RTL9607C
SINGLE-CHIP PON

Quality of Service Application Note

(CONFIDENTIAL: Development Partners Only)

Rev. 1.0.0
31 May 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

REVISION HISTORY

Revision	Release Date	Summary
1.0.0	2017/05/31	First Release

Table of Contents

List of Tables

List of Figures

1. Overview

Quality of Service (QoS) is an important function on most of the networking system. The idea is that it prioritizes the traffic according to its characteristic. Each class of network traffic has different characteristic such that it would be more efficient if it is forwarded with different behavior. For example, audio/video streaming require shorten latency but can tolerate several loses. On the other hand, data transfer cannot afford any loses but won't be so sensitive to the latency. So it would be more feasible to treat different classes with different services.

The most common QoS function on the switch is achieved by 802.1Q and 802.1p. Also a layer 3 DSCP would sometime be used as another way to accomplish QoS too. Switch/Router on the path utilize those QoS parameters to classified the traffic and process it with different means.

2. Internal Priority

ASIC uses a 3-bits internal priority for each forwarding frame. This internal priority decides the frame's final output queue of the port. In other words, the internal priority decides the actual frame forwarding priority. On ASIC, the internal priority sources from 6 different characteristics of frame, including: (see Figure 1)

- Dot1Q-based priority,
- Port-based priority,
- DSCP-based priority,
- ACL-based priority,
- SVLAN-based priority and

Not all of the frames have all above 6 priorities. In that case, the absent priority won't be used as the final internal priority. Each of the source aim for a specific scenario of traffic and will be discuss latter in detail.

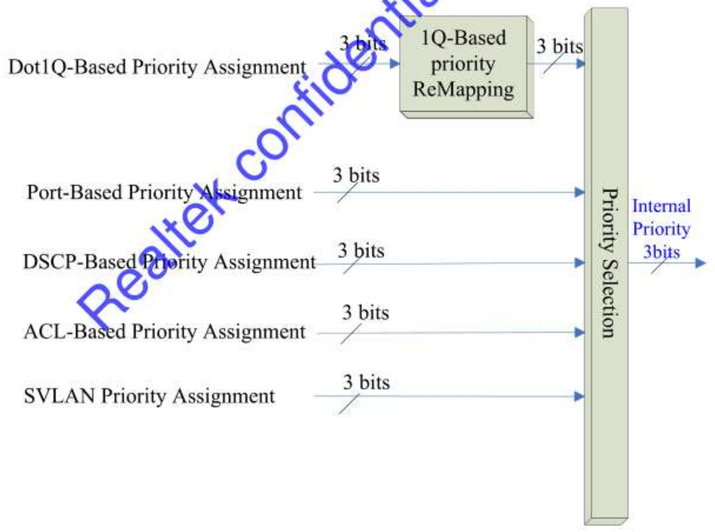

Dot1Q-Based Priority Assignment
3 bits
1Q-Based priority ReMapping
3 bits
Port-Based Priority Assignment
3 bits
DSCP-Based Priority Assignment
3 bits
ACL-Based Priority Assignment
3 bits
SVLAN Priority Assignment
3 bits
Priority Selection
Internal Priority 3bits

Figure 1. Sources of Internal Priority

2.1. *Dot1Q-Based Priority*

Dot1Q-based priority is sourcing from one of the following two priorities, **Tagged priority** and **Port-and-Protocol based VLAN priority**. If a frame is received with VLAN tag or priority tag, then the Dot1Q-based priority is the 3-bits priority field in the tag. If it is untagged frame, then the Dot1Q-based priority would try to use the port-and-protocol-based VLAN priority. If the port-and-protocol based VLAN is absent, then the Dot1Q-based priority will be absent too.

Sometimes the priority in the 802.1Q VLAN can fully represent the importance of the frame; in that case, use the priority from tag would make the QoS function more accuracy. For example, if the network is fully managed by the operator. All kinds of traffic are set with proper tag priorities. In this case, the switch could take advantage with the well classified traffic and uses its tag priority as a factor of internal priority.

