REALTEK

RTL9607C
SINGLE-CHIP PON

VLAN
Application Note
(CONFIDENTIAL: Development Partners Only)

Rev. 1.0.0
01 Jun 2017

REALTEK
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

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

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.

CONFIDENTIALITY

This document is confidential and should not be provided to a third-party without the permission of Realtek Semiconductor Corporation.

REVISION HISTORY

Revision	Release Date	Summary
1.0.0	2016/06/01	First Release

Table of Contents

List of Tables

List of Figures

1. Overview

RTL9607C supports 4K entries in VLAN table and it shares for SVLAN and CVLAN. When SVID and CVID are the same, users cannot set different configurations in the same VLAN entry. For this scenario, users can use classification module to filter CVLAN/SVLAN or treat them. The CVLAN module supports 802.1q VLAN bridging and can operate with the following capabilities:

- VLAN table
- CVLAN member assignment
- Multiple Spanning Tree Instance
- IVL and SVL learning mode
- Decision IVL and SVL learning mode
- Enable/Disable CVLAN function
- Ingress CVLAN filtering
- Port based CVID
- Protocol based CVLAN
- Ingress CVLAN decision
- Egress CVID decision and egress filtering
- CVLAN Leaky
- Per port accept frame type configuration
- Action for Reserved VID 0 and VID 4095
- Egress CVLAN tag format configuration

The SVLAN (Stacking/Service VLAN) module is designed by IEEE 802.1ad. It's known as like Q-in-Q function and can operate with the following capabilities:

- Enable/Disable SVLAN function
- Aware SVLAN tag
- Action for SVLAN un-tagging
- Port based SVID
- SVLAN decision and filtering behavior
- SVLAN forwarding behavior

- SVLAN TPID configuration
- SVLAN trap priority
- S-priority assignment
- Keep DEI of SVLAN
- SP2C configuration

2. VLAN table

The relationship of the VLAN table entry and VLAN ID is direct mapping. For VID = 1000, its configuration is recorded in 1000 index entry in VLAN table. The following figure shows bit order of each field of entry in VLAN table.

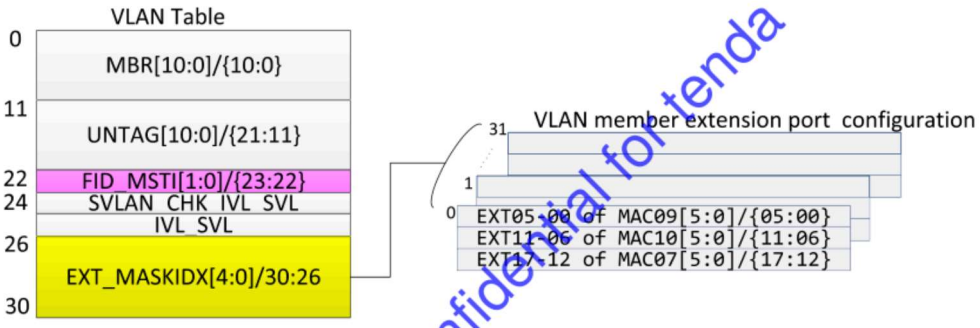

Figure 1. VLAN Table Entry

The below table illustrates description of each field of entry in VLAN table.

Table 1. VLAN Table

Field Name	Bits	Description
MBR	11	VLAN member port mask
UNTAG	11	VLAN Un-tag member port mask
FID_MSTI	2	Filtering Database/Multiple Spanning Tree Instance
SVLAN_CHK_IVL_SVL	1	For SVLAN decision using SVLAN IVL/SVL 0: SVLAN not check IVL_SVL setting 1: force SVLAN check SVL_IVL
IVL_SVL	1	0: SVL 1: IVL
EXT_MASKIDX	5	EXT member index for MAC9/10/7 extension port member

The EXT_MASKIDX field points to 32 entries of VLAN member extension port configuration. Each entry includes EXT 0 to EXT 17. EXT 5-0 of which are belong to MAC 9 and EXT 11-6 are belong to MAC 10. The rests of which are belong to MAC 7.

3. CVLAN member assignment

The each entry of VLAN table provides some fields for assigning the member port mask, un-tag member port mask and extension member port. For the VLAN member port mask, it is used to indicate packet with VID could be forwarded to there. For VLAN un-tag member port mask, it is used to indicate packet with VID would be removed VLAN tag while sending it to egress port. The VLAN member extension port mask is the extension virtual ports for CPU.

4. Multiple Spanning Tree Instance

Per each VLAN, system total provide 2 instances for multiple spanning tree. User can set the spanning tree instance in each entry of VLAN table.

5. IVL and SVL learning mode

System provides IVL (independent VLAN learning) and SVL (shared VLAN learning). Each entry of VLAN table would set the learning mode of this VLAN is IVL or SVL. For SVL mode, each VLAN entry would also set the FID to separate the learning domain for all VLAN entries with SVL mode. System total provide 2 FID. Here gives an example for illustrating learning domain.

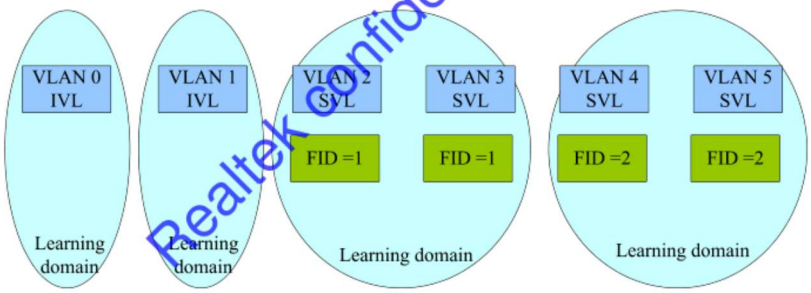

Figure 2. IVL/SVL Learning Domain

6. Decision IVL and SVL learning mode

System provides IVL or SVL learning mode can be decided by SVLAN or CVLAN. The below figure would illustrate decision method for IVL/SVL of packet.

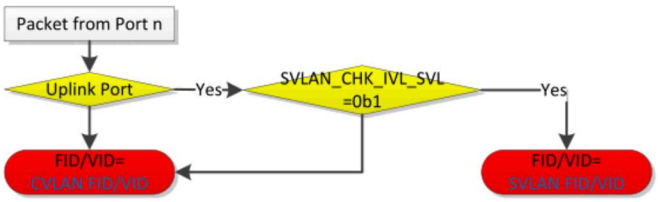

Figure 3. Decision IVL and SVL

7. Enable/Disable CVLAN

The CVLAN function would be global enabled/disabled. When CVLAN function is disabled, the CVLAN ingress/egress filtering function is disabled. In other words, the packets will not be filtered by CVLAN setting and ASIC is like a dump switch. Even if enable CVLAN function and ingress CVLAN filtering, the packet does NOT be restricted by ingress CVLAN filtering and CVLAN system control when it matches ACL rule or belong to Reserved Multicast Address. The below table illustrates CVLAN ingress/egress filtering function configuration.

