RTL8309M-RTL8306MB-RTL8304MB PROGRAMMING GUIDE

V1.0.5
MAY 23, 2016

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

Module rtk_api_ext.h/*

Filename: rtk_api_ext.h

Description

---

rtk_switch_init

rtk_api_ret_t rtk_switch_init(*void***)**

Set chip to default configuration enviroment

Defined in: rtk_api_ext.h

Parameters *void*

Comments The API can set chip registers to default configuration for different release chip model.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

---

rtk_switch_maxPktLen_set

rtk_api_ret_t rtk_switch_maxPktLen_set(rtk_switch_maxPktLen_t *type***, rtk_switch_len_t** *len***)**

Set the max packet length

Defined in: rtk_api_ext.h

Parameters

type
max packet length type

len
max packet length

Comments The API can set max packet length. The len would be values as follows:
MAX_PKTLEN_1522B

MAX_PKTLEN_1526B

MAX_PKTLEN_2048B

MAX_PKTLEN_16000B

MAX_PKTLEN_USER

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	Invalid input parameter

rtk_switch_maxPktLen_get

rtk_api_ret_t rtk_switch_maxPktLen_get(rtk_switch_maxPktLen_t **pType*, rtk_switch_len_t **pLen*)

Get the max packet length

Defined in: rtk_api_ext.h

Parameters

**pType*
max packet length type

**pLen*
the pointer of max packet length type

Comments

The API can get max packet length.The len would be values as follows:
MAX_PKTLEN_1522B

MAX_PKTLEN_1526B

MAX_PKTLEN_2048B

MAX_PKTLEN_16000B

MAX_PKTLEN_USER

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_port_phyReg_set

rtk_api_ret_t rtk_port_phyReg_set(rtk_port_t *phy***, rtk_port_phy_reg_t** *reg***, rtk_port_phy_data_t** *regData***)**

Set PHY register data of the specific port

Defined in: rtk_api_ext.h

Parameters

phy
phy ID(0 ~ 7)

reg
Register id

regData
Register data

Comments

This API can be called to write a phy register provided by IEEE standard. RTL8309N switch has 8 PHYs(PHY 0-7).

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id

rtk_port_phyReg_get

rtk_api_ret_t rtk_port_phyReg_get(rtk_port_t *phy***, rtk_port_phy_reg_t** *reg***, rtk_port_phy_data_t** ***pData***)**

Get PHY register data of the specific port

Defined in: rtk_api_ext.h

Parameters

phy
phy number, 0 ~ 7

reg
Register id

**pData*
the pointer of Register data

Comments

This API can be called to read a PHY register data provided by IEEE standard. RTL8309N switch has 8 PHYs(PHY 0-7).

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_port_phyAutoNegoAbility_set

rtk_api_ret_t rtk_port_phyAutoNegoAbility_set(rtk_port_t *port*, **rtk_port_phy_ability_t** **pAbility*)

Set ethernet PHY auto-negotiation desired ability

Defined in: rtk_api_ext.h

Parameters

port
phy id,0~7

**pAbility*
pointer point to Ability structure

Comments

(1) RTL8309N switch only has 8 phy, so the input phy id should be 0 ~ 7.
(2) In auto-negotiation mode, phy autoNegotiation ability must be enabled

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_INPUT	invalid input parameter

rtk_port_phyAutoNegoAbility_get

rtk_api_ret_t rtk_port_phyAutoNegoAbility_get(rtk_port_t *port*, **rtk_port_phy_ability_t** **pAbility*)

Get ethernet PHY auto-negotiation ability configurations

Defined in: rtk_api_ext.h

Parameters

port
phy id,0~7

pAbility
pointer point to Ability structure

Comments (1) RTL8309N switch only has 8 phy, so the input phy id should be 0~7.
(2) In auto-negotiation mode, phy autoNegotiation ability must be enabled.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_PHY_AUTO_NEGO_MODE	invalid PHY auto

rtk_port_phyForceModeAbility_set

rtk_api_ret_t rtk_port_phyForceModeAbility_set(rtk_port_t *port*, **rtk_port_phy_ability_t** **pAbility*)

Set ethernet PHY force mode desired ability

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pAbility*
pointer point to Ability structure

Comments (1) RTL8309N switch only has 8 phy, so the input phy id should be 0~7.
(2) In force mode, phy autoNegotiation ability must be disabled.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_INPUT	invalid input parameter

rtk_port_phyForceModeAbility_get

rtk_api_ret_t rtk_port_phyForceModeAbility_get(rtk_port_t *port*, **rtk_port_phy_ability_t** **pAbility*)

Get ethernet PHY force mode ability configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pAbility*
pointer point to Ability structure

Comments

(1) RTL8309N switch only has 8 phy, so the input phy id should be 0~7.
(2) In force mode, phy autoNegotiation ability must be disabled.

Return Codes

RT_ERR_OK	ok
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_PHY_AUTO_NEGO_MODE	invalid PHY auto

rtk_port_isolation_set

rtk_api_ret_t rtk_port_isolation_set(rtk_port_t *port*, **rtk_portmask_t** *portmask*)

Set permitted port isolation portmask

Defined in: rtk_api_ext.h

Parameters

port
port id

portmask
Permit port mask

Comments

This API can be called to set port isolation mask for port 0~8.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_PORT_ID	Invalid port number
RT_ERR_PORT_MASK	Invalid portmask

rtk_port_isolation_get

rtk_api_ret_t rtk_port_isolation_get(rtk_port_t *port***, rtk_portmask_t** **pPortmask***)**

Get permitted port isolation portmask

Defined in: rtk_api_ext.h

Parameters

port
port id

**pPortmask*
the pointer of permit port mask

Comments

This API can be called to get port isolation mask.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port number
RT_ERR_NULL_POINTER	Input parameter is a null pointer

rtk_rate_igrBandwidthCtrlRate_set

rtk_api_ret_t rtk_rate_igrBandwidthCtrlRate_set(rtk_port_t *port***, rtk_rate_t** *rate***, rtk_enable_t** *ifg_include***)**

Set port ingress bandwidth control.

Defined in: rtk_api_ext.h

Parameters

port
Port id

rate
Rate of share meter

ifg_include
the pointer of Register data

Comments

For RTL8309N, port0 and port 8's max speed could be 100Mbps, and max speed could only be 100Mbps for port1 to port 7. The rate unit is 64Kbps and the range is from 64Kbps to 100Mbps. The granularity of rate is 64Kbps. interframe gap and preamble.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_ENABLE	invalid enable parameter
RT_ERR_INBW_RATE	invalid input bandwidth

rtk_rate_igrBandwidthCtrlRate_get

rtk_api_ret_t rtk_rate_igrBandwidthCtrlRate_get(rtk_port_t *port***, rtk_rate_t** **pRate***, rtk_enable_t** **pIfg_include***)**

Get port ingress bandwidth control

Defined in: rtk_api_ext.h

Parameters

port
Port id, 0~8

**pRate*
the pointer of rate of share meter

**pIfg_include*
the pointer of Register data

Comments

For RTL8309N, port0 and port 8's max speed could be 100Mbps, and max speed could only be 100Mbps for port1 to port 7. The rate unit is 64Kbps and the range is from 64Kbps to 100Mbps. The granularity of rate is 64Kbps. interframe gap and preamble.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_rate_egrBandwidthCtrlRate_set

rtk_api_ret_t rtk_rate_egrBandwidthCtrlRate_set(rtk_port_t *port***, rtk_rate_t** *rate***, rtk_enable_t** *ifg_include***)**