Once Dot1Q-based priority is decided, it goes through a configurable whole system Dot1Q-based priority remapping table and convert the 3-bits priority to an absolute 3-bits priority. This absolute 3-bits priority will then be used as the Dot1Q-based priority (see Figure 2).

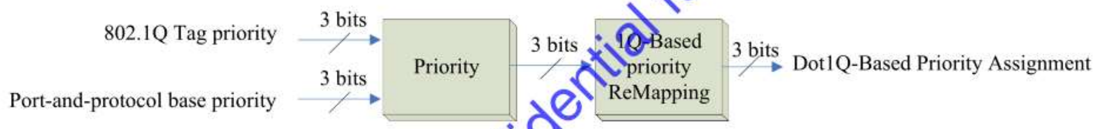

802.1Q Tag priority
3 bits
Port-and-protocol base priority
3 bits
Priority
3 bits
1Q-Based priority ReMapping
3 bits
Dot1Q-Based Priority Assignment

Figure 2. Dot1Q-based Priority and Remapping

2.2. *Port-Based Priority*

Port-based priority is used to prioritize the switch ports. It is a simple concept that if one of the switch port's traffic is much more important than others, it would be reasonable to set its traffic priority higher than others. The configuration is a per-port basis setting to give each port a port-based priority. This internal priority source won't be absent under any circumstance since the per-port configuration would has default value even without any user provision.

2.3. *DSCP-Based Priority*

DSCP-based priority is very similar to the Dot1Q-based priority that uses the priority settings from the ingress frame. Instead of using the 802.1Q tag priority in the frame, DSCP-based priority uses the differential service code point (DSCP) to assign internal priority. DSCP in IP header is a 6-bits value that represents the traffic's service class. Router that supports QoS would be able to route the traffic according to the DSCP value. Due to it have more bits than the internal priority, it is necessary to convert the DSCP before use it as the internal priority. A configurable per system table is provided for this conversion. In the cast that the frame has no DSCP value, the DSCP-based priority would be absent.

2.4. ACL-Based Priority

ACL-based priority uses the action "priority assignment" as one of the source of internal priority. Once the frame hits the ACL rule and the action contains the priority assignment with PRIACT 0x00 (ACL priority), the ACL-based priority would be assigned according to the PRIDX. The concept of ACL-based priority is that for some specific frame/traffic that is consider important, switch can use ACL rules to hit those frame/traffic and assign the higher internal priority. This kind of assignment is especially useful for those frame or traffic that are not appear on fix port or with fix VLAN tags. e.g. the RTP packet of voice over IP (VoIP) traffic. Switch can uses the ACL rules to hit RTP packets without effect other kinds of frames.

In the case that the ingress frame doesn't hit the ACL or the ACL action has no such priority assignment, the ACL-based priority would be absent.

2.5. SVLAN-Based Priority

SVLAN-based priority uses the SVLAN to derive the internal priority. It uses the priority in the service tag as the SVLAN-based priority. The SVLAN-based priority customer VLAN, it might be absent due to the absent of SVLAN and is suitable for the traffic already classified by the service provider.

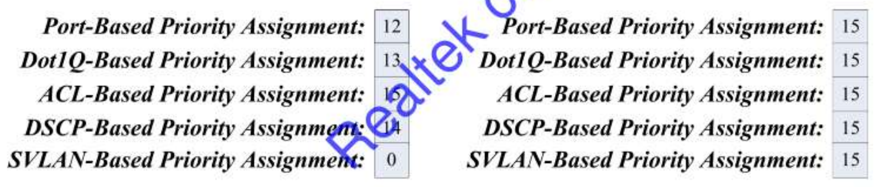
2.6. Priority Selection

Since the final internal priority could be only one of the above 5 priorities, the chip provide a weight table to allow user flexibly assign the internal priority. A 3 bits weight can be assigned to each of the priority source (see Table 1).

