RTL9607C
SINGLE-CHIP PON

Classification Application Note
(CONFIDENTIAL: Development Partners Only)

Rev. 1.0.0
2 June 2017

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	2017/06/02	First Release

Table of Contents

List of Tables

List of Figures

1. Overview

The Classification is an ingress filtering function. This function works when a packet received at specified ports. Once the packet is matching one classification rule, corresponding actions will be taken. Entire Classification function includes 2 main components, Rules (including data & mask) and Actions. The relationship between them and a sample configuration is as following:

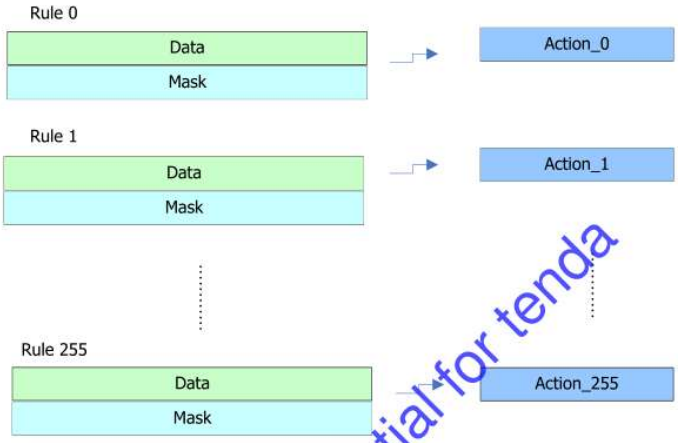


Figure 1. Classification Block Diagram

Classification will use data and mask in rule and compare the specified field in a packet. Each classification rule is corresponding to a single entry in Action table. When an incoming packet which is matched a classification rule, the corresponding action in Action table will be taken.

2. Classification Configuration

The upstream or downstream direction in Classification function is decided by the ingress port and egress port. If the ingress port is CF port, the direction is downstream. If the ingress port is non-CF port, and the egress port is CF port, the direction is upstream. CF port can be configured per port basis.

Table 1. Per port Classification function enable configuration

Field Name	Bits	Description
CF_SEL_Pn_EN	1	Classification port state 0b0: disable 0b1: enable

For those packets are not matched any Classification rules in upstream direction, CF_US_PERMIT, will be applied on them. If user sets this configuration to “Permit as normal forward”, all upstream packets can be forwarded, but the actions specified by Classification would only apply on matched packets. If used “permit without PON port forwarding”, all upstream packets without matching any Classification rules, will not be forwarded to PON port. For downstream packets without matching Classification rules, they will be forwarded by switch core decision.

Table 2. Classification Upstream permit Configuration

Field Name	Bits	Description
CF_US_PERMIT	2	Permit upstream packet which is not hit to any entries. The actions are as below. 0b0: permit as normal forward 0b1: permit without CF_SEL_Pn_EN enabled port forwarding

3. Rules

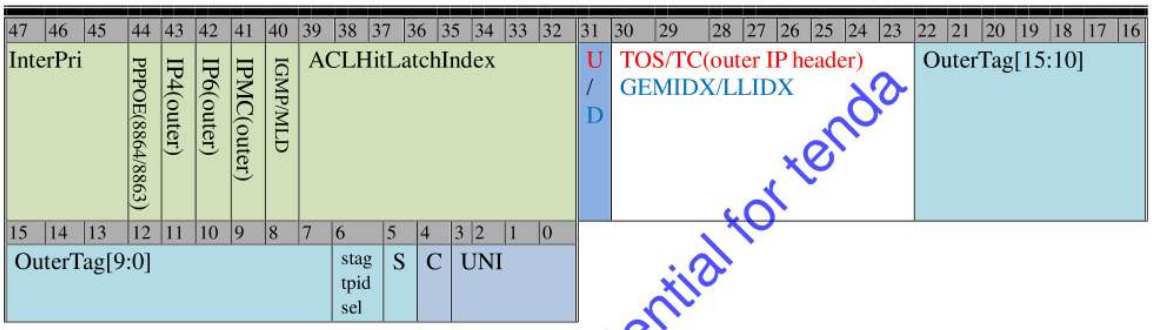
256 Classification rules are supported in switch. Each rule includes 48 bits. There are 3 templates for each rule to select.

