RTL9607C
SINGLE-CHIP PON

PON MAC
Application Note
(CONFIDENTIAL: Development Partners Only)

Rev. 1.0.0
26 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

COPYRIGHT

TRADEMARKS

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	2017/05/26	First Release

Table of Contents

List of Tables

List of Figures

1. Overview

The PON MAC module provided queue scheduling management for PON port upstream packet. The scheduling management includes:

- Queue and scheduler ID mapping
- Flow id to queue mapping
- Per queue configuration :
  - Scheduling type assign (Strict priority, WFQ or WRR)
  - Queue weight
  - Per queue CIR/PIR setting
  - Egress drop enable state

PON MAC provides 128 generic queues which support configuration for schedule type, queue weight, and CIR/PIR rate of each queue. PON MAC provides 32 independent scheduling units. Each queue can mapping to a specific scheduler ID in the allowed range. For GPON the scheduler ID is T-CONT and for EPON the scheduler ID is LLID. Under EPON application, only 8 LLID is supported. If the queue do not mapping to any schedule ID this queue will not be scheduled.

Additional packet buffer named “packet buffer offload (PBO)” for PON port upstream/downstream traffic is supported. It uses the system DRAM as packet buffer to store the upstream/downstream packets. PBO provides up to 32Mb for upstream and downstream PON traffic separately.