Field	Description	Bits
QOS_PORT_WEIGHT	Weighted value for Port-Based Priority Assignment	3
QOS_1Q_WEIGHT	Weighted value for Dot1Q-Based Priority Assignment	3
QOS_DSCP_WEIGHT	Weighted value for DSCP-Based Priority Assignment	3
QOS_ACL_WEIGHT	Weighted value for ACL-Based Priority Assignment	3
QOS_SVLAN_WEIGHT	Weighted value for SVLAN Priority Assignment	3

Table 1. Weight Table of Internal Priority Source

The use of this table is to assign each internal priority with a precedence to be use. The internal priority source with large weight would have higher precedence. If two or more internal priority sources have the same weight, then the highest priority among them will be the final internal priority. For example in the left side of Figure 3, ACL-based priority > DSCP-based priority > Dot1Q-based priority > Port-based priority > SVLAN-based priority. If highest priority source is absent, then the next highest priority would be used. The second example is in the right side of Figure 3. All 9 priority sources have the same weight. That means all the sources have the same priority, so the final internal priority would be the highest priority among these sources.



Port-Based Priority Assignment:	12	*Port-Based Priority Assignment:*	15
Dot1Q-Based Priority Assignment:	13	*Dot1Q-Based Priority Assignment:*	15
ACL-Based Priority Assignment:	15	*ACL-Based Priority Assignment:*	15
DSCP-Based Priority Assignment:	14	*DSCP-Based Priority Assignment:*	15
SVLAN-Based Priority Assignment:	0	*SVLAN-Based Priority Assignment:*	15

Figure 3. Example of Internal Priority Selection

3. Output Queue

3.1. *Queue Mapping*

The chip supports four priorities to queue tables for all ports. Each port can specify any one of the four tables. The table is configured through each priority. Table 2 shows the suggested priorities to queue table.

	Index of output queue mapping table			
Priority	**1st**	**2nd**	**3rd**	**4th**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	2
3	0	0	1	3
4	0	1	2	4
5	0	1	2	5
6	0	1	3	6
7	0	1	3	7

Table 2. Suggested Priorities to Queue Settings

In the above table, the first table uses only one queue, the second table uses two queues, the third table uses four queues and the last table uses all eight queues. User can configure each table through APIs.

3.2. *Management*

The output queues on ASIC support 2 different modes. One is weighted fair queue (WFQ) and the other one is strict priority. The queue mode is configured on per-queue basis. Mix up configuration of two different modes is also supported.

For WFQ, a weight is given for each queue. The weight is 1~1023 for queue 1~7, as to queue 0, it is 1~65535. The usage for this kind of design is that if, in some case, an extreme ration of weight is needed, this design can use queue 0 to create a large ratio between all 8 queues.

For strict priority, the queue with large ID has higher priority. That means the priority of strict priority queues are queue 7 > queue 6 > queue 5 > queue 4 > queue 3 > queue 2 > queue 1 > queue 0.

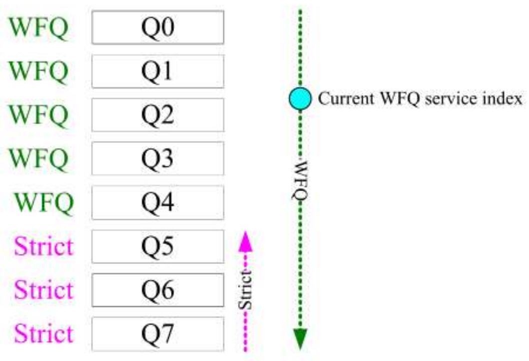


Figure 4. Queue Behavior with Both WFQ and Strict Priority

For the mixed mode, the strict priority queues have the highest priority. The WFQ queues can only be served if all the strict priority queues have nothing to send at any given transmission (see Figure 4). The strict priority queues always scan from the queue with highest queue ID. On the contrary, WFQ always scan from the queue with lowest queue ID.