Set port egress bandwidth control

Defined in: rtk_api_ext.h

Parameters

port
Port id

rate
Rate of bandwidth control

ifg_include
the pointer of Register data

Comments

For RTL8309N, port0 and port 8's max speed could be 100Mbps, and max speed could only be 100Mbps for port from port 1 to port 7. The rate unit is 64Kbps and the range is from 64Kbps to 100Mbps. for rate calculation with or without interframe gap and preamble.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_ENABLE	invalid enable parameter
RT_ERR_QOS_EBW_RATE	invalid egress bandwidth rate

rtk_rate_egrBandwidthCtrlRate_get

rtk_api_ret_t rtk_rate_egrBandwidthCtrlRate_get(rtk_port_t *port*, **rtk_rate_t** **pRate*, **rtk_enable_t** **pIfg_include*)

Get port egress bandwidth control

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pRate*
the pointer of rate of bandwidth control

**pIfg_include*
the pointer of Register data

Comments

For RTL8309N, port0 and port 8's max speed could be 100Mbps, and max speed could only be 100Mbps for port from port 1 to port 7. The rate unit is 64Kbps and the range is from 64Kbps to 100Mbps. for rate calculation with or without interframe gap and preamble.

Return Codes

RT_ERR_OK	ok

RT_ERR_PORT_ID	Invalid port number
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	null pointer

rtk_qos_init

rtk_api_ret_t rtk_qos_init(rtk_queue_num_t *queueNum*)

Configure Qos with default settings

Defined in: rtk_api_ext.h

Parameters

queueNum
Queue number of each port(from 1 to 4)

Comments

This API will initialize related Qos function. First it will set the ASIC's queue number globally for all port. Then it will set priority to queue mapping table based on the queue number for all ports. And it will enable output and input flow control abilities.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_QUEUE_NUM	invalid queue number

rtk_qos_priSrcEnable_set

rtk_api_ret_t rtk_qos_priSrcEnable_set(rtk_port_t *port*, rtk_qosPriSrc_t *priSrc*, rtk_enable_t *enabled*)

Enable/disable Qos priority source for ingress port

Defined in: rtk_api_ext.h

Parameters

port
port id

priSrc
priority source id

enabled
DISABLED or ENABLED

Comments This API will enable Qos priority source for ingress port. The port id is from 0 to 8. priSrc are Port, 1Q, DSCP, IP adress, and CPU tag basded priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	Invalid input parameter

rtk_qos_priSrcEnable_get

rtk_api_ret_t rtk_qos_priSrcEnable_get(rtk_port_t *port***, rtk_qosPriSrc_t** *priSrc***, rtk_enable_t** ***pEnabled*****)**

Enable/disable Qos priority source for ingress port

Defined in: rtk_api_ext.h

Parameters

port
port id

priSrc
priority source id

**pEnabled*
Point to the status of qos priority source

Comments This API will get the status of Qos priority source for ingress port. The port id is from 0 to 8. priSrc are Port, 1Q, DSCP, IP adress, and CPU tag basded priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	error port id
RT_ERR_INPUT	Invalid input parameter

rtk_qos_priSel_set

rtk_api_ret_t rtk_qos_priSel_set(rtk_priority_select_t ***pPriDec*****)**

Configure the priority order among different priority mechanisms.

Defined in: rtk_api_ext.h

Parameters **pPriDec*
pointer point to priority level structure.

Comments (1)For 8309N, there are 4 types of priority source that could be set arbitration level, which are ACL-based, DSCP-based, 1Q-based, Port-based priority. Each one could be set to level from 0 to 4.
(2)ASIC will follow user's arbitration level setting to select internal priority for receiving frame. If two priority mechanisms are the same level, the ASIC will choose the higher priority to assign for the receiving frame.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_INPUT	Invalid input parameter.

rtk_qos_priSel_get

rtk_api_ret_t rtk_qos_priSel_get(rtk_priority_select_t **pPriDec***)**

Get the priority order configuration among different priority mechanism.

Defined in: rtk_api_ext.h

Parameters **pPriDec*
pointer point to priority level structure.

Comments (1)For 8309N, there are 4 types of priority mechanisms that could be set arbitration level, which are ACL-based, DSCP-based, 1Q-based, Port-based priority. Each one could be set to level from 1 to 4.
(2)ASIC will follow user's arbitration level setting to select internal priority for receiving frame. If two priority mechanisms are the same level, the ASIC will choose the higher priority to assign for the receiving frame.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_qos_1pPriRemap_set

rtk_api_ret_t rtk_qos_1pPriRemap_set(rtk_pri_t *dot1p_pri***, rtk_pri_t** *int_pri***)**

Configure 1Q priority mapping to internal absolute priority

Defined in: rtk_api_ext.h

Parameters

dot1p_pri
802.1p priority value, 0~7

int_pri
internal priority value, 0~3

Comments

When DOT1Q tagged packet has been received, 1Q tag priority has 3 bits, and RTL8309N only support 2 bit priority internally. So 3 bit 1Q tag priority has to be mapped to a 2 bit internal priority for further QOS operations.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_VLAN_PRIORITY	invalid 1p priority
RT_ERR_QOS_INT_PRIORITY	invalid internal priority

rtk_qos_1pPriRemap_get

rtk_api_ret_t rtk_qos_1pPriRemap_get(rtk_pri_t *dot1p_pri***, rtk_pri_t** **pInt_pri***)**

Get 1Q priorities mapping to internal absolute priority

Defined in: rtk_api_ext.h

Parameters

dot1p_pri
802.1p priority value

**pInt_pri*
the pointer of internal priority value

Comments

Priority of 802.1Q assignment for internal asic priority, and it is used for queue usage and packet scheduling.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_VLAN_PRIORITY	Invalid 1p priority
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_qos_dscpPriRemap_set

rtk_api_ret_t rtk_qos_dscpPriRemap_set(rtk_dscp_t *dscp_value***, rtk_pri_t** *int_pri***)**

Set DSCP-based priority

Defined in: rtk_api_ext.h

Parameters

dscp_value
dscp value(0~63)

int_pri
internal priority value

Comments

This API can be called to configure a dscp value to a 2-bit internal priority value. RTL8309N support 64 DSCP values and 2-bit internal priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_QOS_INT_PRIORITY	invalid internal priority
RT_ERR_QOS_DSCP_VALUE	invalid DSCP value

rtk_qos_dscpPriRemap_get

rtk_api_ret_t rtk_qos_dscpPriRemap_get(rtk_dscp_t *dscp_value***, rtk_pri_t** **pInt_pri***)**

Get DSCP-based priority

Defined in: rtk_api_ext.h

Parameters

dscp_value
dscp code

**pInt_pri*
the pointer of internal priority value

Comments

This API can be called to get a 2-bit internal priority value for a specified dscp value. RTL8309N support 64 DSCP values and 2-bit internal priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_QOS_DSCP_VALUE	Invalid DSCP value
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_qos_portPri_set

rtk_api_ret_t rtk_qos_portPri_set(rtk_port_t *port*, rtk_pri_t *int_pri*)

Configure priority usage to each port

Defined in: rtk_api_ext.h

Parameters

port
Port id.

int_pri
internal priority value

Comments The API can set port-based priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_QOS_INT_PRIORITY	invalid internal priority

rtk_qos_portPri_get

rtk_api_ret_t rtk_qos_portPri_get(rtk_port_t *port*, rtk_pri_t **pInt_pri*)

Get priority usage to each port

Defined in: rtk_api_ext.h

Parameters

port
Port id.