Table 3. Entry Template Configuration

Field Name	Bits	Description
CF_TEMPLATE_SEL	2	Select the template of entry

Template 0 rules are provided for matching upstream/downstream packet. The data/mask is as following:

Table 4. Classification Template 0 rule definition



Field[47:45]: Internal priority from switch core.

Field bit[44]: PPPoE packet

Field bit[43]: IPv4 packet

Field bit[42]: Ipv6 packet

Field bit[41]: IP multicast packet

Field bit[40]: IGMP or MLD packet

Field[39:32]: ACLHitLatchIndex, filter packet which is matched by ACL entry 0-127 and the action CFHITLATCH of ACL entry 0-127 is set. Value 255 is a special definition and it indicates filtering packet which is not matched by any ACL rule.

Field[31]: 1: filter downstream packet. 0: filter upstream packet.

Field[30:23]:

- Upstream packet: TOS or TC value in IP header
- GPON downstream packet: GEMIDX
- EPON downstream packet: MPCP[27]+LLID[26:23]

Field[22:7]: Outer tag. For S-tag, it is {SPRI/DEI/SVID}. For C-tag, it is {PRI/CFI/CVID}

Field[6]:

- Mask = 0: both VS_TPID and VS_TPID2 will be treated as S-tag TPID
- Mask = 1 and data = 0: only VS_TPID will be treated as S-tag TPID
- Mask = 1 and data = 1: only VS_TPID2 will be treated as S-tag TPID

Field[5]: S-tag packet

Field[4]: C-tag packet

Field [3:0]:

- Upstream packet: port number of packet received.
- Downstream pakcet: destination port number of packet, 0b1111 indicates filtering flood packet.

Template 1 & 2 are as following:

Table 5. Classification Template 1 definition

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EtherType																U/D	TOS/TC/GEMIDX/LLIDX								OuterTag[15:10]						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OuterTag[9:0]									stag tpid sel	S	C	UNI			

Table 6. Classification Template 2 definition

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Ctag(Priority+CFI+VID)																U/D	TOS/TC/GEMIDX/LLIDX								OuterTag[15:10]						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OuterTag[9:0]									stag tpid sel	S	C	UNI			

Field[47:32] in Template 1 is Ethertype. In Template 2, it is ingress Ctag {VID+CFI+Priority}.

4. Action

There are 256 entries in Action table and these entries are one by one mapped to Classification Rule Table directly. For example, if a packet matched Classification rule 28, the action specified in entry 28 of Action table will be applied to this packet.

4.1. Upstream Action

When the incoming packet matched the upstream classification rule, it can use the upstream action as below.

Table 7. Upstream action

Field Name	Bits	Description
CSACT	3	0b000: nop (follow switch-core) 0b001: add classification tag which TPID as VS_TPID (reference to CSVID_ACT/CSPRI_ACT) 0b010: add classification tag which TPID as VS_TPID2 (reference to CSVID_ACT/CSPRI_ACT) 0b011: delete Stag 0b100: transparent 0b101: add classification tag which TPID as original Stag(if without Stag, using VS_TPID) Other: reserved
CS_VID	12	Assigned VID
CS_PRI	3	Assigned P-bits
CSVID_ACT	2	0b00: nop 0b01: Assigned to VID 0b10: Copy from 1st tag VID(if none 1st tag, then as CS_VID) 0b11: Copy from 2nd tag VID(if none 2nd tag, then asCS_VID)
CSPRI_ACT	3	0b000: nop 0b001: Assigned to CSPRI 0b010: Copy from 1st tag P-bits(if none 1st tag, then as CS_PRI) 0b011: Copy from 2nd tag P-bits(if none 2nd tag, then as CS_PRI) 0b100: Assign from internal priority 0b101: Assign from DSCP-based Pirority Assignment table(If no DSCP field, used Assigned CS_PRI, DSCP is got from outer IP header) Other: reserved
CACT	2	0b00: nop 0b01: tagging(reference to CVID_ACT/CPRI_ACT) 0b10: un-tagging 0b11: transparent
CVID	12	Assigned VID
CPRI	3	Assigned P-bits
CVID_ACT	3	0b000: nop 0b001: Assigned to VID 0b010: Copy from 1st tag VID (if tag is not existed, then using CVID) 0b011: Copy from 2nd tag VID (if tag is not existed, then using CVID) Other: reserved
CPRI_ACT	3	0b000: nop