Table 2. CVLAN Filtering Setting

Field Name	Bits	Description
VLAN_FILTERING	1	System VLAN ingress/egress function setting 0b0: disable CVLAN ingress/egress filtering 0b1: enable CVLAN ingress/egress filtering

8. Ingress CVLAN filtering

For CVLAN module per port could enable or disable the ingress CVLAN filter function. If ingress CVLAN filtering function is enabled and the source port does NOT belong to CVLAN member port, the ingress packet would be dropped by CVLAN module. The below table illustrates per-port ingress CVLAN filter function configuration.

Table 3. Per-port Ingress CVLAN Filtering Setting

Field Name	Bits	Description
VLAN_PORTn_INGRESS	1	Per-port VLAN ingress check for source member VLAN 0b0: disable source member VLAN ingress 0b1: enable source member VLAN ingress

9. Port based CVID

Each port can be assigned the port-based CVID. When the ingress packet does not match any CVID setting, the CVID will be assigned by port-based CVID. The below table illustrates port-based CVID configuration of MAC 0-10 and EXT 0-17.

Table 4. CVLAN PVID Setting

Field Name	Bits	Description
VLAN_PORTn_VID	12	VLAN ID for MAC 0-10
VLAN_EXTn_VID	12	VLAN ID for MAC 9 EXT 5-0/MAC10 EXT11-6/MAC7 EXT 17-12

10. Protocol based CVLAN

There are 4 entries of VLAN protocol in the system. Each entry provides 3 frame types: Ethernet, LLC-Other and RFC1042. The below figure describes the protocol match rule.

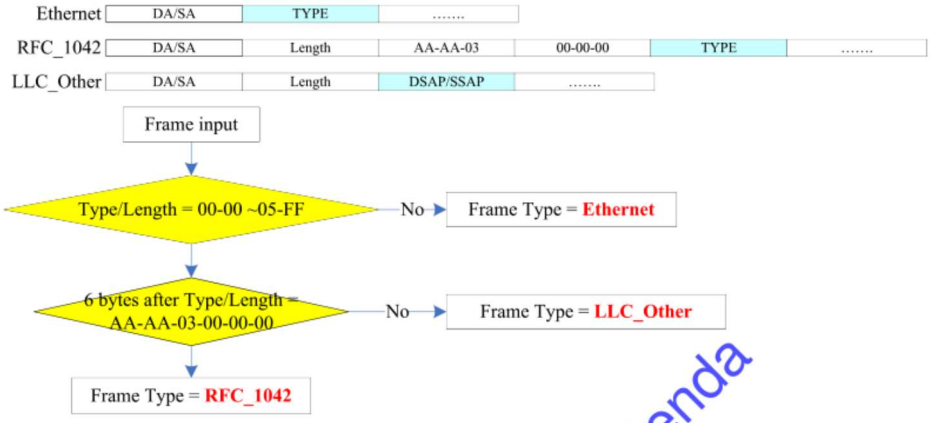

Figure 4. Protocol VLAN Check Rule

If the packet matches the frame type, it will check the Ether-Type is exactly match this protocol entry. The following table illustrates each field of CVLAN protocol entry configuration.

Table 5. VLAN Protocol Entry Setting

Field Name	Bits	Description
VLAN_PPBn_FRAME_TYPE	2	Frame Type: 0b00: Ethernet 0b01: LLC_Other 0b10: RFC1042 0b11:As usage disabled
VLAN_PPBn_VALID	4	Valid port mask for this protocol type
VLAN_PPBn_ETHERTYPE	16	Ether type value

There is independent protocol based CVLAN setting entry for each port. For example, if the packets match the VLAN protocol entry, the protocol VLAN assignment is according to per port configuration.

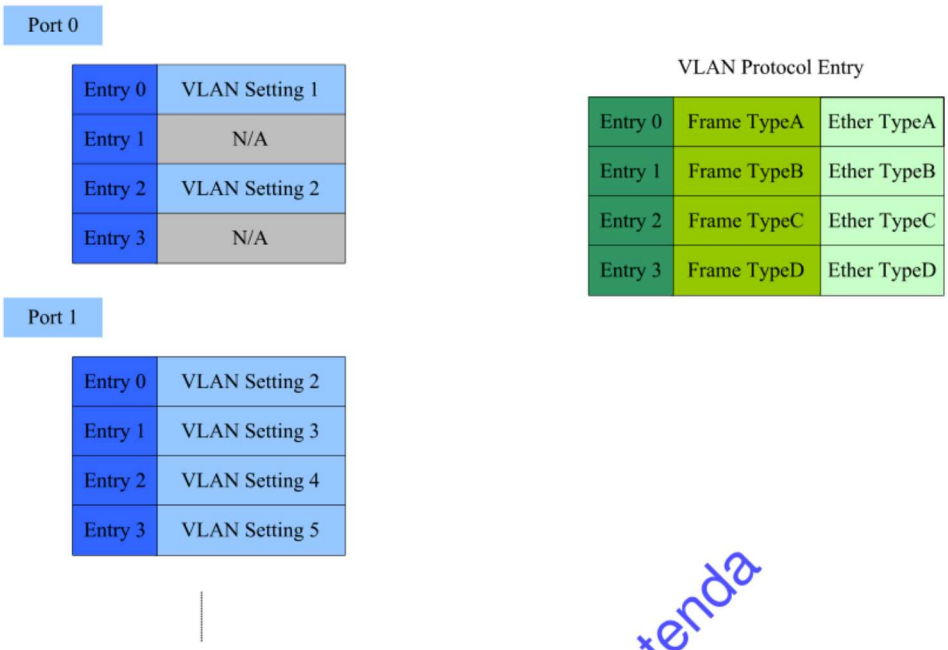

Figure 5. Per Port Protocol VLAN Entry and VLAN Protocol Entry Mapping

- If the packet is received from port 0 and matched the protocol entry 0, this packet will apply to VLAN setting 1.
- If the packet is received from port 0 and matched the protocol entry 1, this packet will not apply the protocol VLAN.
- If the packet is received from port 1 and matched the protocol entry 0, this packet will apply to VLAN setting 2.
- If the packet is received from port 1 and matched the protocol entry 1, this packet will apply to VLAN setting 3.

The following table illustrates each field of Port-and-Protocol-based VLAN configuration.

