = IN\}\}$

When a port is 802.1X enabled and OperControlledDirections set to "Both", it should be removed from the *FLD_PMSK* before authorized. Otherwise, we put this port to *FLD_PMSK* again.

Table 144: L2_PORT_LM_ACT Register

Field Name	Bits	Description
L2_UC_LM_ACT	2	Per port L2 unicast packet lookup miss action. Note: A L2 unicast packet is deemed to KNOWN if its DMAC and forwarding VID are equal to switchs MAC and the PVID of CPU port respectively. 0b00: forward (Flooding to L2_UNKN_UC_FLD_PMSK) 0b01: drop 0b10: trap 0b11: copy to CPU

API REFERENCE

```
typedef enum rtk_l2_lookupMissType_e
{
    DLF_TYPE_IPMC = 0,      /* Applicable to 8328, 8390, 8380 */
    DLF_TYPE_IP6MC,         /* Applicable to 8328, 8390, 8380 */
    DLF_TYPE_UCAST,         /* Applicable to 8328, 8390, 8380 */
```

Realtek CONFIDENTIAL CAMEO Communications, Inc

```
    DLF_TYPE_BCAST,          /* Applicable to 8328 */
    DLF_TYPE_MCAST,          /* Applicable to 8328, 8390, 8380 */
    DLF_TYPE_ANY,            /* Applicable to 8389 */
    DLF_TYPE_END
} rtk_l2_lookupMissType_t;

rtk_l2_portLookupMissAction_get(uint32 unit, rtk_port_t port, rtk_l2_lookupMissType_t type, rtk_action_t *pAction);
rtk_l2_portLookupMissAction_set(uint32 unit, rtk_port_t port, rtk_l2_lookupMissType_t type, rtk_action_t action);
rtk_l2_lookupMissFloodPortMask_get(uint32 unit, rtk_l2_lookupMissType_t type, rtk_portmask_t *pFlood_portmask);
rtk_l2_lookupMissFloodPortMask_add(uint32 unit, rtk_l2_lookupMissType_t type, rtk_port_t flood_port);
rtk_l2_lookupMissFloodPortMask_del(uint32 unit, rtk_l2_lookupMissType_t type, rtk_port_t flood_port);
```

19.2 MAC-based authentication

Besides port-based authentication, we could also implement the MAC-based authentication on a per-port configured basis. In port-based authentication, if a port got authorized, all hosts connected to which are permitted to access network. But in MAC-based authentication, only the host with authorized source MAC could access the LAN service.

19.2.1 Receiving Control Packet

When a port is configured to MAC-based authentication enabled, depending on the application, the MAC-based authentication process may have or may not have EAPOL control packet. If it also uses EAPOL as the control packet, we can use the method described in section 1.1.1. If the application uses the data packets credentials, we can just trap these packets use L2_PORT_SALRN.

Table 145: L2_PORT_SALRN Register

Field Name	Bits	Description
SALRN	2	L2 table SA learning behavior 00b: ASIC Auto Learn 01b: learn as a suspend entry 10b: not Learn 11b: reserved

Table 146: L2_PORT_NEW_SA_FWD Register

Field Name	Bits	Description
NEW_SA_FWD	2	Packet forwarding behavior when receives a packet with new SMAC. 0b00: forward 0b01: drop 0b10: trap 0b11: copy to CPU

If the whole authentication process needs only one packet, we can set the L2_PORT_SALRN to "learn as suspend" and ASIC will trap the first packet of the same SMAC to CPU and drop others in a limited time period (depends on the L2 entry aging-time configuration). If the authentication process needs many packets, we should set the L2_PORT_SALRN to "not Learn". In both the above two scenarios, we set the L2_PORT_NEW_SA_FWD to "trap".

If administrator decides to disable MAC-based authentication on this port, we just set the L2_PORT_SALRN and L2_PORT_NEW_SA_FWD to "ASIC Auto Learn" and "forward" respectively.