		0b001: Assigned to CPRI 0b010: Copy from 1st tag P-bits (if tag is not existed, then using CPRI) 0b011: Copy from 2nd tag P-bits (if tag is not existed, then using CPRI) 0b100: Assign from internal priority 0b101: Assign from DSCP-based Pirority Assignment table(If no DSCP field, used Assigned CPRI, DSCP is got from outer IP header)) Other: reserved
SID_ACT	1	0b0:nop 0b1:Assign to SID per GPON/LLID per EPON
ASSIGN_IDX	7	Assigned PON MAC SID/LLID
FORWARD_ACT	2	0b00: nop 0b01: drop 0b10: trap to CPU 0b11: drop packet to pon

4.2. Downstream Action

When the incoming packet matched the downstream classification rule, it can use the downstream action as below.

Table 8. Downstream action

Field Name	Bits	Description
CSACT	3	0b000: nop(follow switch core) 0b001: add classification tag which TPID as VS_TPID (reference to CSVID_ACT/CSPRI_ACT) 0b010: add classification tag which TPID as VS_TPID2 (reference to CSVID_ACT/CSPRI_ACT) 0b011: delete Stag 0b100: transparent 0b101: add classification tag which TPID as original Stag(if without Stag, using VS_TPID) Other: reserved
CS_VID	12	Assigned VID
CS_PRI	3	Assigned P-bits
CSVID_ACT	3	0b000: nop 0b001: Assigned to VID 0b010: Copy from 1st tag VID(if none 1st tag, then as CS_VID) 0b011: Copy from 2nd tag VID(if none 2nd tag, then as CS_VID) 0b100: Translation with SP2C table with input VID is 1st tag VID or ingress CVID, and replace the Stag VID (if unhit SP2C will be untag) Other:reserved
CSPRI_ACT	3	0b000: nop 0b001: Assigned to CSPRI 0b010: Copy from 1st tag P-bits(if none 1st tag, then as CS_PRI) 0b011: Copy from 2nd tag P-bits (if none 2nd tag, then as CS_PRI) 0b100: Assign from internal priority 0b101: Assign from DSCP-based Pirority Assignment table(If no DSCP field, used Assigned CS_PRI, DSCP is got from outer IP header)) 0b110: Translation with SP2C table with input VID is 1st tag VID or ingress CVID, and replace the Stag PRI (if unhit SP2C will used as nop)