Table 6. Port-and-Protocol-based VLAN Setting

Field Name	Bits	Description
VLAN_PPBm_PORTn_VID	12	VLAN ID for port n in protocol and port based item m
VLAN_PPBm_ EXTn_VID	12	VLAN ID for EXT 17-0 in protocol and port based item m
VLAN_PPBm_PORTn_PRIORITY	3	Priority for port n in protocol and port based item m
VLAN_PPBm_PORTn_VALID	1	Valid control for port n in protocol and port based item m

11. Ingress CVLAN decision

While a packet was received by system, its CVLAN tag is determined by ASIC. The detail flow chart is shown as below.

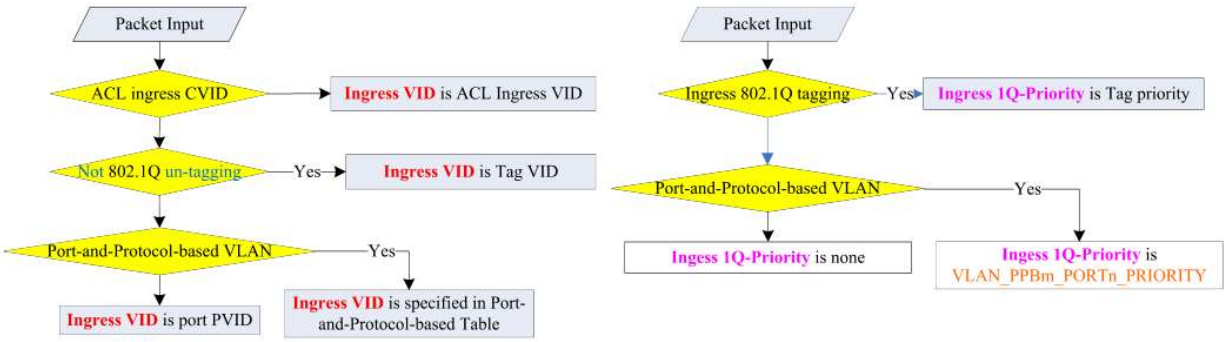

Figure 6. CVLAN Decision Flow

12. Egress CVID decision and egress filtering

In order to avoid the packet is dropped by egress CVLAN filtering function, user need configure VLAN member ports which belong to egress VID that may be modified by ACL egress, L34 and Classification functions. The below figure shows egress CVID decision and CVLAN egress filtering that is according to VLAN member port in VLAN table entry.

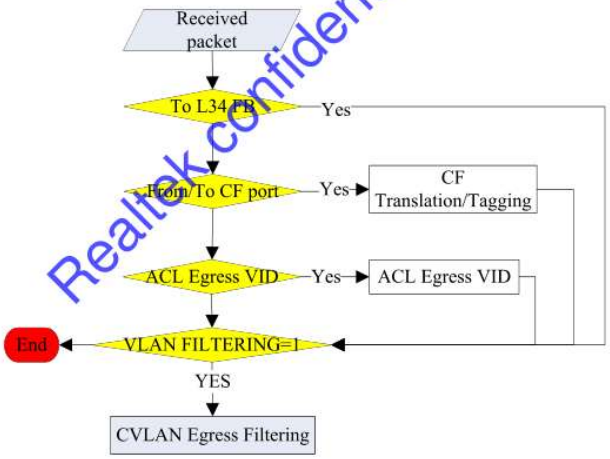

Figure 7. Egress CVID Decision and Filtering Flow

13. CVLAN Leaky

CVLAN provides leaky function for specific traffic type. When leaky type is enabled, this type of packet will not be filtered by CVLAN function. Here list the supported packet type for CVLAN leaky:

For whole system setting:

- Reserved Multicast Address (01-80-C2-00-00-00 ~ 01-80-C2-00-00-2F)

- CSSTP (01-00-0C-CC-CC-CD)
- CDP (01-00-0C-CC-CC-CC)
- IGMP packet

For per port setting:

- IP Multicast packet

The Reserved Multicast Address some address groups share the same setting. The following table is listed the sharing groups.

Table 7. RMA Group Table

RMA Type	Group address range
Bridge Group Address	01-80-C2-00-00-00
IEEE Std 802.3， 1988 Edition， Full Duplex PAUSE operation	01-80-C2-00-00-01
IEEE Std 802.3ad Slow Protocols-Multicast address	01-80-C2-00-00-02
IEEE Std 802.1X PAE address	01-80-C2-00-00-03
Reserved	01-80-C2-00-00-04 ~ 01-80-C2-00-00-07 & 01-80-C2-00-00-09 ~ 01-80-C2-00-00-0C& 01-80-C2-00-00-0F
Provider Bridge Group Address	01-80-C2-00-00-08
Provider Bridge GVRP Address	01-80-C2-00-00-0D
IEEE Std. 802.1AB Link Layer Discovery Protocol multicast address	01-80-C2-00-00-0E
All LANs Bridge Management Group Address	01-80-C2-00-00-10
Load Server Generic Address	01-80-C2-00-00-11
Loadable Device Generic Address	01-80-C2-00-00-12
Reserved	01-80-C2-00-00-13 ~ 01-80-C2-00-00-17 & 01-80-C2-00-00-19 & 01-80-C2-00-00-1B ~ 01-80-C2-00-00-1F
Generic Address for All Manager Stations	01-80-C2-00-00-18
Generic Address for All Agent Stations	01-80-C2-00-00-1A
GMRP Address	01-80-C2-00-00-20
GVRP address	01-80-C2-00-00-21
Undefined GARP address	01-80-C2-00-00-22 ~ 01-80-C2-00-00-2F

14. Accept Frame Type

Per port we can configured the accept frame type. The accept type would be:

- Accept all frames (both tag and un-tag)
- Accept tag frame only

- Accept un-tag frame and priority-tagged frame
- Accept 1Q and 1P tagged frame

15. Action for Reserved VID 0 and VID 4095

For reserved VID 0 and VID 4095 system could set the action for each reserved VID. The action could be:

- RESVID_ACTION_TAG -- The packet will be treat as TAG packet
- RESVID_ACTION_UNTAG -- The packet will be treat as UN-TAG packet

16. Egress CVLAN tag format configuration

The selection of egress packet format is dependent on CVLAN egress type setting. By the way, even if system is disabled CVLAN function, CVLAN tag format of egress packet is still controlled by per port egress mode.

16.1. Egress tag mode

The selection of egress packet format is based on CVLAN tag format mode setting of egress port.