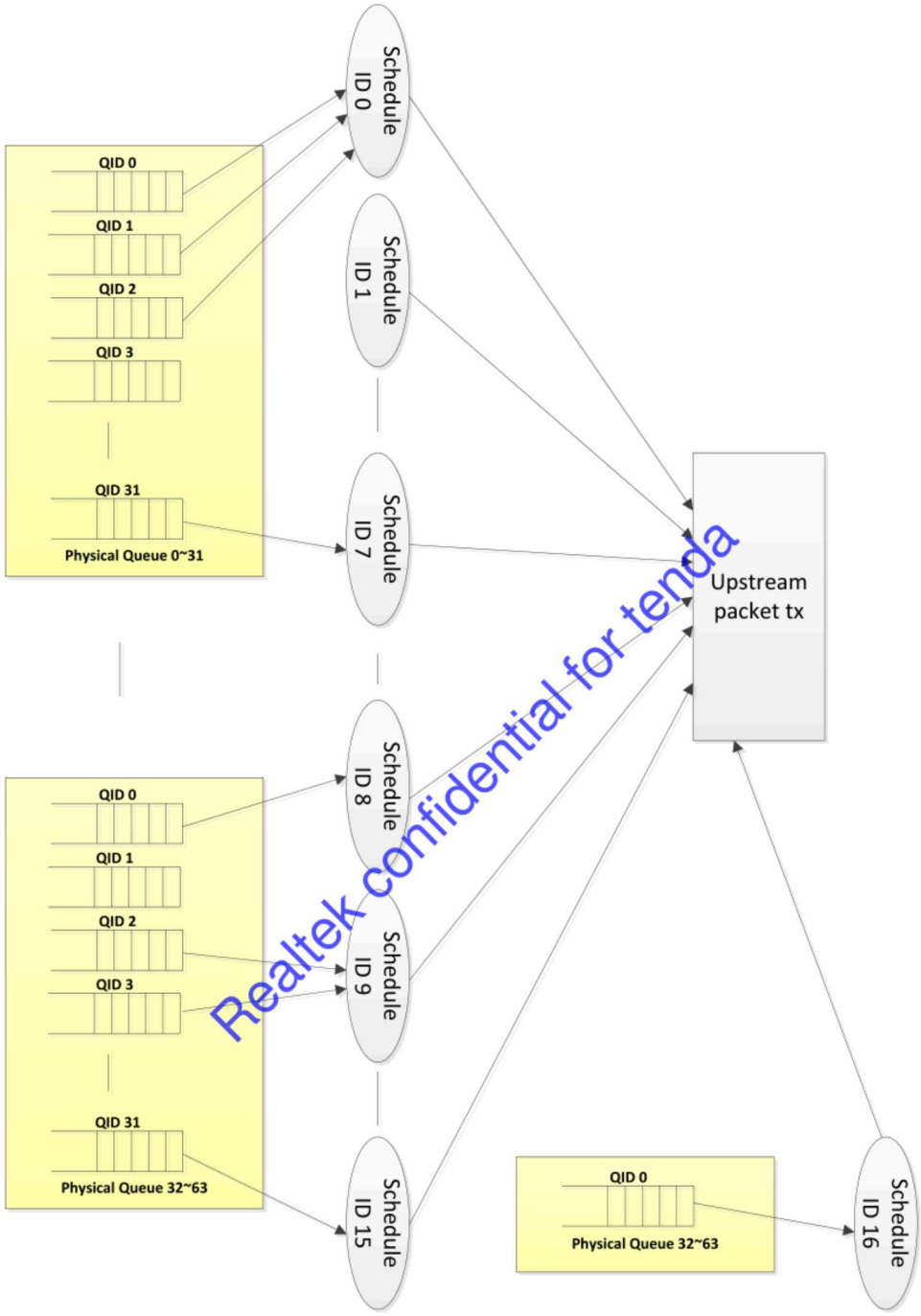

Figure 1. Queue to schedule ID mapping

2. Queue to schedule unit mapping

PON MAC provides 128 generic queues for upstream packet. They are divided into 4 scheduling groups. Each scheduling group shares 32 queues. The physical Queue id 0~ 31 can only mapping to scheduler ID 0~7, physical Queue ID 32~ 63 can only mapping to scheduler id 8~15, physical Queue ID 64~ 95 can only mapping to scheduler id 16~23, physical Queue ID 96~ 127 can only mapping to scheduler id 24~31.

Table 1. Schedule ID and physical queue mapping

Scheduler ID	Physical queue range
0~7	0 ~ 31
8~15	32 ~ 63
16~23	64 ~ 95
24~31	96~127

For RTK API system used local queue id to identify a queue id. The logical queue id is {schedule ID, queue id} pair. If the queue do not mapping to any schedule ID this queue will not be scheduled. Mapping a queue to multiple schedulers will cause the schedulers' behavior unpredictable. The API use queue adds to mapping the physical queue to the logical queue pair. Before call any "rtk_ponmac_queue_add" API the mapping is empty.

Table 2. Logical queue id and physical queue mapping before add any logical queue

Logical queue pair {schedule ID, queue id}	Physical queue id
Not mapped	0
Not mapped	1
Not mapped	2
Not mapped	3
Not mapped	4
Not mapped	5
Not mapped	6
Not mapped	7
Not mapped	8
Not mapped	9
…	…
Not mapped	120
Not mapped	121
Not mapped	122
Not mapped	123
Not mapped	124
Not mapped	125
Not mapped	126
Not mapped	127

After call "rtk_ponmac_queue_add" for logical queue pair, the mapping will be established.

Add

{Schedule id 0, queue 0}

{Schedule id 7, queue 6}

{Schedule id 3, queue 7}

{Schedule id 1, queue 9}

{Schedule id 15, queue 24}

{Schedule id 9, queue 26}

{Schedule id 8, queue 30}

{Schedule id 31, queue 31}

Table 3. Logical queue id and physical queue mapping example

Logical queue pair {schedule ID, queue id}	**Physical queue id**
{0 , 0}	0
Not mapped	1
Not mapped	2
Not mapped	3
Not mapped	4
Not mapped	5
{7, 6}	6
{3, 7}	7
Not mapped	8
{1, 9}	9
…	…
{15, 24}	56
Not mapped	57
{9, 26}	58
Not mapped	59
Not mapped	60
Not mapped	61
{8, 30}	62
…	…
{31, 31}	127

3. Flow id to queue mapping

For upstream packet the classification module will assign the flow id, according to flow id the PON MAC module provide a mapping table to assign the flow id to the schedule queue. System total provide 128 queues for upstream packet, and it divided into 4 schedule groups, each scheduler group share 32 queues. Flow ID 127 is used for GPON OMCI usage.