**pInt_pri*
the pointer of internal priority value

Comments This API can be called to get port-based priority

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_qos_priMap_set

rtk_api_ret_t rtk_qos_priMap_set(rtk_port_t *port*, **rtk_queue_num_t** *queue_num*, **rtk_qos_pri2queue_t** **pPri2qid*)

Set internal priority mapping to queue ID for different queue number

Defined in: rtk_api_ext.h

Parameters

port
port id

queue_num
Queue number usage

**pPri2qid*
pointer point to Priority and queue ID mapping table

Comments

ASIC supports priority mapping to queue with different queue number from 1 to 4. For different queue numbers usage, ASIC supports different internal available queue IDs. pPri2qid has 4 members, which is from queue id for priority 0 to queue id for priority 3.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_QUEUE_NUM	invalid queue number
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_qos_priMap_get

rtk_api_ret_t rtk_qos_priMap_get(rtk_port_t *port*, **rtk_queue_num_t** *queue_num*, **rtk_qos_pri2queue_t** **pPri2qid*)

Get priority to queue ID mapping table parameters

Defined in: rtk_api_ext.h

Parameters

port
port id

queue_num
queue number usage

**pPri2qid*
pointer point to Priority and queue ID mapping table

Comments

ASIC supports priority mapping to queue with different queue number from 1 to 4. For different queue numbers usage, ASIC supports different internal available queue IDs. pPri2qid has 4 members, which is from queue id for priority 0 to queue id for priority 3.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_QUEUE_NUM	invalid queue number
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_qos_1pRemarkEnable_set

rtk_api_ret_t rtk_qos_1pRemarkEnable_set(rtk_port_t *port***, rtk_enable_t** *enabled***)**

Enable 802.1P remarking ability

Defined in: rtk_api_ext.h

Parameters

port
port number

enabled
DISABLED or ENABLED

Comments

This API can be called to enable or disable 802.1P remarking ability for a port of RTL8309N. The 802.1P remarking function here is used to assign a new 3-bit priority for a tx packet instead of its old 2-bit priority. The assignment is based on the user's definition.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id

rtk_qos_1pRemarkEnable_get

rtk_api_ret_t rtk_qos_1pRemarkEnable_get(rtk_port_t *port*, **rtk_enable_t** **pEnable*)

Get enabled status of 802.1P remarking ability

Defined in: rtk_api_ext.h

Parameters

port
port number

**pEnable*
pointer point to the ability status

Comments

This API can be called to get the enabled status of 802.1P remarking ability for a port of RTL8309N. The 802.1P remarking function here is used to assign a new 3-bit priority for a tx packet instead of its old 2-bit priority. The assignment is based on the user's definition.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_qos_1pRemark_set

rtk_api_ret_t rtk_qos_1pRemark_set(rtk_pri_t *int_pri*, **rtk_pri_t** *dot1p_pri*)

Set 802.1P remarking priority

Defined in: rtk_api_ext.h

Parameters

int_pri
Packet internal priority(0~4)

dot1p_pri
802.1P priority(0~7)

Comments

RTL8309N support 2-bit internal priority and 3-bit dot1p priotiy. User can use this API to map a 2-bit internal priority to a 3-bit dot1p priority.

Return Codes

RT_ERR_OK	ok

RT_ERR_FAILED	failed
RT_ERR_VLAN_PRIORITY	Invalid 1p priority
RT_ERR_QOS_INT_PRIORITY	Invalid internal priority

rtk_qos_1pRemark_get

rtk_api_ret_t rtk_qos_1pRemark_get(rtk_pri_t *int_pri*, **rtk_pri_t** **pDot1p_pri*)

Get 802.1P remarking priority

Defined in: rtk_api_ext.h

Parameters

int_pri
Packet priority(0~4)

**pDot1p_pri*
the pointer of 802.1P priority(0~7)

Comments

This API can be called to get a 2-bit internal priority and a 3-bit dot1p priority mapping.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_QOS_INT_PRIORITY	Invalid internal priority

rtk_trap_unknownMcastPktAction_set

rtk_api_ret_t rtk_trap_unknownMcastPktAction_set(rtk_port_t *port*, **rtk_mcast_type_t** *type*, **rtk_trap_mcast_action_t** *mcast_action*)

Set behavior of unknown multicast

Defined in: rtk_api_ext.h

Parameters

port
port id

type
unknown multicast packet type

mcast_action
unknown multicast action

Comments

When receives an unknown multicast packet, switch may forward, drop this packet The unknown multicast packet type is as following:
- MCAST_IPV4
- MCAST_IPV6 The unknown multicast action is as following:
- MCAST_ACTION_FORWARD
- MCAST_ACTION_DROP

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_INPUT	Invalid input parameter

rtk_trap_unknownMcastPktAction_get

rtk_api_ret_t rtk_trap_unknownMcastPktAction_get(rtk_port_t *port*, **rtk_mcast_type_t** *type*, **rtk_trap_mcast_action_t** **pMcast_action*)

Get behavior of unknown multicast

Defined in: rtk_api_ext.h

Parameters

port
port id

type
unknown multicast packet type

**pMcast_action*
the pointer of unknown multicast action

Comments

When receives an unknown multicast packet, switch may forward, drop this packet. The unknown multicast packet type is as following:
- MCAST_IPV4
- MCAST_IPV6 The unknown multicast action is as following:
- MCAST_ACTION_FORWARD
- MCAST_ACTION_DROP

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_INPUT	invalid input parameter

RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_trap_igmpCtrlPktAction_set

rtk_api_ret_t rtk_trap_igmpCtrlPktAction_set(rtk_igmp_type_t *type*, **rtk_trap_igmp_action_t** *igmp_action***)**

Set IGMP/MLD trap function

Defined in: rtk_api_ext.h

Parameters

type
IGMP/MLD packet type

igmp_action
IGMP/MLD action

Comments

This API can set both IPv4 IGMP/IPv6 MLD with/without PPPoE header trapping function. All 4 kinds of IGMP/MLD function can be set separately. The IGMP/MLD packet type is as following:
- IGMP_IPV4
- IGMP_MLD
- IGMP_PPPOE_IPV4
- IGMP_PPPOE_MLD The IGMP/MLD action is as following:
- IGMP_ACTION_FORWARD IGMP_ACTION_COPY2CPU
- IGMP_ACTION_TRAP2CPU IGMP_ACTION_DROP

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	Invalid input parameter
RT_ERR_NOT_ALLOWED	Actions not allowed by the function

rtk_trap_igmpCtrlPktAction_get

rtk_api_ret_t rtk_trap_igmpCtrlPktAction_get(rtk_igmp_type_t *type*, **rtk_trap_igmp_action_t** **pIgmp_action***)**