4. Remarking

4.1. *802.1Q Remarking*

The chip supports a whole system priority regeneration table to support 802.1Q remarking feature. Once the frame decided its 3-bits user priority, it will then be regenerated by the remarking table. The remarking function has a per-port basis configuration to enable or disable it. The priority after remarking would be the final egress priority in the 802.1Q tag. Note that the SVLAN's priority won't be affected by 802.1Q remarking.

4.2. *DSCP Remarking*

DSCP remarking is also supported by using several per-system tables and configurations. Each port can enable or disable the remarking independently. The DSCP remarking supports two sources to convert as DSCP values. They are internal priority and original DSCP value. Each port can configure to use one of the two conversion sources. As to the remarking table, internal priority use a table to convert the 3-bit priority to the 6-bit DSCP while original DSCP using a 6-bit to 6-bit table to achieve that.
If multiple DSCP remarking functions act concurrently, the precedence is

- **L34 FB DSCP remarking > ACL DSCP remarking > Port-Based DSCP remarking.**

5. API

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

5.1. *Initialization*

rtk_qos_init is the first API user should invoke before setup any configuration.

5.2. *Internal Priority*

5.2.1. Dot1Q Remapping

rtk_qos_1pPriRemapGroup_set set the remapping table of Dot1Q-based priority to remap the Dot1Q-based priority to one of the sources of internal priority. The grpIdx is for future usage, it is should be 0 for now. The drop precedence dp is not supported currently. For feature detail description, please reference to section 2.1

```
/* Example of configuration for Dot1Q-based priority remapping table
 * Dot1Q-based Priority    Internal Priority
 *           0                     7
 *           1                     6
 *           2                     5
 *           3                     4
 *           4                     3
 *           5                     2
 *           6                     1
 *           7                     0
 */

int ret;
int i;
```

```
for(i = 0; i <= 7; i++)
{
    ret = rtk_qos_1pPriRemapGroup_set(0,    /* grpIdx */
                                      i,    /* dot1pPri */
                                      7-i, /* intPri */
                                      0     /* dp */);

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

5.2.2. Port-Based Priority Assignment

rtk_qos_portPri_set set the port-based priority for each port. It is used as one of the internal priority source. For feature detail description, please reference to section 2.2

```

/* Example of configuration for Port-based priority
 * for port RTK_PORT_UTP0, RTK_PORT_UTP1 and RTK_PORT_PON
 */

int ret;
rtk_port_t cfgPort;

rtk_switch_phyPortId_get(RTK_PORT_UTP0, &cfgPort);
if((ret = rtk_qos_portPri_set(cfgPort, 7)) != RT_ERR_OK)
{
    return ret;
}

rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
if((ret = rtk_qos_portPri_set(cfgPort, 5)) != RT_ERR_OK)
{
    return ret;
}
```

```
rtk_switch_phyPortId_get(RTK_PORT_PON, &cfgPort);
if((ret = rtk_qos_portPri_set(cfgPort, 3)) != RT_ERR_OK)
{
    return ret;
}
```

5.2.3. DSCP Remapping

rtk_qos_dscpPriRemapGroup_set set the DSCP remapping table to remap the DSCP value to one of the internal priority source. The grpIdx is for future usage, it is should be 0 for now. The drop precedence dp is not supported currently. For feature detail description, please reference to section 2.3

```
/* Example of configuration for DSCP remapping table
 * DSCP    Internal Priority
 *  0             0
 * ...            0
 *  7             0
 *  8             1
 * ...            1
 *  15            1
 *  16            2
 * ...            2
 *  23            2
 *  24            3
 * ...            3
 *  31            3
 *  32            4
 * ...            4
 *  39            4
 * ...           ...
 */

int ret;
int i;

for(i = 0; i < 64 ; i++)
```

```
{
    ret = rtk_qos_dscpPriRemapGroup_set (
              0, /* grpIdx */
              i, /* dscp */
              (rtk_pri_t)(i/8), /* intPri */
              0  /* dp */ );
    if(ret != RT_ERR_OK){
        return ret;
    }
}
```

5.2.4. Priority Selection Weight Configuration

rtk_qos_priSelGroup_set uses the rtk_qos_priSelWeight_t struct to configure all of the priority selection weight at once. The grpIdx is for future usage, it is should be 0 for now. For feature detail description, please reference to section 2.6