Table 8. Egress Port Tag Mode Setting

Field Name	Bits	Description
EGRESS_MODE	2	Per-port CVLAN tag egress format 0b00: Original mode. Output frame will follow ASIC's CVLAN decision. 0b01: Keep format mode. Output frame will keep CVLAN original format. (tag-in → tag-out, un-tag-in → un-tag-out) 0b10: Priority tag mode. Output frame will be priority tag. 0b11: reserved

- Original mode: egress packet format will follow ASIC original decision
- Keep format mode: egress packet tag format will follow ingress packet CVLAN tag format
  - Ingress packet is un-tag→ egress packet is un-tag
  - Ingress packet is tag→ egress packet is tag
- Priority tag mode: egress packet tag format will always be priority tag
  - Ingress packet with C-priority→ egress packet with ingress C-priority
  - Ingress packet is un-tag→ egress packet with port-based priority

16.2. IP4MC Egress tag mode

In regard to downstream IPv4 multicast from CF port, the egress packet format is based on IP4MC egress tag setting of egress port.

Table 9. Egress Port IP4MC Tag Mode Setting

Field Name	Bits	Description
IP4MC_EGRESS_MODE	2	Per-port CVLAN tag egress format 0b00: Original mode. Output frame will follow ASIC's CVLAN decision. 0b01: Keep format mode. Output frame will keep CVLAN original format. (tag-in → tag-out, un-tag-in → un-tag-out) 0b10: Priority tag mode. Output frame will be priority tag. 0b11: as VLAN_PORTn_EGRESS_MODE setting For non-FB packet and check outer DIP only

16.3. IP6MC Egress tag mode

In regard to downstream IPv6 multicast from CF port, the egress packet format is based on IP6MC egress tag setting of egress port.

Table 10. Egress Port IP6MC Tag Mode Setting

Field Name	Bits	Description
IP4MC_EGRESS_MODE	2	Per-port CVLAN tag egress format 0b00: Original mode. Output frame will follow ASIC's CVLAN decision. 0b01: Keep format mode. Output frame will keep CVLAN original format. (tag-in → tag-out, un-tag-in → un-tag-out) 0b10: Priority tag mode. Output frame will be priority tag. 0b11: as VLAN_PORTn_EGRESS_MODE setting For non-FB packet and check outer DIP6 only

16.4. Keep CFI of CVLAN

About CFI value in CVLAN tag, system provides CVLAN CFI setting to decide the CFI value of egress packet. The CFI setting is 1 bit here list the CFI keep behavior.

Table 11. CVLAN CFI Setting

Field Name	Bits	Description
VLAN_CFI_KEEP	1	Keep ingress tag CFI 0b0: always egress CFI=0 0b1: Keep ingress tag CFI value to egress tag

Table 12. CVLAN CFI Keep Behavior

VLAN_CFI_KEEP	Ingress Packet Format	Egress Packet	Egress CFI
0	Tag	Tag and tag touched	0
		Tag	0
		Un-tag	N/A
	Un-Tag	Tag	0
		Un-tag	N/A
1	Tag	Tag and tag touched	Keep
		Tag	Keep
		Un-tag	N/A
	Un-Tag	Tag	0
		Un-tag	N/A

17. Enable/Disable SVLAN function

The SVLAN function could be global enabled/disabled. When SVLAN function is disabled, the SVLAN ingress/egress filtering function will be disabled. In other words, the packets will not be filtered by SVLAN setting. Both SVLAN member assignment and decision of SVLAN IVL/SVL rest with VLAN table entry. The control register is shown as below table.

Table 13. SVLAN Filtering Setting

Field Name	Bits	Description
VS_FILTERING	1	System SVLAN ingress/egress function setting 0b0: disable SVLAN ingress/egress filtering 0b1: enable SVLAN ingress/egress filtering

18. Aware SVLAN tag

The SVLAN tag can be aware with per-port configured. If the port is configured to be uplink port, ASIC will parse packets to strip STAG (SVLAN tag) and forward packets inside SVLAN which packets belong to. Besides, ASIC supports multi ports to be uplink ports. The control register is shown as below table.

Table 14. Aware SVLAN Tag Setting

Field Name	Bits	Description
VS_PMSK	11	VLAN Stacking tag aware port mask

19. Action for SVLAN un-tagging

The SVLAN module provides configuration for SVLAN un-tagging packet behavior when uplink port receives packet without STAG. Here list the actions for SVLAN un-tagging packet.

Table 15. Action of SVLAN Un-tagging Setting

Field Name	Bits	Description
VS_UNTAG	2	Action for un-stagged packet 0b00: drop 0b01: trap to CPU 0b10: assign ingress SVID as VS_PORTn_SVID 0b11: reserved

20. Port based SVID

Each port and extension port can be assigned the port based SVID. When ingress packet without STAG and SVLAN un-tagging behavior is configured to assigned the port-based SVID, the ingress packet would be inserted port based SVID as STAG. The control registers are shown as below table.

Table 16. Port-based SVID Setting

Field Name	Bits	Description
VS_PORTn_SVID	12	Per ingress port default SVID

VS_EXTn_SVID	12	SVID for EXT 17-0

21. SVLAN decision and filtering behavior

In the SVLAN module, there are different treatments for downstream (packet stream is received from service port) and upstream (packet stream is sent to service port). The below figures illustrate the processing flow for downstream and upstream.

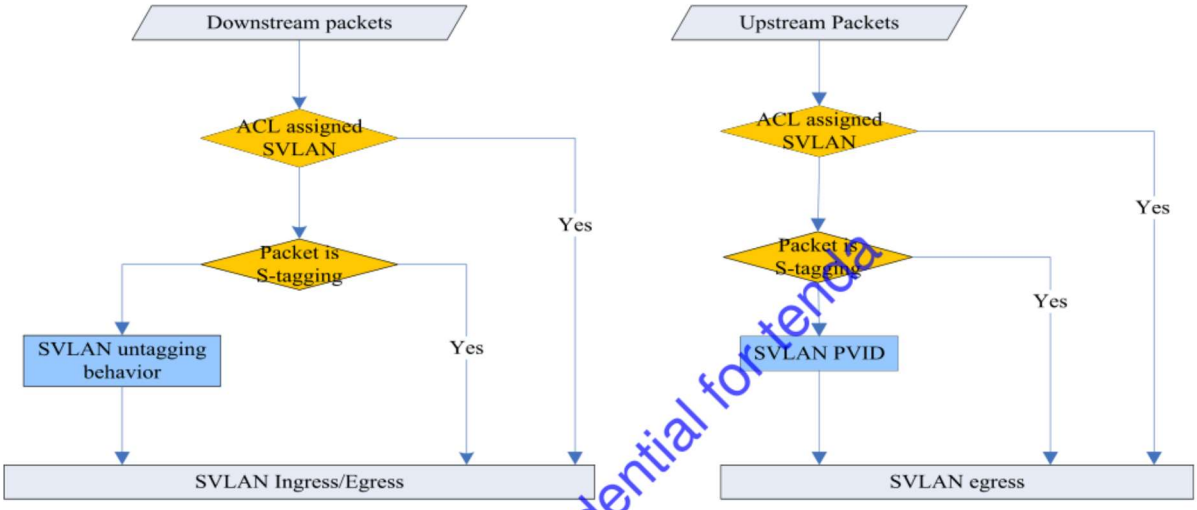

Figure 8. Downstream/Upstream SVLAN Decision and Filtering

The following figure illustrates the processing flow while uplink port receives packet with STAG and CTAG.