Get IGMP/MLD trap function

Defined in: rtk_api_ext.h

Parameters

type
IGMP/MLD packet type

**pIgmp_action*
the pointer of IGMP/MLD action

Comments

This API can get both IPv4 IGMP/IPv6 MLD with/without PPPoE header trapping function. All 4 kinds of IGMP/MLD function can be set seperately. The IGMP/MLD packet type is as following:
- IGMP_IPV4
- IGMP_MLD
- IGMP_PPPOE_IPV4
- IGMP_PPPOE_MLD The IGMP/MLD action is as following:
- IGMP_ACTION_FORWARD
- IGMP_ACTION_TRAP2CPU

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	Invalid input parameter
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_vlan_init

rtk_api_ret_t rtk_vlan_init(*void***)**

Initialize VLAN

Defined in: rtk_api_ext.h

Parameters

void

Comments

VLAN function is disabled for ASIC after reset by default. User has to call this API to enable VLAN before using it. And It will set a default VLAN(vid 1) including all ports and set all ports's vlan index pointed to the default VLAN. So all port's PVID are 1.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_FAILED	failed

rtk_vlan_set

rtk_api_ret_t rtk_vlan_set(rtk_vlan_t *vid***, rtk_portmask_t** *mbrmsk***, rtk_portmask_t** *untagmsk***, rtk_fid_t** *fid***)**

Set a VLAN entry

Defined in: rtk_api_ext.h

Parameters

vid
VLAN ID to configure, should be 1~4094

mbrmsk
VLAN member set portmask

untagmsk
VLAN untag set portmask

fid
filtering database id, should be 0

Comments

There are 16 VLAN entry supported for RTL8309N. User could configure the member port set and untag member port set for specified vid through this API. The vid is from 0 to 4095. The vid 0 is used for priority tagged frames which is treated as untagged frames. The vid 4095 is reserved for further usage. The portmask's bit N means port N. For example, mbrmask 0x17 = 010111 means that port 0,1,2,4 are in the vlan's member port set. FID is for SVL/IVL usage, and the range is from 0 to 3. RTL8309N can only support 4 filtering database with the use of FID.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_VLAN_VID	Invalid vid
RT_ERR_PORT_MASK	invalid port mask
RT_ERR_INPUT	Invalid input parameter
RT_ERR_TBL_FULL	Input table full

rtk_vlan_get

rtk_api_ret_t rtk_vlan_get(rtk_vlan_t *vid***, rtk_portmask_t** **pMbrmsk***,* **rtk_portmask_t** **pUntagmsk***,* **rtk_fid_t** **pFid***)*

Get a VLAN entry

Defined in: rtk_api_ext.h

Parameters

vid
VLAN ID to configure

**pMbrmsk*
VLAN member set portmask

**pUntagmsk*
VLAN untag set portmask

**pFid*
filtering database id

Comments

There are 16 VLAN entry supported for RTL8309N. User could configure the member port set and untag member port set for specified vid through this API. The vid is from 0 to 4095. The vid 0 is used for priority tagged frames which is treated as untagged frames. The vid 4095 is reserved for further usage. The portmask's bit N means port N. For example, mbrmask 0x17 = 010111 means that port 0,1,2,4 are in the vlan's member port set. FID is for SVL/IVL usage, and the range is from 0 to 3. RTL8309N can only support 4 filtering database with the use of FID.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_VLAN_VID	Invalid vid
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_VLAN_ENTRY_NOT_FOUND	specified vlan entry not found

rtk_vlan_destroy

rtk_api_ret_t rtk_vlan_destroy(rtk_vlan_t *vid*)

delete a vlan entry from vlan table with specified vid

Defined in: rtk_api_ext.h

Parameters

vid
VLAN ID to configure

Comments

This API can be called to delet a vlan entry from vlan table with specified vid. After it is called, the content of vlan entry will set to zero.

Return Codes

RT_ERR_OK	ok
RT_ERR_VLAN_VID	Invalid vid
RT_ERR_VLAN_ENTRY_NOT_FOUND	Specified vlan entry not found

rtk_vlan_portPvid_set

rtk_api_ret_t rtk_vlan_portPvid_set(rtk_port_t *port***, rtk_vlan_t** *pvid***, rtk_pri_t** *priority***)**

Set port to specified VLAN ID(PVID)

Defined in: rtk_api_ext.h

Parameters

port
Port id

pvid
Specified VLAN ID

priority
802.1p priority for the PVID, 0~3 for RTL8309N

Comments

The API is used for Port-based VLAN. The untagged frame received from the port will be classified to the specified port-based VLAN and assigned to the specified priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_VLAN_VID	Invalid vid
RT_ERR_VLAN_PRIORITY	Invalid 1p priority
RT_ERR_VLAN_ENTRY_NOT_FOUND	Specified vlan entry not found

rtk_vlan_portPvid_get

rtk_api_ret_t rtk_vlan_portPvid_get(rtk_port_t *port***, rtk_vlan_t** **pPvid***,* **rtk_pri_t** **pPriority***)*

Get VLAN ID(PVID) on specified port

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pPvid*
Specified VLAN ID

**pPriority*
802.1p priority for the PVID

Comments

The API is used for Port-based VLAN. The untagged frame received from the port will be classified to the specified port-based VLAN and assigned to the specified priority.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_vlan_portIFilterEnable_set

rtk_api_ret_t rtk_vlan_portIFilterEnable_set(rtk_port_t *port*, rtk_enable_t *igr_filter*)

Set VLAN ingress for each port

Defined in: rtk_api_ext.h

Parameters

port
Port id, no use for RTL8309N

igr_filter
VLAN ingress function enable status

Comments

RTL8309N use one vlan ingress filter configuration for whole system, not for each port, so any port you set will affect all ports's ingress filter setting. While VLAN function is enabled, ASIC will decide VLAN ID for each received frame and get member ports for this vlan from VLAN table. If packets ingress port is in VLAN, ASIC will drop the received frame if VLAN ingress filter function is enabled.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_PORT_ID	invalid port id

rtk_vlan_portIFilterEnable_get

rtk_api_ret_t rtk_vlan_portIFilterEnable_get(rtk_port_t *port***, rtk_enable_t** **pIgr_filter***)**

get VLAN ingress for each port

Defined in: rtk_api_ext.h

Parameters

port
Port id, no use for RTL8309N

**pIgr_filter*
the pointer of VLAN ingress function enable status

Comments

RTL8309N use one ingress filter configuration for whole system, not for each port, so any port you set will affect all ports ingress filter setting. While VLAN function is enabled, ASIC will decide VLAN ID for each received frame and get belonged member ports from VLAN table. If received port is not belonged to VLAN member ports, ASIC will drop received frame if VLAN ingress function is enabled.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_vlan_portAcceptFrameType_set

rtk_api_ret_t rtk_vlan_portAcceptFrameType_set(rtk_port_t *port***, rtk_vlan_acceptFrameType_t** *accept_frame_type***)**