```
/* Example of priority selection weight configuration
 *   Port-based priority: 12
 *  Dot1Q-based priority: 13
 *   DSCP-based priority: 15
 *    ACL-based priority: 14
 *  SVLAN-based priority: 0
 */

int ret;
rtk_qos_priSelWeight_t selWeight;

selWeight.weight_of_portBased = 12;
selWeight.weight_of_dot1q = 13;
selWeight.weight_of_dscp = 15;
selWeight.weight_of_acl = 14;
selWeight.weight_of_svlanBased = 0;

if((ret = rtk_qos_priSelGroup_set(0, &selWeight)) != RT_ERR_OK)
{
    return ret;
}
```

5.2.5. CPU Port Priority Remapping Table Configuration

rtk_qos_fwd2CpuPriRemap_set set the remapping table of CPU port priority to remap the internal priority to the final priority forward to CPU port. For feature detail description, please reference to section 1

5.3. Output Queue

5.3.1. Priority to Queue Table Configuration

rtk_qos_priMap_set uses the rtk_qos_pri2queue_t structure to configure one priority to queue table at once. There are up to 4 tables supported in ASIC. For feature detail description, please reference to section 3.1

```
/* Example of priority to queue table configuration
 *        Priority
 * Group    0   1   2   3   4   5   6   7
 *        ================================
 *   0      0   1   2   3   4   5   6   7
 *   1      0   0   0   0   0   0   0   0
 */
int ret;
int i;
rtk_qos_pri2queue_t pri2QTable;

/* Configure the second table */
memset(&pri2Qtable, 0, sizeof(rtk_qos_pri2queue_t));
if((ret = rtk_qos_priMap_set(1, &pri2QTable)) != RT_ERR_OK)
{
    return ret;
}

/* Configure the first table */
for(i = 0; i < RTK_MAX_NUM_OF_PRIORITY; i++)
```

```
{
    pri2Qtable.pri2queue[i] = i;
}
if((ret = rtk_qos_priMap_set(0, &pri2QTable)) != RT_ERR_OK)
{
    return ret;
}
```

5.3.2. Port Configuration of Priority to Queue Table

rtk_qos_portPriMap_set configures which one of the four priority to queue tables to be used. It would be proper to configure the priority to queue table first before the port uses it. For feature detail description, please reference to section 3.1

```
/* Example of per-port priority to queue table usage
*/

int ret;
rtk_port_t cfgPort;
uint32 grpIdx;

/* Configure the port 0 uses first table */
grpIdx = 0;
rtk_switch_phyPortId_get(RTK_PORT_UTP0, &cfgPort);
if((ret = rtk_qos_portPriMap_set(cfgPort, grpIdx)) != RT_ERR_OK)
{
    return ret;
}

/* Configure the port 1 uses first table */
rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
if((ret = rtk_qos_portPriMap_set(cfgPort, grpIdx)) != RT_ERR_OK)
{
    return ret;
}

/* Configure the PON port uses second table */
```

```
grpIdx = 1;
rtk_switch_phyPortId_get(RTK_PORT_PON, &cfgPort);
if((ret = rtk_qos_portPriMap_set(cfgPort, grpIdx)) != RT_ERR_OK)
{
    return ret;
}

/* Configure the CPU port uses second table */
grpIdx = 1;
rtk_switch_phyPortId_get(RTK_PORT_CPU, &cfgPort);
if((ret = rtk_qos_portPriMap_set(cfgPort, grpIdx)) != RT_ERR_OK)
{
    return ret;
}
```

5.3.3. Port Configuration of Output Queue

rtk_qos_schedulingQueue_set uses the rtk_qos_queue_weights_t structure to configure all queue type and weight of a single port at once. Each port's queue configuration is independent from each other. For feature detail description, please reference to section 3.2