		Other:reserved
CACT	2	0b00: nop(following switch-core) 0b01: tagging (reference to CVID_ACT/CPRI_ACT) 0b10: un-tagging 0b11: transparent
CVID	12	Assigned tag VID
CPRI	3	Assigned tag P-bits
CVID_ACT	3	0b000: nop 0b001: Assigned to VID 0b010: Copy from 1st tag VID(if none 1st tag, then as CVID) 0b011: Copy from 2nd tag VID(if none 2nd tag, then as CVID) 0b100: Translation with SP2C table with input VID is 1st tag VID or ingress CVID, and replace the Ctag VID (if unhit will be untag) 0b101: Egress CVLAN VID by LUT MAC VID learning(if lookup miss, the packet will be C untag) Other:reserved
CPRI_ACT	2	0b000: nop 0b001: Assigned to PRI 0b010: Copy from 1st tag Priority(if none 1st tag, then as CPRI) 0b011: Copy from 2nd tag Priority(if none 2nd tag, then as CPRI) 0b100: Assign from internal priority 0b101: Assign from DSCP-based Priority Assignment table(If no DSCP field, used Assigned CPRI, DSCP is got from outer IP header)) 0b110: Translation with SP2C table with input VID is 1st tag VID or ingress CVID, and replace the Ctag PRI (if unhit will be nop) Other:reserved
CFPRI_ACT	1	Classification priority assignment for packets 0b0:nop 0b1:Forced CF internal priority to CFPRI
CFPRI	3	Assigned classification priority
FORWARD_ACT	2	0b00:nop 0b01:forced forward to UNI_MASK 0b10:trap to CPU. 0b11: forwarding member mask to UNI_MASK only
UNI_MASK	4	forced forward/flooding port mask

CFPRI_ACT set to 0 means internal priority follow switch core, and set to 1 means force assigned an internal priority to CFPRI.

FORWARD_ACT control the forwarding behavior of the matched packet. Set to 0, will follow switch core forwarding. Set to 1, will force to forward the matched packet to UNI_MASK. Set to 3, the forwarding mask will be limited by UNI_MASK.

4.3. Multiple Rules Action

Classification supports only one rules action. If more than one rule is matched, only the action specified in first matched rule will be taken as packet's action.

5. API

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

5.1. Initialization

rtk_classify_init is the first API users should call before setup any configuration. This API will clear all Classification rules and actions.

5.2. Rules

Use ***rtk_classify_field_add*** API to add the field in the link list of rule structure.

Example:

```
/* Configure Downstream rule for tag-vid 1000, tag priority 5 */

rtk_classify_cfg_t  entry;
rtk_classify_field_t *pField_vid, *pField_pri;
int32  ret;

rtk_classify_init();

memset(&entry, 0, sizeof(rtk_classify_cfg_t));

entry.index = 128;
entry.direction = CLASSIFY_DIRECTION_DS;
entry.valid = ENABLED;
entry.templateIdx = 0;

pField_vid = (rtk_classify_field_t *) alloc(sizeof(rtk_classify_field_t));
pField_vid->fieldType = CLASSIFY_FIELD_TAG_VID;
pField_vid->classify_pattern.tagVid.value = 1000;
pField_vid->classify_pattern.tagVid.mask = 0xfff;
if((ret = rtk_classify_field_add(&entry, pField_vid)) != RT_ERR_OK)
    return ret;
```

```
pField_pri = (rtk_classify_field_t *) alloc(sizeof(rtk_classify_field_t));
pField_pri->fieldType = CLASSIFY_FIELD_TAG_PRI;
pField_pri->classify_pattern.tagPri.value = 5;
pField_pri->classify_pattern.tagPri.mask = 7;
if((ret = rtk_classify_field_add(&entry, pField_pri)) != RT_ERR_OK)
    return ret;

if((ret = rtk_classify_cfgEntry_add(&entry)) != RT_ERR_OK)
    return ret;

free(pField_vid);
free(pField_pri);
```

5.3. Actions

Fill the Classification action data structure directly.

Example:

```
/* Configure Downstream action for add/replace tag-vid 100, copy tag priority
*/

classify_cfg_t  entry;
int32  ret;

rtk_classify_init();

memset(&entry, 0, sizeof(rtk_classify_cfg_t));

entry.index = 128;
entry.direction = CLASSIFY_DIRECTION_DS;
entry.valid = ENABLED;
entry.templateIdx = 0;

entry.act.dsAct.csAct = CLASSIFY_DS_CSACT_DEL_STAG;
entry.act.dsAct.cAct = CLASSIFY_DS_CACT_ADD_CTAG_8100;
entry.act.dsAct.cVidAct = CLASSIFY_DS_VID_ACT_ASSIGN;
entry.act.dsAct.cPriAct = CLASSIFY_DS_PRI_ACT_FROM_1ST_TAG;
```

```
entry.act.dsAct.cTagVid = 100;

if((ret = rtk_classify_cfgEntry_add(&entry)) != RT_ERR_OK)
    return RT_ERR_FAILED;
```

5.4. Sample code

This section gives some examples for setup a configuration of Classification rules and actions. The first step of Classification function is to call ***rtk_classify_init.*** After calling this API, the following code doesn't need to call ***rtk_classify_init*** again.