Set VLAN support frame type

Defined in: rtk_api_ext.h

Parameters

port
Port id

accept_frame_type
accept frame type

Comments The API is used for ingress port to check 802.1Q tagged frames. The ingress ports's accept frame type could be set to values as follows: ACCEPT_FRAME_TYPE_AL ACCEPT_FRAME_TYPE_TAG_ONLY ACCEPT_FRAME_TYPE_UNTAG_ONLY

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_VLAN_ACCEPT_FRAME_TYPE	invalid accept frame type

rtk_vlan_portAcceptFrameType_get

rtk_api_ret_t rtk_vlan_portAcceptFrameType_get(rtk_port_t *port*, rtk_vlan_acceptFrameType_t **pAccept_frame_type*)

Get VLAN support frame type

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pAccept_frame_type*
accept frame type

Comments The API is used for ingress port to check 802.1Q tagged frames. The ingress ports's accept frame type could be set to values as follows: ACCEPT_FRAME_TYPE_AL ACCEPT_FRAME_TYPE_TAG_ONLY ACCEPT_FRAME_TYPE_UNTAG_ONLY

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_VLAN_ACCEPT_FRAME_TYPE	Invalid accept frame type

rtk_stp_mstpState_set

rtk_api_ret_t rtk_stp_mstpState_set(rtk_stp_msti_id_t *msti***, rtk_port_t** *port***, rtk_stp_state_t** *stp_state***)**

Configure spanning tree state per port

Defined in: rtk_api_ext.h

Parameters

msti
Multiple spanning tree instance, no use for RTL8309N

port
Port id

stp_state
Spanning tree state

Comments

Because RTL8309N does not support multiple spanning tree, so msti is no use. There are four states supported by ASIC. STP_STATE_DISABLED STP_STATE_BLOCKING STP_STATE_LEARNING STP_STATE_FORWARDING

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_MSTP_STATE	Invalid spanning tree status

rtk_stp_mstpState_get

rtk_api_ret_t rtk_stp_mstpState_get(rtk_stp_msti_id_t *msti***, rtk_port_t** *port***, rtk_stp_state_t** **pStp_state***)**

Get Configuration of spanning tree state per port

Defined in: rtk_api_ext.h

Parameters

msti
Multiple spanning tree instance, no use for RTL8309N

port
Port id

**pStp_state*
the pointer of Spanning tree state

Comments Because RTL8309N does not support multiple spanning tree, so msti is no use. There are four states supported by ASIC. STP_STATE_DISABLED STP_STATE_BLOCKING STP_STATE_LEARNING STP_STATE_FORWARDING

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_l2_addr_add

rtk_api_ret_t rtk_l2_addr_add(rtk_mac_t *pMac, rtk_fid_t *fid*, rtk_l2_ucastAddr_t **pL2_data*)

Add a unicast entry into LUT table

Defined in: rtk_api_ext.h

Parameters

**pMac*
pointer point to structure of unicastmac address

fid
fid value(0~3)

**pL2_data*
the pointer of Spanning tree state

Comments (1)The lut has a 4-way entry due to an index. If the macAddress has existed in the lut, it will update the entry with the user's defined entry content, otherwise the function will find an empty entry to put it. When the index is full, it will find a dynamic & unauth unicast macAddress entry to replace with it.
(2)If the mac address has been added into LUT, function return value is SUCCESS, *pEntryaddr is recorded the entry address of the Mac address stored. If all the four entries can not be replaced, it will return a RTL8309N_LUT_FULL error, you can delete one of them manually and rewrite the unicast address.
(3) The age of the look up table entry could be: AGE_TIME_OUT AGE_TIME_100S AGE_TIME_200S AGE_TIME_300S

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_INPUT	Invalid input parameter
RT_ERR_MAC	invalid mac address
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_L2_INDEXTBL_FULL	The L2 index table is full

rtk_l2_addr_get

rtk_api_ret_t rtk_l2_addr_get(rtk_mac_t **pMac*, **rtk_fid_t** *fid*, **rtk_l2_ucastAddr_t** **pL2_data*)

Get a unicast entry from LUT table

Defined in: rtk_api_ext.h

Parameters

**pMac*
6 bytes unicast(I/G bit is 0) mac address to be gotten

fid
filtering database id, could be any value for RTL8309N switch

**pL2_data*
the mac address attributes

Comments

(1)The lut has a 4-way entry due to an index. If the macAddress has existed in the lut, This API will return the entry content.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_L2_FID	invalid fid
RT_ERR_MAC	invalid mac address
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_L2_ENTRY_NOTFOUND	Specified entry not found

rtk_l2_addr_del

rtk_api_ret_t rtk_l2_addr_del(rtk_mac_t **pMac*, **rtk_fid_t** *fid*)

Delete a LUT unicast entry

Defined in: rtk_api_ext.h

Parameters

**pMac*
6 bytes unicast mac address to be deleted

fid
filtering database id, could be any value for RTL8309N switch

Comments

If the mac has existed in the LUT, it will be deleted. Otherwise, it will return RT_ERR_L2_ENTRY_NOTFOUND.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_MAC	Wrong mac address, must be unicast mac
RT_ERR_L2_FID	invalid fid
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_L2_ENTRY_NOTFOUND	Specified entry not found

rtk_l2_mcastAddr_add

rtk_api_ret_t rtk_l2_mcastAddr_add(rtk_mac_t **pMac*, rtk_fid_t *fid*, rtk_portmask_t *portmask*)

Add a LUT multicast entry

Defined in: rtk_api_ext.h

Parameters

**pMac*
6 bytes unicast mac address to be deleted

fid
filtering database id, could be any value for RTL8309N switch

portmask
Port mask to be forwarded to

Comments

If the multicast mac address already existed in the LUT, it will udpate the port mask of the entry. Otherwise, it will find an empty or asic auto learned entry to write. If all the entries with the same hash value can't be replaced, ASIC will return a RT_ERR_L2_INDEXTBL_FULL error.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_MASK	Invalid port mask
RT_ERR_MAC	invalid mac address
RT_ERR_NULL_POINTER	input parameter is null pointer

RT_ERR_L2_INDEXTBL_FULL	the L2 index table is full

rtk_l2_mcastAddr_get

rtk_api_ret_t rtk_l2_mcastAddr_get(rtk_mac_t *_pMac_, rtk_fid_t _fid_, rtk_portmask_t *_pPortmask_)

Get a LUT multicast entry

Defined in: rtk_api_ext.h

Parameters

**pMac*
6 bytes multicast(I/G bit is 0) mac address to be gotten

fid
filtering database id, could be any value for RTL8309N switch

**pPortmask*
the pointer of port mask

Comments

If the multicast mac address existed in LUT, it will return the port mask where the packet should be forwarded to, Otherwise, it will return a RT_ERR_L2_ENTRY_NOTFOUND error.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_MAC	invalid mac address
RT_ERR_L2_ENTRY_NOTFOUND	specified entry not found

rtk_l2_mcastAddr_del

rtk_api_ret_t rtk_l2_mcastAddr_del(rtk_mac_t *_pMac_, rtk_fid_t _fid_)