```
/* Example of port queue management
 * Queie ID  Type    Weight
 *        0  WFQ     30000
 *        1  WFQ     64
 *        2  WFQ     64
 *        3  WFQ     64
 *        4  Strict  -
 *        5  Strict  -
 *        6  Strict  -
 *        7  Strict  -
 */
int ret;
int i;
rtk_port_t cfgPort;
rtk_qos_queue_weights_t weights;
```

```
/* Reset the structure */
memset(&weights, 0, sizeof(rtk_qos_queue_weights_t));
weights.weights[0] = 30000;
weights.weights[1] = 64;
weights.weights[2] = 64;
weights.weights[3] = 64;
weights.weights[4] = 0;
weights.weights[5] = 0;
weights.weights[6] = 0;
weights.weights[7] = 0;

/* Apply the settings to port 1 */
rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
if((ret = rtk_qos_schedulingQueue_set (cfgPort, &weights)) != RT_ERR_OK)
{
    return ret;
}
```

5.4. Remarking

5.4.1. Port Configuration of 802.1Q Priority Remarking

rtk_qos_1pRemarkEnable_set configures the 802.1Q remarking function of each port. This configuration enables or disables the per-port priority remarking function. If it is enabled, the priority would be remarked using the per-system remarking table. For feature detail description, please reference to section 4.1

```

/* Example of per-port 802.1Q priority remarking function
 */

int ret;
rtk_port_t cfgPort;

/* Configure port 0 to enable 802.1Q priority remarking */
rtk_switch_phyPortId_get(RTK_PORT_UTP0, &cfgPort);
```

```
if((ret = rtk_qos_1pRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}

/* Configure port 1 to enable 802.1Q priority remarking */
rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
if((ret = rtk_qos_1pRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}

/* Configure PON port to enable 802.1Q priority remarking */
rtk_switch_phyPortId_get(RTK_PORT_PON, &cfgPort);
if((ret = rtk_qos_1pRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}
```

5.4.2. 802.1Q Priority Remarking Table Configuration

rtk_qos_1pRemarkGroup_set configures the 802.1Q priority remarking table to remark the user priority to the final egress tag priority. The grpIdx is for future usage, it is should be 0 for now. For feature detail description, please reference to section 4.1

```
/* Example of 802.1Q priority remarking table configuration
 * User Priority   Tag Priority
 *         0              0
 *         1              1
 *         2              2
 *         3              3
 *         4              4
 *         5              7
 *         6              5
 *         7              6
 */
```

```
int ret;
int i;

for(i = 0; i < 5; i++)
{
    ret = rtk_qos_1pRemarkGroup_set(0,  /* grpIdx */
                                    i,  /* intPri */
                                    0,  /* dp */
                                    i); /* dot1pPri */
    if(ret != RT_ERR_OK)
    {
        return ret;
    }
}

/* Set user priority 5 remark as tag priority 7 */
if((ret = rtk_qos_1pRemarkGroup_set(0, 5, 0, 7)) != RT_ERR_OK)
{
    return ret;
}

/* Set user priority 6 remark as tag priority 5 */
if((ret = rtk_qos_1pRemarkGroup_set(0, 6, 0, 5)) != RT_ERR_OK)
{
    return ret;
}

/* Set user priority 7 remark as tag priority 6 */
if((ret = rtk_qos_1pRemarkGroup_set(0, 7, 0, 6)) != RT_ERR_OK)
{
    return ret;
}
```

5.4.3. Port Configuration of DSCP Remarking

rtk_qos_dscpRemarkEnable_set configures the DSCP remarking function of each port. This configuration enables or disables the per-port DSCP remarking function. If it is enabled, the DSCP would be remarked using the per-system remarking table. For feature detail description, please reference to section 4.2

```
/* Example of per-port DSCP remarking function
 */

int ret;
rtk_port_t cfgPort;

/* Configure port 0 to enable DSCP remarking */
rtk_switch_phyPortId_get(RTK_PORT_UTP0, &cfgPort);
if((ret = rtk_qos_dscpRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}