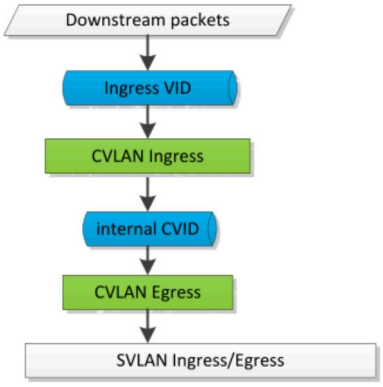

Figure 9. Downstream Filtering Procedure

22. SVLAN forwarding behavior

The following figure shows that the packet with STAG will be aware while receiving from uplink port and ASIC will only forward it in SVLAN 100 member port if SVLAN function is enabled. For this example, system is disabled CVLAN function and configures the member ports of SVLAN 100 to port 0 as well as uplink port, whereof port 0 belong to un-tag set of SVLAN member. In addition, the forwarding decision of this packet is decided by SVLAN and L2 filtering database. While receiving this packet from uplink port, system would retrieve the 100 index entry from VLAN table to look for SVLAN FID/VID if this entry is configured to using SVLAN to check SVL_IVL. Then, L2 filtering database would use SVLAN FID/VID as hash key and look for hash entry in L2 filtering database. If this hash entry is found, the egress port mask of this packet is calculated by SVLAN member port mask AND-operator destination port mask which is decided by L2 filtering database. If there is a STAG in broadcast or unknown packet which is from "not uplink port", ASIC will forward it as normal packet which may be forwarded to all ports. For upstream packet which egress to uplink port, it would be assigned port based SVID if the action of SVLAN un-tagging is configured to port based SVID. Here list formula for SVLAN egress filtering.

- For downstream packet → [(SVLAN member port mask) & (Destination port mask)]
- For upstream packet → {[(!Uplink port Mask) | (SVLAN member port mask)] & (Destination port mask)}

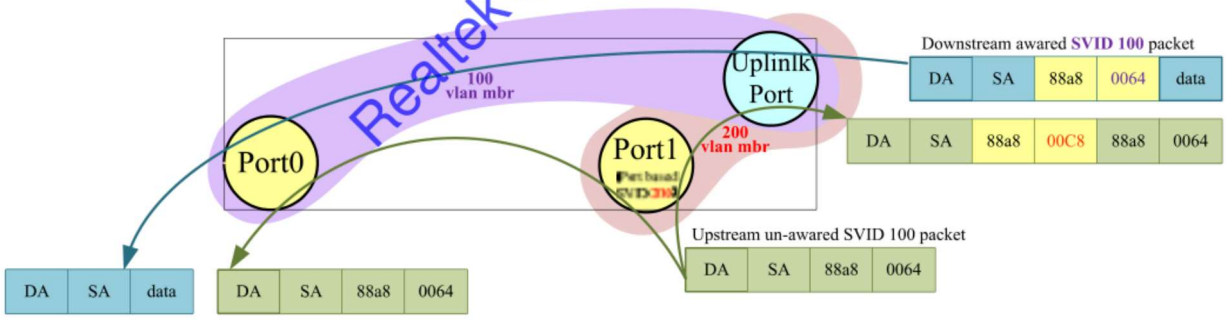

Figure 10. SVLAN Forwarding Behavior

23. SVLAN TPID configuration

There are two entries for SVLAN TPID in RTL9607C and the default TPID (Tag Protocol Identification) of the first entry is 88-A8. These two values of SVLAN TPID can be changed by register. By default, the egress packet to uplink port would be assigned an STAG with TPID is equal to 0x88A8 if uplink port is in tag-set of SVLAN member. For ingress packet from uplink port, the first TPID of STAG is 0x88A8.

Besides, ASIC also supports aware the second TPID of STAG if ASIC enable aware the second SVLAN TPID. Here list the SVLAN TPID configuration.

Table 17. SVLAN TPID Setting

Field Name	Bits	Description
VS_TPID	16	VLAN Stacking Protocol Type (default: 0x88A8), both egress and ingress usage
VS_TPID2_EN	1	Enable aware the second SVLAN TPID
VS_TPID2	16	The second SVLAN TPID configuration, both egress and ingress usage (can used for egress by classification CSACT in upstream)

24. SVLAN trap priority

The SVLAN module provides SVLAN trapping priority assignment for SVLAN un-tagging packet with trap to CPU. The control register is shown as below table.

Table 18. SVLAN Trap Priority Setting

Field Name	Bits	Description
VS_TRAP_PRI	3	SVLAN trapping priority assignment

25. S-priority assignment

For priority value in SVLAN tag, system provides S-priority assignment of received packet. The S-priority assignment is 2 bits here list the S-priority assignment of received packet setting.

Table 19. S-priority Assignment Setting

Field Name	Bits	Description
VS_SPRISEL	2	S-priority assignment 0b00: use internal priority 0b01: use c-tag priority(if ingress cvlan untag frame, then S-priority is from QOS_PORTn_PRIORITY) 0b10: use outer tag priority(if ingress frame is non-tag, then S-priority is from QOS_PORTn_PRIORITY) 0b11: using port based priority QOS_PORTn_PRIORITY.

26. Keep DEI of SVLAN

About DEI value in SVLAN tag, system provides SVLAN DEI setting to decide the DEI value of the egress packet. The SVLAN DEI setting is 1 bit here list behavior for keep SVLAN DEI keep.

Table 20. SVLAN DEI Setting

Field Name	Bits	Description
VS_DEI_KEEP	1	Keep SVLAN ingress tag DEI 0b0: Always egress DEI=0 0b1: Keep ingress tag DEI value to egress tag

27. SP2C configuration

There are 64 entries in SP2C table. It is used for the downstream classification CSVID/CVID action which is configured to translation with SP2C table. The below table would describe each field of SP2C entry.

Table 21. SP2C Entry Setting

Field Name	Bits	Description
VALID	1	Valid bit
Ingress VID	12	Need translated VID
DSTPORT	2	Destination port number of downstream packet
Egress VID	12	Egress VID of CTAG or STAG
PRIORITY	3	Egress priority of CTAG or STAG

28. API

Realtek API provides a series of interface to let users setup the CVLAN/SVLAN function without writing register and table directly. This section will discuss these APIs and gives the example.

28.1. Initialization

rtk_vlan_init is the first API users should call before setup any configuration. VLAN init will apply following setting.