Delete a LUT unicast entry

Defined in: rtk_api_ext.h

Parameters

**pMac*
6 bytes multicast(I/G bit is 1) mac address to be gotten

fid
filtering database id, could be any value for RTL8309N switch

Comments If the mac has existed in the LUT, it will be deleted. Otherwise, it will return RT_ERR_L2_ENTRY_NOTFOUND.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_MAC	invalid mac address
RT_ERR_L2_FID	invalid fid
RT_ERR_L2_ENTRY_NOTFOUND	specified entry not found

rtk_l2_limitLearningSysCntEnable_set

rtk_api_ret_t rtk_l2_limitLearningSysCntEnable_set(rtk_enable_t *enabled***)**

Enable system mac learning limit function

Defined in: rtk_api_ext.h

Parameters *enabled*
ENABLED or DISABLED

Comments For RTL8309N, mac learning limit function can be enabled or disabled for a whole system.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_l2_limitLearningSysCntEnable_get

rtk_api_ret_t rtk_l2_limitLearningSysCntEnable_get(rtk_enable_t **pEnabled***)**

Get enabled status of system mac learning limit function

Defined in: rtk_api_ext.h

Parameters **pEnabled*
ENABLED or DISABLED

Comments For RTL8309N, mac learning limit function can be enabled or disabled for a whole system.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_l2_limitLearningSysCnt_set

rtk_api_ret_t rtk_l2_limitLearningSysCnt_set(rtk_mac_cnt_t *mac_cnt***, rtk_portmask_t** *mergeMask***)**

Set system mac limitting max value and port merge mask

Defined in: rtk_api_ext.h

Parameters

mac_cnt
system mac limitting value

mergeMask
a set describing the ports whose port mac limitting value are counted into system mac limitting value

Comments (1) This API can be called to set system mac limitting max value and port merge mask. (2) mac_cnt: the whole system mac limitting max value, it's value is from 0 - 0xFF; (3) mergeMask: the ports whose mac limitting counter value are counted into the system mac limitting counter value, it's value is from 0 - 0x1FF. If bit n is 1, it means that port n is counted.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_INPUT	invalid input parameter

rtk_l2_limitLearningCnt_set

rtk_api_ret_t rtk_l2_limitLearningCnt_set(rtk_port_t *port***, rtk_mac_cnt_t** *mac_cnt***)**

Set per-Port auto learning limit counter max value

Defined in: rtk_api_ext.h

Parameters

port
Port id

mac_cnt
mac limit counter value

Comments

(1) Per port mac learning limit function can be enabled or disabled independently;
(2) mac_cnt: port mac learning limit max value, it's value is from 0 - 0x1F;

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id.
RT_ERR_LIMITED_L2ENTRY_NUM	invalid limited L2 entry number

rtk_l2_limitLearningCnt_get

rtk_api_ret_t rtk_l2_limitLearningCnt_get(rtk_port_t *port*, **rtk_mac_cnt_t** **pMac_cnt*)

Get per-Port auto learning limit counter max value

Defined in: rtk_api_ext.h

Parameters

port
Port id.

**pMac_cnt*
mac limit counter value

Comments

(1) Per port mac learning limit function can be enabled or disabled independently;
(2) mac_cnt: port mac learning limit max value, it's value is from 0 - 0x1F;

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id.
RT_ERR_NULL_POINTER	input parameter is NULL pointer

rtk_l2_limitLearningCntAction_set

rtk_api_ret_t rtk_l2_limitLearningCntAction_set(rtk_port_t *port*, **rtk_l2_limitLearnCntAction_t** *action*)

Configure auto learn over limit number action.

Defined in: rtk_api_ext.h

Parameters

port
port id, no usage for RTL8309N

action
Auto learning entries limit number

Comments

(1)The API can set SA unknown packet action while auto learn limit number is over. The action symbol as following: LIMIT_LEARN_CNT_ACTION_DROP LIMIT_LEARN_CNT_ACTION_TO_CPU

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port number.
RT_ERR_NOT_ALLOWED	Invalid learn over action

rtk_l2_limitLearningCntAction_get

rtk_api_ret_t rtk_l2_limitLearningCntAction_get(rtk_port_t *port*, **rtk_l2_limitLearnCntAction_t** **pAction*)

Get auto learn over limit number action.

Defined in: rtk_api_ext.h

Parameters

port
port id, no usage for RTL8309N

**pAction*
Learn over action

Comments

(1)The API can get SA unknown packet action while auto learn limit number is over. The action symbol as following: LIMIT_LEARN_CNT_ACTION_DROP LIMIT_LEARN_CNT_ACTION_TO_CPU

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id.
RT_ERR_NULL_POINTER	null pointer
	specified entry not found

rtk_l2_learningSysCnt_get

rtk_api_ret_t rtk_l2_learningSysCnt_get(rtk_mac_cnt_t *_pMac_cnt_)

Get current value of system auto learning mac counter

Defined in: rtk_api_ext.h

Parameters

*_pMac_cnt_
port id, no usage for RTL8309N

Comments

(1)The API can get SA unknown packet action while auto learn limit number is over. The action symbol as following: LIMIT_LEARN_CNT_ACTION_DROP LIMIT_LEARN_CNT_ACTION_TO_CPU

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_NULL_POINTER	Invalid port id.

rtk_l2_learningCnt_get

rtk_api_ret_t rtk_l2_learningCnt_get(rtk_port_t _port_, rtk_mac_cnt_t *_pMac_cnt_)

Get current value of per-Port auto learning counter

Defined in: rtk_api_ext.h

Parameters

*_port_
Port id.

*_pMac_cnt_
ASIC auto learning entries number

Comments

The API can get per-port ASIC auto learning number

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port number.
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_cpu_enable_set

rtk_api_ret_t rtk_cpu_enable_set(rtk_enable_t *enabled*)

Enable cpu port ability

Defined in: rtk_api_ext.h

Parameters

enabled
enable or disable

Comments

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_cpu_enable_get

rtk_api_ret_t rtk_cpu_enable_get(rtk_enable_t **pEnabled*)

Enable cpu port ability

Defined in: rtk_api_ext.h

Parameters

**pEnabled*
enable or disable

Comments

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_cpu_tagPort_set

rtk_api_ret_t rtk_cpu_tagPort_set(rtk_port_t *port*, rtk_enable_t *enTag*)

Set cpu port and insert cpu tag

Defined in: rtk_api_ext.h

Parameters

port
port id

enTag
enable insert cpu tag, enable or disable

Comments

Return Codes

RT_ERR_PORT_ID	invalid port id
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_cpu_tagPort_get

rtk_api_ret_t rtk_cpu_tagPort_get(rtk_port_t **pPort*, **rtk_enable_t** **pEnTag***)**

Get cpu port and insert cpu tag status

Defined in: rtk_api_ext.h

Parameters

**pPort*
port id(0

**pEnTag*
enable insert cpu tag, enable or disable

Comments

Return Codes

RT_ERR_PORT_ID	invalid port id
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_mirror_portBased_set

rtk_api_ret_t rtk_mirror_portBased_set(rtk_port_t *mirroring_port*, **rtk_portmask_t** **pMirrored_rx_portmask*, **rtk_portmask_t** **pMirrored_tx_portmask***)**