API REFERENCE

Realtek CONFIDENTIAL CAMEO Communications, Inc.

```
rtk_l2_newMacOp_get(
    uint32                      unit,
    rtk_port_t                  port,
    rtk_l2_newMacLrnMode_t   *pLrnMode,
    rtk_action_t                *pFwdAction);

rtk_l2_newMacOp_set(
    uint32                      unit,
    rtk_port_t                  port,
    rtk_l2_newMacLrnMode_t   lrnMode,
    rtk_action_t                fwdAction);
```

19.2.2 Control on Network Access

Once the L2_PORT_SALRN to "learn as suspend", ASIC will drop all following packets except the first one. Thus prevent from unauthorized access. If L2_PORT_NEW_SA_FWD is set to "trap", we can drop all incoming packets in CPU, or even set the L2_PORT_NEW_SA_FWD to "drop" further based on our policy.

If the host passes the authentication criteria, we just simply add a L2 entry of its SMAC. Then, all traffic from this host is granted to the system and LAN services are still unavailable to other hosts.

API REFERENCE

```
rtk_l2_ucastAddr_s
{
    rtk_vlan_t   vid;
    rtk_mac_t    mac;
    rtk_port_t   port;
    uint32                  trk_gid;                /* The field is valid if flags enable
                                RTK_L2_UCAST_FLAG_TRUNK_PORT bit */
    uint32       flags;             /* Refer to RTK_L2_UCAST_FLAG_XXX */
    uint32       state;             /* Refer to RTK_L2_UCAST_STATE_XXX */
    uint32       l2_idx;            /* returned l2 entry index after a successful read/write */
    rtk_vlan_t   agg_vid;           /* The field is auto-learned by H/W and utilized by MAC-based N:1 VLAN
                                    translation */
    uint8        vlan_target;       /* Target VLAN to swap. 0:inner 1:outer
                                        * The field is used only if RTK_L2_UCAST_FLAG_NEXTHOP is set.
                                        */
    uint32       route_idx;         /* Index to routing table.
                                        * The field is used only if RTK_L2_UCAST_FLAG_NEXTHOP is set.
                                        */
} rtk_l2_ucastAddr_t;

rtk_l2_addr_add(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_get(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_set(uint32 unit, rtk_l2_ucastAddr_t *pL2_addr);
rtk_l2_addr_del(uint32 unit, rtk_vlan_t vid, rtk_mac_t *pMac);
```

19.2.3 OperControlledDirections

Unlike port-based authentication, the OperControlledDirections in MAC-based authentication is global setting. Fortunately, we could make use of the L2 lookup miss flooding portmask again.

- OperControlledDirections = Both, `FLD_PMSK = { ALL_port }`
- OperControlledDirections = In, `FLD_PMSK = { ALL_port - macBasedEnable[port]}`

An authorized host must have registered its SMAC in L2 table, so we just remove the ports that enable MAC-based authentication when OperControlledDirections is configured as "In" mode.

19.3 Port-based Guest VLAN

Guest VLAN is an environment that provides unauthorized host limited network service. Our system could easily implement port-based Guest VLAN.

When a port failed in authentication, we could put this port on Guest VLAN by set its port-based VLAN ID and move its membership form original VLAN member set to the Guest VLAN set. For detail VLAN manipulation, please refer to "*8390_Developer_Guide_VLAN*". By this way, the guest traffic is separated from normal traffic.

As soon as the port is authenticated, we can restore the VLAN configuration to original.