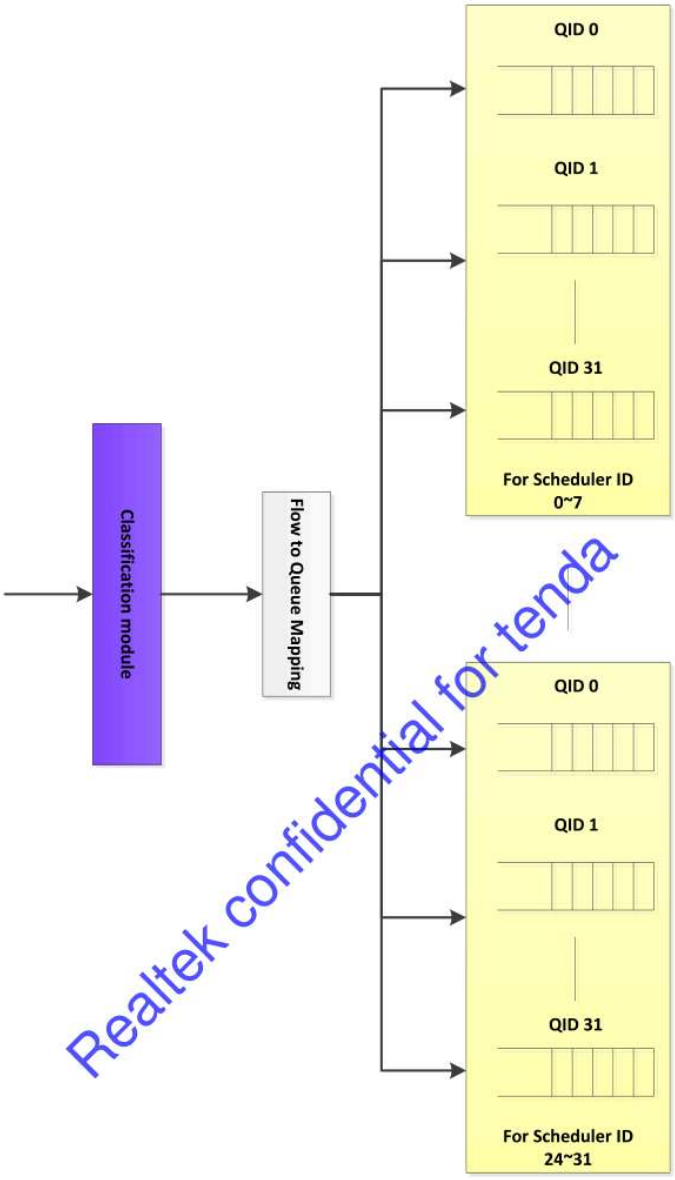

Figure 2. Flow to Queue Block Diagram

4. Flow Valid

To prevent packet being put into queue before scheduler can schedule, a flow valid configure is provided for each flow. Software should turn on the valid bit when the PON ready. Packet being sent to invalid flow will be discarded.

5. Per Queue configurations

5.1. Queue Type setting

For PON Mac queues it provides Strict, WFQ and WRR setting. The scheduler will service strict type queue first according to queue number (higher queue id will be service first). The WFQ/WRR will be service according to the WFQ/WRR weight setting. System also provide WFQ/WRR and Strict mixed hybrid mode in the schedule unit. All combinations of WFQ/WRR and SP are allowed.