- Enable VLAN function
- Per port enable ingress VLAN filtering function
- Create default VLAN as VID 1, and assign all port to this VID
- Accept all from type for all port
- Disable all leaky function
- Disable protocol VLAN function
- Egress tag mode set to “Original mode”
- Disable keep CFI
- Reserve VID 0 and VID 4095 action set to Un-Tag

rtk_svlan_init is the first API users should call before setup any configuration. SVLAN init will apply following setting.

- Invalid 64 entries of SP2C table
- Disable service port for all port

- Disable keep DEI function
- SLVAN un-tagging action set to “Drop”
- Disable the second TPID
- Disable SVLAN function

28.2. Enable VLAN Function

The ***rtk_vlan_vlanFunctionEnable_set*** API will enable or disable VLAN function.

Example:

```
/*
   Enalbe VLAN filter function
*/

int32 ret;

rtk_enable_t enable = ENABLED;

if ((ret = rtk_vlan_vlanFunctionEnable_set(enable)) != RT_ERR_OK)
{
    return ret;
}
```


28.3. Create/Delete VLAN entry

The ***rtk_vlan_create*** API would create a VLAN entry.

The ***rtk_vlan_destroy*** API would destroy a VLAN entry.

The ***rtk_vlan_destroyAll*** API would destroy all VLAN entries include or exclude default VLAN.

Example:

```
/* Crate VID 100 */

int32 ret;

if ((ret = rtk_vlan_create(100)) != RT_ERR_OK)
{
    return ret;
}
```

```
/* Destroy VID 100 */

int32 ret;

if ((ret = rtk_vlan_destroy(100)) != RT_ERR_OK)
{
    return ret;
}
```

```
/* Destroy all VLAN entry, but restore default vlan*/

int32 ret;
uint32 restoreDefaultVlan = 1;

if ((ret = rtk_vlan_destroyAll(restoreDefaultVlan)) != RT_ERR_OK)
{
    return ret;
}
```


28.4. Assign VLAN member

The ***rtk_vlan_port_set*** API will assign the VLAN member to the given VID. This API don't care the original VLAN members and replace with new configure directly.

For extension port please use ***rtk_vlan_extPortmaskIndex_set*** and ***rtk_vlan_extPortmaskCfg_set.***

Example:

```
/*
  Assign all port to VID 100
  Assign UTP Port 0 & 1 to VLAN 100 un-tag set
*/

rtk_portmask_t memberPortmask;
rtk_portmask_t untagPortmask;
int32 ret;

rtk_switch_allPortMask_set(&memberPortmask);
RTK_PORTMASK_RESET(&untagPortmask);
```

```
rtk_switch_port2PortMask_set(&untagPortmask, RTK_PORT_UTP0);
rtk_switch_port2PortMask_set(&untagPortmask, RTK_PORT_UTP1);

if ((ret = rtk_vlan_port_get(100, &memberPortmask, &untagPortmask)) !=
RT_ERR_OK)
{
    return ret;
}
```

28.5. Port based VLAN Assign

The ***rtk_vlan_portPvid_set*** API will assign the port based VID to the given port id.

For extension port please use ***rtk_vlan_extPortPvid_set.***

Example:

```
/*
    Set UTP Port 2 port-based vid to 200
*/

int32 ret;
uint32 port;

rtk_switch_phyPortId_get(RTK_PORT_UTP2, &port);
if ((ret = rtk_vlan_portPvid_set(port, 200)) != RT_ERR_OK)
{
    return ret;
}
```


28.6. VLAN IVL/SVL Assign

The ***rtk_vlan_fidMode_set*** API will assign the IVL/SVL mode to the given VID. Using ***rtk_vlan_fid_set*** API would assign the fid for given VID.

Example:

```
/*
    Set VLAN 100 to IVL mode
    Set VLAN 200 to SVL mode and assign FID to 1
    Set VLAN 300 to SVL mode and assign FID to 2
*/
```

```
int32 ret;

if ((ret = rtk_vlan_fidMode_set(100, VLAN_FID_IVL)) != RT_ERR_OK)
{
    return ret;
}
if ((ret = rtk_vlan_fidMode_set(200, VLAN_FID_SVL)) != RT_ERR_OK)
{
    return ret;
}
if ((ret = rtk_vlan_fidMode_set(300, VLAN_FID_SVL)) != RT_ERR_OK)
{
    return ret;
}

if ((ret = rtk_vlan_fid_set(200, 1)) != RT_ERR_OK)
{
    return ret;
}
if ((ret = rtk_vlan_fid_set(300, 2)) != RT_ERR_OK)
{
    return ret;
}
```


28.7. Decision IVL and SVL

The ***rtk_vlan_lutSvlanHashState_set*** API will force to assign the IVL/SVL mode based on SVLAN or does NOT check SVLAN IVL/SVL setting, so the IVL/SVL mode based on CVLAN.

Example:

```
/*
   Set SVLAN 400 to check IVL/SVL setting
*/

int32 ret;

if ((ret = rtk_vlan_lutSvlanHashState_set(400, ENABLED)) != RT_ERR_OK)
{
    return ret;
```

```
}
```

28.8. VLAN Ingress Filter

The ***rtk_vlan_portIgrFilterEnable_set*** API will enable/disable ingress VLAN filter function. This function is enabled by default.

Example:

```
/*
   Disable ingress VLAN filter for  UTP Port 0~1
*/

int32 ret;
uint32 port;

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
if ((ret = rtk_vlan_portIgrFilterEnable_set(port, DISABLED)) !=
RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &port);
if ((ret = rtk_vlan_portIgrFilterEnable_set(port, DISABLED)) !=
RT_ERR_OK)
{
    return ret;
}
```


28.9. VLAN Leaky

For VLAN leaky function system per port based (***vlan_portLeaky_set***) and system based (***rtk_vlan_leaky_set***) leaky API. The leaky function is disabled by default.

For port based leaky function support LEAKY_IPMULTICAST.