Set port mirror function parameters

Defined in: rtk_api_ext.h

Parameters

mirroring_port
Monitor port, 7 means no monitor port

**pMirrored_rx_portmask*
the pointer of Rx mirror port mask

**pMirrored_tx_portmask*
the pointer of Tx mirror port mask

Comments The API is called to set mirroring port and mirrored rx and tx port mask.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_MASK	Invalid port mask
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_mirror_portBased_get

rtk_api_ret_t rtk_mirror_portBased_get(rtk_port_t **pMirroring_port*, **rtk_portmask_t** **pMirrored_rx_portmask*, **rtk_portmask_t** **pMirrored_tx_portmask*)

Get port mirror function parameters

Defined in: rtk_api_ext.h

Parameters

**pMirroring_port*
Monitor port, 7 means no monitor port

**pMirrored_rx_portmask*
the pointer Monitor port, 7 means no monitor port

**pMirrored_tx_portmask*
the pointer of Rx mirror port mask

Comments The API is to set mirror function of source port and mirror port.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_mirror_macBased_set

rtk_api_ret_t rtk_mirror_macBased_set(rtk_mac_t **macAddr*, rtk_enable_t *enabled*)

Set Mac address for mirror packet

Defined in: rtk_api_ext.h

Parameters

**macAddr*
mirrored mac address, it could be SA or DA of the packet

enabled
enable mirror packet by mac address

Comments

The API is to set mirror function of source port and mirror port.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_mirror_macBased_get

rtk_api_ret_t rtk_mirror_macBased_get(rtk_mac_t **macAddr*, uint32 **pEnabled*)

get Mac address for mirror packet

Defined in: rtk_api_ext.h

Parameters

**macAddr*
mirrored mac address, it could be SA or DA of the packet

**pEnabled*
mirrored mac address, it could be SA or DA of the packet

Comments

The API is to set mirror function of source port and mirror port.

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_dot1x_unauthPacketOper_set

rtk_api_ret_t rtk_dot1x_unauthPacketOper_set(rtk_port_t *port*, **rtk_dot1x_unauth_action_t** *unauth_action***)**

Set 802.1x unauth action configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id, no use for RTL8309N switch

unauth_action
802.1X unauth action

Comments

This API can set 802.1x unauth action configuration, for RTL8309N switch, the action is by whole system, so port could be any value of 0~8. The unauth action is as following: DOT1X_ACTION_DROP DOT1X_ACTION_TRAP2CPU

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_DOT1X_PROC	Unauthorized behavior error

rtk_dot1x_unauthPacketOper_get

rtk_api_ret_t rtk_dot1x_unauthPacketOper_get(rtk_port_t *port*, **rtk_dot1x_unauth_action_t** **pUnauth_action***)**

Get 802.1x unauth action configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id, no use for RTL8309N switch

**pUnauth_action*
the pointer of 802.1X unauth action

Comments

This API can set 802.1x unauth action configuration, for RTL8309N switch, the action is by whole system, so port could be any value of 0~8. The unauth action is as following: DOT1X_ACTION_DROP DOT1X_ACTION_TRAP2CPU

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_dot1x_portBasedEnable_set

rtk_api_ret_t rtk_dot1x_portBasedEnable_set(rtk_port_t *port***, rtk_enable_t** *enabled***)**

Set 802.1x port-based enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

enabled
enable or disable

Comments

The API can update the port-based port enable register content. If a port is 802.1x port based network access control "enabled", it should be authenticated so packets from that port won't be dropped or trapped to CPU. The status of 802.1x port-based network access control is as following:
- DISABLED
- ENABLED

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id

rtk_dot1x_portBasedEnable_get

rtk_api_ret_t rtk_dot1x_portBasedEnable_get(rtk_port_t *port***, rtk_enable_t** **pEnable***)**

Get 802.1x port-based enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pEnable*
the pointer of enable or disable

Comments The API can update the port-based port enable register content. If a port is 802.1x port based network access control "enabled", it should be authenticated so packets from that port won't be dropped or trapped to CPU. The status of 802.1x port-based network access control is as following:
- DISABLED
- ENABLED

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id

rtk_dot1x_portBasedAuthStatus_set

rtk_api_ret_t rtk_dot1x_portBasedAuthStatus_set(rtk_port_t *port*, **rtk_dot1x_auth_status_t** *port_auth***)**

Set 802.1x port-based enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

port_auth
The status of 802.1x port

Comments The authenticated status of 802.1x port-based network access control is as following: UNAUTH AUTH

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_DOT1X_PORTBASEDAUTH	Port

rtk_dot1x_portBasedAuthStatus_get

rtk_api_ret_t rtk_dot1x_portBasedAuthStatus_get(rtk_port_t *port*, **rtk_dot1x_auth_status_t** **pPort_auth***)*

Set 802.1x port-based enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pPort_auth*
The status of 802.1x port

Comments The authenticated status of 802.1x port-based network access control is as following: UNAUTH AUTH

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_DOT1X_PORTBASEDAUTH	Port

rtk_dot1x_portBasedDirection_set

rtk_api_ret_t rtk_dot1x_portBasedDirection_set(rtk_port_t *port*, **rtk_dot1x_direction_t** *port_direction***)**

Set 802.1x port-based operational direction configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

port_direction
Operation direction

Comments The operate controlled direction of 802.1x port-based network access control is as following: BOTH IN

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_DOT1X_PORTBASEDOPDIR	Port

rtk_dot1x_portBasedDirection_get

rtk_api_ret_t rtk_dot1x_portBasedDirection_get(rtk_port_t *port*, **rtk_dot1x_direction_t** **pPort_direction*)

Get 802.1x port-based operational direction configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pPort_direction*
the pointer of Operation direction

Comments

The operate controlled direction of 802.1x port-based network access control is as following: BOTH IN

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id
RT_ERR_NULL_POINTER	Input parameter is null pointer

rtk_dot1x_macBasedEnable_set

rtk_api_ret_t rtk_dot1x_macBasedEnable_set(rtk_port_t *port*, **rtk_enable_t** *enabled*)

Set 802.1x mac-based port enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

enabled
The status of 802.1x mac

Comments

If a port is 802.1x MAC based network access control "enabled", the incoming packets should be authenticated so packets from that port won't be dropped or trapped to CPU.

Return Codes

RT_ERR_OK	ok

RT_ERR_FAILED	failed
RT_ERR_PORT_ID	Invalid port id

rtk_dot1x_macBasedEnable_get

rtk_api_ret_t rtk_dot1x_macBasedEnable_get(rtk_port_t *port*, **rtk_enable_t** **pEnable*)

Get 802.1x mac-based port enable configuration

Defined in: rtk_api_ext.h

Parameters

port
Port id

**pEnable*
the pointer of the status of 802.1x mac

Comments

If a port is 802.1x MAC based network access control "enabled", the incoming packets should be authenticated so packets from that port won't be dropped or trapped to CPU.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_dot1x_macBasedDirection_set

rtk_api_ret_t rtk_dot1x_macBasedDirection_set(rtk_dot1x_direction_t *mac_direction*)

Set 802.1x mac-based operational direction configuration

Defined in: rtk_api_ext.h

Parameters

mac_direction
Operation direction

Comments

The operate controlled direction of 802.1x mac-based network access control is as following: BOTH IN