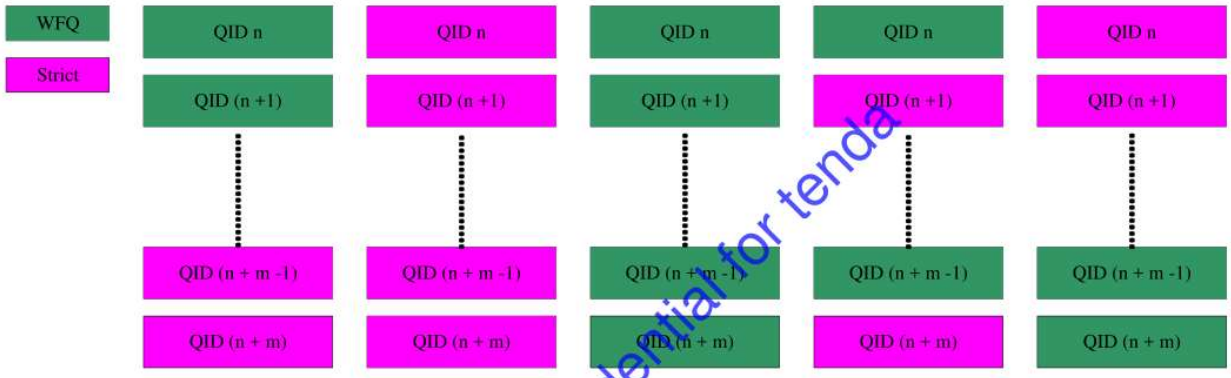

Figure 3. Valid queue type assign for hybrid mode

5.2. CIR/PIR rate setting

For PON MAC the schedulers provide CIR (Committed Information Rate) and PIR (Peak Information Rate) setting.

CIR schedule

The CIR rate is committed rate the scheduler. The scheduler will service queues which egress rate is under CIR first. If there are more than one queue egress rate under its CIR rate, the scheduler will schedule these queue according to queue type setting. When the queue egress rate reached the CIR rate, this queue will stop schedule until all queue in this schedule unit have reached their CIR rate. If the egress rates of all queues have reached their CIR rate in the schedule unit, the scheduler will start to schedule all queue according to queue type.

PIR schedule (up-bound rate limitation)

The PIR rate setting is use to limit the max egress rate for a queue. Once the egress rate reached the PIR rate, the scheduler will stop service this queue.

5.3. Egress Drop setting

Each queue also provides egress drop setting. The flow control function will be disabled when egress drop function is enabled for this queue.

6. Packet Buffer Offload

Packet buffer offload (PBO) provides additional packet buffer for PON application which reduces the usage of switch packet buffer. Upstream and downstream PON traffic use independent upstream/downstream PBO to offload packet buffer separately. Upstream/Downstream PBO use 8 x 1024 pages to buffer the packet, each page can be configured as 128/256/512 bytes. The total size of additional packet buffer would be 8Mb/16Mb/32Mb for different page size in upstream/downstream usage.

7. API

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

7.1. Initialization

rtk_ponmac_init is the first API users should call before setup any configuration.

7.2. Add queue to schedule unit

The ***rtk_ponmac_queue_add*** API will mapping the queue to the scheduler id and set the queue setting.

Example:

```
/* Add a queue 8 to schedule id 3
   The CIR is 0x10 unit 8Kbps
   The PIR is 0x8000 unit 8Kbps
   Queue type is WFQ, weight is 15
   Egress drop disable
*/

rtk_ponmac_queueCfg_t   queueCfg;
rtk_ponmac_queue_t logicalQueue;
Int32 ret ;

memset(&queueCfg, 0, sizeof(rtk_ponmac_queueCfg_t));

logicalQueue.schedulerId = 3 ;
logicalQueue.queueId     = 8 ;
```

```
queueCfg.cir        = 0x10;
queueCfg.pir        = 0x8000;
queueCfg.type       = WFQ_WRR_PRIORITY;
queueCfg.weight     = 15;
queueCfg.egrssDrop = DISABLED;

if((ret= rtk_ponmac_queue_add(&logicalQueue, &queueCfg)) != RT_ERR_OK)
{
    return ret;
}
```

7.3. Delete queue from schedule unit

The ***rtk_ponmac_queue_del*** API will remove the queue from the schedule id.

Example:

```
/* Delete a queue 8 from schedule id 3 */

rtk_ponmac_queue_t logicalQueue;
Int32 ret ;

logicalQueue.schedulerId = 3 ;
logicalQueue.queueId     = 8 ;

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


7.4. Mapping flow id to a logical queue

The ***rtk_ponmac_flow2Queue_set*** API will map the flow id to the given queue id.

Example:

```
/* mapping flow id 16 to queue 10 in schedule id 3 */

rtk_ponmac_queue_t logicalQueue;
Int32 ret ;

logicalQueue.schedulerId = 3 ;
logicalQueue.queueId     = 10 ;

if((ret= rtk_ponmac_flow2Queue_set (16,&logicalQueue)) != RT_ERR_OK)
{
    return ret;
}
```

7.5. Enable/Disable Flow