For system based leaky function support

LEAKY_BRG_GROUP

LEAKY_FD_PAUSE

LEAKY_SP_MCAST

LEAKY_1X_PAE

LEAKY_UNDEF_BRG_04

LEAKY_UNDEF_BRG_05

LEAKY_UNDEF_BRG_06

LEAKY_UNDEF_BRG_07

LEAKY_PROVIDER_BRIDGE_GROUP_ADDRESS

LEAKY_UNDEF_BRG_09

LEAKY_UNDEF_BRG_0A

LEAKY_UNDEF_BRG_0B

LEAKY_UNDEF_BRG_0C

LEAKY_PROVIDER_BRIDGE_GVRP_ADDRESS

LEAKY_8021AB

LEAKY_UNDEF_BRG_0F

LEAKY_BRG_MNGEMENT

LEAKY_UNDEFINED_11

LEAKY_UNDEFINED_12

LEAKY_UNDEFINED_13

LEAKY_UNDEFINED_14

LEAKY_UNDEFINED_15

LEAKY_UNDEFINED_16

LEAKY_UNDEFINED_17

LEAKY_UNDEFINED_18

LEAKY_UNDEFINED_19

LEAKY_UNDEFINED_1A

LEAKY_UNDEFINED_1B

LEAKY_UNDEFINED_1C

LEAKY_UNDEFINED_1D

LEAKY_UNDEFINED_1E

LEAKY_UNDEFINED_1F

LEAKY_GMRP

LEAKY_GVRP

LEAKY_UNDEF_GARP_22

LEAKY_UNDEF_GARP_23

LEAKY_UNDEF_GARP_24

LEAKY_UNDEF_GARP_25

LEAKY_UNDEF_GARP_26

LEAKY_UNDEF_GARP_27

LEAKY_UNDEF_GARP_28

LEAKY_UNDEF_GARP_29

LEAKY_UNDEF_GARP_2A

LEAKY_UNDEF_GARP_2B

LEAKY_UNDEF_GARP_2C

LEAKY_UNDEF_GARP_2D

LEAKY_UNDEF_GARP_2E

LEAKY_UNDEF_GARP_2F

LEAKY_IGMP

LEAKY_CDP

LEAKY_SSTP

Example:

```
/*
   Enable IGMP VLAN Leaky
   UTP port 0 ~ UTP port 1 enable IPMULTICAST leaky
*/

int32 ret;
uint32 port;

if ((ret = rtk_vlan_leaky_set(LEAKY_IGMP, ENABLED)) != RT_ERR_OK)
{
```

```
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
if ((ret = rtk_vlan_portLeaky_set(port, LEAKY_IPMULTICAST, ENABLED)) !=
RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &port);
if ((ret = rtk_vlan_portLeaky_set(port, LEAKY_IPMULTICAST, ENABLED)) !=
RT_ERR_OK)
{
    return ret;
}
```

28.10. Protocol VLAN

The ***rtk_vlan_protoGroup_set*** API can set the protocol VLAN entry. The ***rtk_vlan_portProtoVlan_set*** API can bind the port to protocol entry and assign the VLAN parameter for this protocol on this port.

Example:

```
/*
   Set protocol VLAN entry index 0:
       Frame type : FRAME_TYPE_ETHERNET
       Ether Type :0x0800

   Set protocol VLAN entry index 1:
       Frame type : FRAME_TYPE_ETHERNET
       Ether Type :0x8864

   UTP Port 0 binding protocol entry 0 to VID 200
   UTP Port 0 binding protocol entry 1 to VID 500
   UTP Port 1 binding protocol entry 0 to VID 1200
   UTP Port 1 binding protocol entry 1 to VID 1500
*/

int32 ret;
uint32 port;
```

```
rtk_vlan_protoGroup_t   protoGroup;
rtk_vlan_protoVlanCfg_t vlanCfg;

protoGroup.frametype = FRAME_TYPE_ETHERNET;
protoGroup.framevalue= 0x0800;
if ((ret = rtk_vlan_protoGroup_set (0, &protoGroup)) != RT_ERR_OK)
{
    return ret;
}

rtk_vlan_protoGroup_t   protoGroup;
protoGroup.frametype = FRAME_TYPE_ETHERNET;
protoGroup.framevalue = 0x8864;
if ((ret = rtk_vlan_protoGroup_set(1, &protoGroup)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
vlanCfg. valid = 1;
vlanCfg. vid   = 200;
vlanCfg. pri   = 2;
if ((ret = rtk_vlan_portProtoVlan_set(port, 0, &vlanCfg)) != RT_ERR_OK)
{
    return ret;
}

vlanCfg. valid = 1;
vlanCfg. vid   = 500;
vlanCfg. pri   = 2;

if ((ret = rtk_vlan_portProtoVlan_set(port, 1, & vlanCfg)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &port);
vlanCfg. valid = 1;
vlanCfg. vid   = 1200;
vlanCfg. pri   = 2;
```

```
if ((ret = rtk_vlan_portProtoVlan_set(port, 0, &vlanCfg)) != RT_ERR_OK)
{
    return ret;
}

vlanCfg. valid = 1;
vlanCfg. vid   = 1500;
vlanCfg. pri   = 2;

if ((ret = rtk_vlan_portProtoVlan_set(port, 1, &vlanCfg)) != RT_ERR_OK)
{
    return ret;
}
```

28.11. Egress VLAN Tag Format

The ***rtk_vlan_tagMode_set*** API can set egress tag format for given egress port.

The tag mode would be:

VLAN_TAG_MODE_ORIGINAL (depend on chip normal decision)

VLAN_TAG_MODE_KEEP_FORMAT (keep ingress format to egress)

VLAN_TAG_MODE_PRI (always priority tag out)

Example:

```
/*
   UTP Port 0 egress tag always priority tag
   UTP Port 1 egress tag always follow ASIC original decision
*/

int32 ret;
uint32 port;

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
if ((ret = rtk_vlan_tagMode_set(port, VLAN_TAG_MODE_PRI)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &port);
```

```
if ((ret = rtk_vlan_tagMode_set(port, VLAN_TAG_MODE_ORIGINAL)) !=
RT_ERR_OK)
{
    return ret;
}
```

28.12. Egress VLAN Tag Format

The ***rtk_vlan_portAcceptFrameType_set*** API can per port set VLAN accept frame type.

The frame type would be:

ACCEPT_FRAME_TYPE_ALL

ACCEPT_FRAME_TYPE_TAG_ONLY

ACCEPT_FRAME_TYPE_UNTAG_ONLY

ACCPET_FRAME_TYPE_1P_1Q_TAG_ONLY

Example:

```
/*
   UTP Port 0 admit all frames
   UTP Port 1 1Q and 1P tagged frame (VID 0 ~ 4095)
*/

int32 ret;
uint32 port;

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
if ((ret = rtk_vlan_portAcceptFrameType_set(port,
ACCEPT_FRAME_TYPE_ALL)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &port);
if ((ret = rtk_vlan_portAcceptFrameType_set (port,
ACCPET_FRAME_TYPE_1P_1Q_TAG_ONLY)) != RT_ERR_OK)
{
    return ret;
}
```

28.13. Reserve VID 0 and 4095 Type

The ***rtk_vlan_reservedVidAction_set*** API can set VID 0 or VID 4095 as TAG or UN-TAG respectively.