PROGRAMMING EXAMPLE

```
/* Set port-based unauthorized packet action */

rtk_l2_newMacLrnMode_t oldsa_mode;
rtk_action_t old_action;

tk_l2_newMacOp_get(physicPort.dev, physicPort.port, &oldsa_mode, &old_action);
rtk_l2_newMacOp_set(physicPort.dev, physicPort.port, oldsa_mode, action);

/* store the configuration for conveniently */
portUnauthAct[port] = action;

/* Set rate limit for trap2cpu action */
```

```
/* Set port-based authentication status */

switch (status)
{
    case SYS_AUTH_DISABLE:
    case SYS_AUTH:
        rtk_l2_newMacOp_set(physicPort.dev, physicPort.port,
                                                HARDWARE_LEARNING,
                                                ACTION_FORWARD);
        break;

    case SYS_UNAUTH:
    default:
    {
        rtk_l2_flushCfg_t config;

        osal_memset(&config, 0, sizeof(rtk_l2_flushCfg_t));
        config.flushByPort = TRUE;
        config.portOrTrunk = TRUE; /* port-based */
        config.port = physicPort.port;

        rtk_l2_newMacOp_get(port, &action));
```

```
            rtk_l2_newMacOp_set(physicPort.dev, physicPort.port,
                                                    NOT_LEARNING,
                                                    action);
            rtk_l2_ucastAddr_flush(physicPort.dev, &config);
            break;
        }
}

/* store the configuration for conveniently */
portAuthEbl[port] = status;
```

Realtek
CONFIDENTIAL
CAMEO Communications ,Inc.

20 Cable Diagnostic

Cable diagnostic is supported by "RTCT" (RealTek Cable Tester) feature of Realtek PHYs. RTCT can detect open, short or normal in each differential pairs with cable length information. It transmits a signal of known amplitude to one pair of an attached cable. The transmitted signal will continue down the cable until it reflects off a cable imperfection. The magnitude of the reflection and the time it takes for the reflection to come back is then used to locate the type of problem and the distance to the problem.

For a normal link up port, its port state is linked down when RTCT starts and the link state is recovered after RTCT stops. If links down is not desirable for a normal link up port, there is another feature "Green Ethernet" can be used to retrieve the cable length.

20.1 RTCT

RTCT flow chart is shown as below:

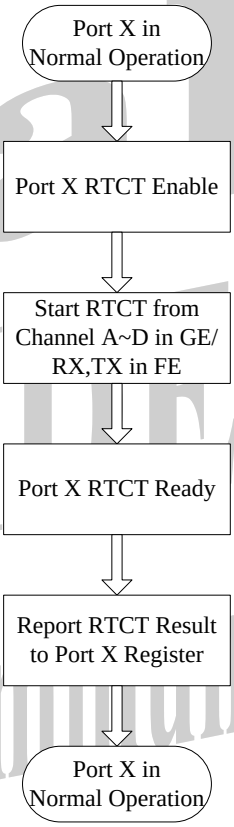


Figure 19: RTCT Flow Chart

Four states are supplied by RTCT:

- Short

A short is detected on the cable.

- Open

An opening is detected on the cable. One scenario is the cable doesn't plug to the link partner.

- Impedance Mismatch

The impedance is mismatched. The normal impedance should be 100Ω, impedance mismatch is detected if the impedance measured is not in the range 70Ω~130Ω.

Realtek CONFIDENTIAL CAMEO Communications ,Inc.

- Line Driver

The high impedance is detected. One scenario is the cable plug to a power down link partner.

The cable length reported is the length where Short/Open/Impedance Mismatch/Line Driver is detected and **the cable length reported by RTCT could have a plus or minus 3 meters inaccuracy.**

API REFERENCE

```
rtk_diag_rtctEnable_set(uint32 unit, rtk_portmask_t *pPortmask);
rtk_diag_portRtctResult_get(uint32 unit, rtk_port_t port, rtk_rtctResult_t *pRtctResult);
```

20.2 Green Ethernet

Green Ethernet is another feature that can be used to retrieve the cable length when the port is linked on. **The cable length reported by Green Ethernet could have a plus or minus 15 meters inaccuracy.**

API REFERENCE

```
rtk_diag_rtctEnable_set(uint32 unit, rtk_portmask_t *pPortmask);
rtk_diag_portRtctResult_get(uint32 unit, rtk_port_t port, rtk_rtctResult_t *pRtctResult);

*The API inside automatically switch to Green Ethernet if the diagnostic port to be checked is link up.
```

Realtek CONFIDENTIAL CAMEO Communications ,Inc.