/* Configure port 1 to enable DSCP remarking */
rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
if((ret = rtk_qos_dscpRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}

/* Configure PON port to enable DSCP remarking */
rtk_switch_phyPortId_get(RTK_PORT_PON, &cfgPort);
if((ret = rtk_qos_dscpRemarkEnable_set(cfgPort, ENABLED)) != RT_ERR_OK)
{
    return ret;
}
```

5.4.4. DSCP Remarking Table Configuration

5.4.4.1 Internal Priority / User Priority to DSCP

rtk_qos_dscpRemarkGroup_set configures the DSCP remarking table to remark the internal/user priority to the final egress DSCP. The grpIdx is for future usage, it is should be 0 for now. For feature detail description, please reference to section 4.2

```
/* Example of DSCP remarking table configuration
 *      Priority   DSCP
 *         0          0
 *         1          8
 *         2         16
 *         3         24
 *         4         32
 *         5         40
 *         6         48
 *         7         56
 */

int ret;
int i;

for(i = 0; i < 8; i++)
{
    ret = rtk_qos_dscpRemarkGroup_set(0,       /* grpIdx */
                                      i,       /* intPri */
                                      0,       /* dp */
                                      i * 8); /* dscp */

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

5.4.4.2 Ingress DSCP to Egress DSCP

rtk_qos_dscp2DscpRemarkGroup_set configures the DSCP remarking table to remark the ingress DSCP to the final egress DSCP. The grpIdx is for future usage, it is should be 0 for now. For feature detail description, please reference to section 4.2

```
/* Example of DSCP remarking table configuration
 * Ingress DSCP   Egress DSCP
 *      0              0
 *      1              1
 *     ...            ...
 *     56             56
 *     57             63
 *     58             62
 *     59             61
 *     60             60
 *     61             59
 *     62             58
 *     63             57
 */


int ret;
int i;


for(i = 0; i <= 56; i++)
{
    ret = rtk_qos_dscp2DscpRemarkGroup_set(0,   /* grpIdx */
                                           i,   /* dscp */
                                           i); /* rmkDscp */
    if(ret != RT_ERR_OK)
    {
        return ret;
    }
}



for(i = 57; i <= 63; i++)
{
    ret = rtk_qos_dscp2DscpRemarkGroup_set(0,                /* grpIdx */
                                           i,                /* dscp */
                                           57 + (63 - i)); /* rmkDscp */
    if(ret != RT_ERR_OK)
    {
        return ret;
    }
}
```

5.4.5. DSCP Remarking Source Configuration

rtk_qos_portDscpRemarkSrcSel_set configures the DSCP remarking function of each port. This configuration selects the DSCP remarking source of per-port DSCP remarking function. A structure rtk_qos_dscpRmkSrc_t is used to define all possible sources. For feature detail description, please reference to section 4.2

```
/* Example of per-port DSCP remarking source selection function
 */

int ret;
rtk_port_t cfgPort;
rtk_qos_dscpRmkSrc_t type;

/* Configure port 0 to use internal priority as DSCP remarking source */
rtk_switch_phyPortId_get(RTK_PORT_UTP0, &cfgPort);
type = DSCP_RMK_SRC_INT_PRI;
if((ret = rtk_qos_portDscpRemarkSrcSel_set(cfgPort, type)) != RT_ERR_OK)
{
    return ret;
}

/* Configure port 1 to use DSCP as DSCP remarking source */
rtk_switch_phyPortId_get(RTK_PORT_UTP1, &cfgPort);
type = DSCP_RMK_SRC_DSCP;
if((ret = rtk_qos_portDscpRemarkSrcSel_set (cfgPort, type)) != RT_ERR_OK)
{
    return ret;
}

/* Configure PON port to use DSCP as DSCP remarking source */
rtk_switch_phyPortId_get(RTK_PORT_PON, &cfgPort);
type = DSCP_RMK_SRC_USER_PRI;
if((ret = rtk_qos_portDscpRemarkSrcSel_set (cfgPort, type)) != 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