Example:

```
/*
   VID 0 type is TAG
   VID 4095 type is UNTAG
*/

int32 ret;
uint32 port;

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
if ((ret = rtk_vlan_reservedVidAction_set(RESVID_ACTION_TAG,
RESVID_ACTION_UNTAG)) != RT_ERR_OK)
{
    return ret;
}
```


28.14. Basic SVLAN configuration

This section gives some examples such as enable SVLAN ingress/egress filtering function, create a SVLAN, assign SVLAN member port, and aware service port. The ***rtk_svlan_svlanFunctionEnable_set*** API can enable or disable SVLAN ingress/egress filtering. The ***rtk_svlan_servicePort_set*** API can per port enable which port need to be SVLAN service port. The ***rtk_svlan_create*** API can create an entry of VLAN table. The ***rtk_svlan_memberPort_set*** API can assign member ports for the created SVLAN. The following sample code shows that system enable SVLAN filtering function, create a VLAN entry of SVID 1000 for default SVLAN in all ports, and setup RTK_PORT_PON as service port.

Example:

```
/* SVID 1000 */
int32                   ret = RT_ERR_FAILED;
uint32                  port;
rtk_portmask_t          memberPortmask;
rtk_portmask_t          untagPortmask;
rtk_switch_port_name_t portName;

if ((ret = rtk_svlan_init()) != RT_ERR_OK)
```

```
{
    return ret;
}

if ((ret = rtk_svlan_svlanFunctionEnable_set(ENABLED)) != RT_ERR_OK)
{
     return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_PON, &port);
if ((ret = rtk_svlan_servicePort_set(port, ENABLED)) != RT_ERR_OK)
{
    return ret;
}

if ((ret = rtk_svlan_create(1000)) != RT_ERR_OK)
{
     return ret;
}

rtk_switch_allPortMask_set(memberPortmask);
rtk_switch_allPortMask_set(untagPortmask);
rtk_switch_port2PortMask_clear(&untagPortmask, RTK_PORT_PON);
if ((ret = rtk_svlan_memberPort_set(1000, &memberPortmask,
&untagPortmask)) != RT_ERR_OK)
{
     return ret;
}

for (portName = RTK_PORT_UTP0; portName <= RTK_PORT_PON; portName++)
{
     rtk_switch_phyPortId_get(portName, &port);
     if ((ret = rtk_svlan_portSvid_set(port, 1000)) != RT_ERR_OK)
     {
         return ret;
     }
}
```

28.15. Sample code

This section gives combined examples while enable SVLAN/CVLAN ingress/egress filtering function. The following sample code show that it is possible for one UTP port is transparent SVLAN and the others UTP ports are normal de-tagging SVLAN while received downstream without CTAG from uplink port. Besides, system will drop all un-tagging downstream packets.

Example:

```
/* SVID 2000 is transparent in UTP port 0 and PON port. Drop all ingress
untagging downtream pacekts */
int32                   ret = RT_ERR_FAILED;
uint32                  port;
rtk_portmask_t          memberPortmask;
rtk_portmask_t          untagPortmask;

if ((ret = rtk_svlan_init()) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_vlan_init()) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_svlan_svlanFunctionEnable_set(ENABLED)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_PON, &port);
if ((ret = rtk_svlan_servicePort_set(port, ENABLED)) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_svlan_create(2000)) != RT_ERR_OK)
{
    return ret;
```

```
}

rtk_switch_allPortMask_set(memberPortmask);
rtk_switch_allPortMask_set(untagPortmask);
rtk_switch_port2PortMask_clear(&untagPortmask, RTK_PORT_PON);
rtk_switch_port2PortMask_clear(&untagPortmask, RTK_PORT_UTP0);

if ((ret = rtk_svlan_memberPort_set(2000, &memberPortmask,
&untagPortmask)) != RT_ERR_OK)
{
    return ret;
}

if ((ret = rtk_svlan_untagAction_set(SVLAN_ACTION_DROP, 0)) != RT_ERR_OK)
{
    return ret;
}

if ((ret = rtk_vlan_create(200)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_allPortMask_set(&memberPortmask);
rtk_switch_allPortMask_set(&untagPortmask);
if ((ret = rtk_vlan_port_get(200, &memberPortmask, &untagPortmask)) !=
RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_PON, &port);
if ((ret = rtk_vlan_portPvid_set(port, 200)) != RT_ERR_OK)
{
    return ret;
}
```


The following sample code shows the configuration of SVLAN and CVLAN as below:

For CVLAN, system enable CVLAN ingress/egress filtering and create CVID 300 for default CVLAN in PON port, whereof all port belong to the member port and un-tag-set of CVID 300. For SVLAN, system

enable SVLAN ingress/egress filtering, setup RTK_PORT_PON as service port, create SVID 3000 for default SVLAN in UTP port 0, assign all port in member port of SVID 3000, and only PON port doesn't belong to un-tag-set of SVID 3000. According to above configuration, the egress format of upstream packet consists of SVID 3000 but doesn't include CTAG. The downstream packet with SVID 3000 would be forwarded to UTP port 0 if its destination MAC were already learned form source UTP port 0 and its egress format is untagged.

Example:

```
int32                   ret = RT_ERR_FAILED;
uint32                  port;
rtk_portmask_t          memberPortmask;
rtk_portmask_t          untagPortmask;

if ((ret = rtk_svlan_init()) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_vlan_init()) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_svlan_svlanFunctionEnable_set(ENABLED)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_PON, &port);
if ((ret = rtk_svlan_servicePort_set(port, ENABLED)) != RT_ERR_OK)
{
   return ret;
}

if ((ret = rtk_svlan_create(3000)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &port);
```

```
if ((ret = rtk_svlan_portSvid_set(port, 3000)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_allPortMask_set(&memberPortmask);
rtk_switch_allPortMask_set(&untagPortmask);
rtk_switch_port2PortMask_clear(&untagPortmask, RTK_PORT_PON);
if ((ret = rtk_svlan_memberPort_set(3000, &memberPortmask,
&untagPortmask)) != RT_ERR_OK)
{
    return ret;
}

if ((ret = rtk_vlan_create(300)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_allPortMask_set(&memberPortmask);
rtk_switch_allPortMask_set(&untagPortmask);
if ((ret = rtk_vlan_port_get(300, &memberPortmask, &untagPortmask)) !=
RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_PON, &port);
if ((ret = rtk_vlan_portPvid_set(port, 300)) != RT_ERR_OK)
{
    return ret;
}
```


Realtek Semiconductor Corp.
Headquarters
No. 2, Innovation Road II, Hsinchu Science Park,
Hsinchu 300, Taiwan, R.O.C.
Tel: 886-3-5780211 Fax: 886-3-5776047
www.realtek.com

全文阅读已结束，下载本文需要使用

400 积分

下载此文档

阅读了该文档的用户还阅读了这些文档

RTL9607C_LED_Apr

RTL9607C_RTK_Por

RTL9607C_L2_Table

RTL9607C_Storm_Fi

EngNote_RTL9607C note_V01(2019111

Application_Nc

发表评论

验证码： 换一张

匿名评论

提交