The basic operation of adding a Classification rule can be separated into 2 steps: Add fields into a configuration, and write this configuration to H/W register.

API ***rtk_classify_field_add*** is used to add fields into a configuration. At this stage, all field information of classification configuration is only kept in software database. It is possible to add multiple fields into single configuration.

Once all fields are added to the configuration, users can call API ***rtk_classify_cfgEntry_add*** to write the current configuration into H/W register. The direction, action, valid bit and template index are also set by this API.

The following sample code shows how to add a downstream rule, to filtering Stream-ID 10, tag VID 1000, priority 5, and action to forward to UNI_0, translate VID 1000 to C-VID 100, copy priority.

Example:

```
rtk_classify_cfg_t  entry;
rtk_classify_field_t *pField_vid, *pField_pri, *pField_sid;
rtk_portmask_t portMask;
int32  ret;

rtk_classify_init();

/* DS: filtering SID 10, C-tag VID 1000, priority 5, and action to forward
to UNI_0, translate VID 1000 to VID 100, copy priority*/
memset(&entry, 0, sizeof(rtk_classify_cfg_t));

entry.index = 128;
```

```
entry.direction = CLASSIFY_DIRECTION_DS;
entry.valid = ENABLED;
entry.templateIdx = 0;

pField_vid = (rtk_classify_field_t *) alloc(sizeof(rtk_classify_field_t));
pField_vid->fieldType = CLASSIFY_FIELD_TAG_VID;
pField_vid->classify_pattern.tagVid.value = 1000;
pField_vid->classify_pattern.tagVid.mask = 0xfff;
if((ret = rtk_classify_field_add(&entry, pField_vid)) != RT_ERR_OK)
    return ret;

pField_pri = (rtk_classify_field_t *) alloc(sizeof(rtk_classify_field_t));
pField_pri->fieldType = CLASSIFY_FIELD_TAG_PRI;
pField_pri->classify_pattern.tagPri.value = 5;
pField_pri->classify_pattern.tagPri.mask = 7;
if((ret = rtk_classify_field_add(&entry, pField_pri)) != RT_ERR_OK)
    return ret;

pField_sid = (rtk_classify_field_t *) alloc(sizeof(rtk_classify_field_t));
pField_sid->fieldType = CLASSIFY_FIELD_TOS_DSIDX;
pField_sid->classify_pattern.tosDsidx.value = 10;
pField_sid->classify_pattern.tosDsidx.mask = 0x7f;
if((ret = rtk_classify_field_add(&entry, pField_sid)) != RT_ERR_OK)
    return ret;

entry.act.dsAct.csAct = CLASSIFY_DS_CSACT_DEL_STAG;
entry.act.dsAct.cAct = CLASSIFY_DS_CACT_ADD_CTAG_8100;
entry.act.dsAct.cVidAct = CLASSIFY_DS_VID_ACT_ASSIGN;
entry.act.dsAct.cPriAct = CLASSIFY_DS_PRI_ACT_FROM_1ST_TAG;
entry.act.dsAct.cTagVid = 100;
entry.act.dsAct.uniAct = CLASSIFY_DS_UNI_ACT_FORCE_FORWARD;
rtk_switch_port2PortMask_set(&entry.act.dsAct.uniMask, RTK_PORT_UTP0);

if((ret = rtk_classify_cfgEntry_add(&entry)) != RT_ERR_OK)
    return RT_ERR_FAILED;

free(pField_vid);
free(pField_pri);
free(pField_sid);
```