```
/* Enable flow 4 disable flow 8 */

if((ret= rtk_ponmac_sidValid_set(4, 1)) != RT_ERR_OK)
{
    return ret;
}

if((ret= rtk_ponmac_sidValid_set(8, 0)) != RT_ERR_OK)
{
    return ret;
}
```


7.6. Sample code

This section gives an example for setup a configuration of PON MAC. The example setup PON MAC as flow list:

Schedule id	Queue id	Queue setting	Flow id
0	0	CIR = 0x10 PIR = 0x8000 Queue type = STRICT_PRIORITY Egress drop disable	1
0	1	CIR = 0x10	2

		PIR = 0x8000 Queue type = STRICT_PRIORITY Egress drop disable	
8	0	CIR = 0x0 PIR = 0x1FFFF Queue type = STRICT_PRIORITY Egress drop disable	30
8	1	CIR = 0x0 PIR = 0x1FFFF Queue type = STRICT_PRIORITY Egress drop disable	31

Example:

```
rtk_ponmac_queueCfg_t   queueCfg;
rtk_ponmac_queue_t logicalQueue;
Int32 ret ;
/*for scheduler 0 queue 0*/
memset(&queueCfg, 0, sizeof(rtk_ponmac_queueCfg_t));

logicalQueue.schedulerId = 0 ;
logicalQueue.queueId     = 0 ;

queueCfg.cir       = 0x10;
queueCfg.pir       = 0x8000;
queueCfg.type      = STRICT_PRIORITY;
queueCfg.egrssDrop = DISABLED;
/*mapping the queue to scheduler id*/
if((ret= rtk_ponmac_queue_add(&logicalQueue, &queueCfg)) != RT_ERR_OK)
{
    return ret;
}
/*mapping flow id to queue id*/
if((ret= rtk_ponmac_flow2Queue_set (1,&logicalQueue)) != RT_ERR_OK)
{
    return ret;
}

/*for scheduler 0 queue 1*/
memset(&queueCfg, 0, sizeof(rtk_ponmac_queueCfg_t));

logicalQueue.schedulerId = 0 ;
```

```
logicalQueue.queueId     = 1 ;

queueCfg.cir       = 0x10;
queueCfg.pir       = 0x8000;
queueCfg.type      = STRICT_PRIORITY;
queueCfg.egrssDrop = DISABLED;
/*mapping the queue to scheduler id*/
if((ret= rtk_ponmac_queue_add(&logicalQueue, &queueCfg)) != RT_ERR_OK)
{
    return ret;
}
/*mapping flow id to queue id*/
if((ret= rtk_ponmac_flow2Queue_set (2,&logicalQueue)) != RT_ERR_OK)
{
    return ret;
}

/*for scheduler 8 queue 0*/
memset(&queueCfg, 0, sizeof(rtk_ponmac_queueCfg_t));

logicalQueue.schedulerId = 8 ;
logicalQueue.queueId     = 0 ;

queueCfg.cir       = 0x0;
queueCfg.pir       = 0x1FFFF;
queueCfg.type      = STRICT_PRIORITY;
queueCfg.egrssDrop = DISABLED;
/*mapping the queue to scheduler id*/
if((ret= rtk_ponmac_queue_add(&logicalQueue, &queueCfg)) != RT_ERR_OK)
{
    return ret;
}
/*mapping flow id to queue id*/
if((ret= rtk_ponmac_flow2Queue_set (30,&logicalQueue)) != RT_ERR_OK)
{
    return ret;
}

/*for scheduler 0 queue 0*/
memset(&queueCfg, 0, sizeof(rtk_ponmac_queueCfg_t));
```

```
logicalQueue.schedulerId = 8 ;
logicalQueue.queueId     = 1 ;

queueCfg.cir       = 0x0;
queueCfg.pir       = 0x1FFFF;
queueCfg.type      = STRICT_PRIORITY;
queueCfg.egrssDrop = DISABLED;
/*mapping the queue to scheduler id*/
if((ret= rtk_ponmac_queue_add(&logicalQueue, &queueCfg)) != RT_ERR_OK)
{
    return ret;
}
/*mapping flow id to queue id*/
if((ret= rtk_ponmac_flow2Queue_set (31,&logicalQueue)) != 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