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_DOT1X_MACBASEDOPDIR	MAC

rtk_dot1x_macBasedDirection_get

rtk_api_ret_t rtk_dot1x_macBasedDirection_get(rtk_dot1x_direction_t **pMac_direction*)

Get 802.1x mac-based operational direction configuration

Defined in: rtk_api_ext.h

Parameters

**pMac_direction*
Operation direction

Comments

The operate controlled direction of 802.1x mac-based network access control is as following: BOTH IN

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	Input parameter is null pointer
RT_ERR_PORT_ID	Wrong port ID

rtk_dot1x_macBasedAuthMac_add

rtk_api_ret_t rtk_dot1x_macBasedAuthMac_add(rtk_mac_t **pAuth_mac*, rtk_fid_t *fid*)

Add an authenticated MAC to ASIC

Defined in: rtk_api_ext.h

Parameters

**pAuth_mac*
The authenticated MAC

fid
no use for RTL8309N

Comments

The API can add a 802.1x authorised MAC address to port. If the MAC does not exist in LUT, user can't add this MAC with authorised status.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_L2_ENTRY_NOTFOUND	Specified entry not found
RT_ERR_L2_FID	invalid fid
RT_ERR_MAC	invalid mac address
RT_ERR_DOT1X_MAC_PORT_MISMATCH	Auth MAC and port mismatch eror
RT_ERR_PORT_ID	invalid port id

rtk_dot1x_macBasedAuthMac_del

rtk_api_ret_t rtk_dot1x_macBasedAuthMac_del(rtk_mac_t **pAuth_mac*, **rtk_fid_t** *fid***)**

Delete an authenticated MAC to ASIC

Defined in: rtk_api_ext.h

Parameters

**pAuth_mac*
The authenticated MAC

fid
no use for RTL8309N

Comments

The API can delete a 802.1x authenticated MAC address to port. It only change the auth status of the MAC and won't delete it from LUT.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_L2_ENTRY_NOTFOUND	Specified entry not found
RT_ERR_DOT1X_MAC_PORT_MISMATCH	Auth MAC and port mismatch eror
RT_ERR_L2_FID	invalid fid

rtk_filter_igrAcl_init

rtk_api_ret_t rtk_filter_igrAcl_init(*void***)**

Initialize ACL

Defined in: rtk_api_ext.h

Parameters *void*

Comments The API init ACL module.

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed

rtk_filter_igrAcl_rule_add

rtk_api_ret_t rtk_filter_igrAcl_rule_add(rtk_filter_rule_t *_pRule_)

Add an acl rule into acl table

Defined in: rtk_api_ext.h

Parameters **pRule*
the pointer of rule structure

Comments (1)The API add an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT, RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_TBL_FULL	input table full
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_filter_igrAcl_rule_get

rtk_api_ret_t rtk_filter_igrAcl_rule_get(rtk_filter_rule_t *_pRule_)

Get ACL rule priority and action

Defined in: rtk_api_ext.h

Parameters **pRule*
the pointer of rule structure

Comments (1)The API add an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_filter_igrAcl_rule_del

rtk_api_ret_t rtk_filter_igrAcl_rule_del(rtk_filter_rule_t **pRule*)

Delete an acl rule from acl table

Defined in: rtk_api_ext.h

Parameters **pRule*
the pointer of rule structure

Comments (1)The API delet an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_OK	ok
RT_ERR_FAILED	failed
RT_ERR_INPUT	invalid input parameter
RT_ERR_NULL_POINTER	input parameter is null pointer

rtk_storm_filterEnable_set

rtk_api_ret_t rtk_storm_filterEnable_set(rtk_port_t *port***, rtk_rate_storm_group_t** *storm_type***, rtk_enable_t** *enabled***)**

Enable storm filter

Defined in: rtk_api_ext.h

Parameters

port
port id

storm_type
storm filter type

enabled
enable or disable

Comments

(1)The API delet an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter

rtk_storm_filterEnable_get

rtk_api_ret_t rtk_storm_filterEnable_get(rtk_port_t *port***, rtk_rate_storm_group_t** *storm_type***, rtk_enable_t** ***pEnabled***)**

Enable storm filter

Defined in: rtk_api_ext.h

Parameters

port
port id

storm_type
storm filter type

**pEnabled*
enable or disable

Comments (1)The API delet an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_FAILED	failed
RT_ERR_OK	ok
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter

rtk_storm_filterAttr_set

rtk_api_ret_t rtk_storm_filterAttr_set(rtk_port_t *port*, **rtk_rate_storm_group_t** *storm_type*, **rtk_storm_attr_t** **pStorm_data*)

Set storm filter attributes

Defined in: rtk_api_ext.h

Parameters

port
port id

storm_type
storm filter type

**pStorm_data*
storm filter data

Comments (1)The API delet an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_PORT_ID	invalid port id

RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_storm_filterStatus_set

rtk_api_ret_t rtk_storm_filterStatus_set(rtk_port_t *port***, rtk_rate_storm_group_t** *storm_type***, rtk_enable_t** *enabled***)**

Get storm filter attributes

Defined in: rtk_api_ext.h

Parameters

port
port id(0

storm_type
storm filter type

enabled
pointer point to structure describing storm filter data

Comments

(1)The API delet an ACL rule. phyport could be port 0~8, RTK_ACL_INVALID_PORT and RTK_ACL_ANYPORT;
(2)protocol could be : ACL_PRO_ETHER ACL_PRO_TCP ACL_PRO_UDP ACL_PRO_TCPUDP
(3)prority could be 0-3;
(4)action could be : ACL_ACT_DROP ACL_ACT_PERMIT ACL_ACT_TRAP2CPU ACL_ACT_MIRROR

Return Codes

RT_ERR_PORT_ID	invalid port id
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_INPUT	invalid input parameter
RT_ERR_FAILED	failed
RT_ERR_OK	ok
/	Auth MAC and port mismatch eror
RT_ERR_PORT_ID	externrtk_api_ret_trtk_storm_filterAttr_get(rtk_port_tport,rtk_rate_storm_group_tstorm_type,rtk_storm_attr_t

/

rtk_storm_filterStatus_set

Description:

Clearingstormfilterflag	
Input:	
port	port id
storm_type	storm filter type
enabled	enable or disable
Output:	
none	
Return:	
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_mib_get

rtk_api_ret_t rtk_mib_get(rtk_port_t *port*, **rtk_mib_counter_t** *counter*, **rtk_mib_cntValue_t** **pValue*)

Get storm filter flag status

Defined in: rtk_api_ext.h

Parameters

port
port id

counter
storm filter type

**pValue*
exceed storm filter, exceed or not

Comments

mib counter named MIB_TXBYTECNT and MIB_RXBYTECNT are counted by unit of byte. And the counter values are 64bits long. So when these mib counter value are needed to read out, pValue should be pointed to a array with 2 unsigned 32bits data elements. To read out other mib counter, the unit is packet and pValue is pointed to a unsigned 32 bits value.

Return Codes

RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_FAILED	failed
RT_ERR_OK	ok

/	Auth MAC and port mismatch eror
RT_ERR_PORT_ID	externrtk_api_ret_trtk_storm_filterStatus_get(rtk_port_tport,rtk_rate_storm_group_tstorm_type,uint32m_attr_t
	/
rtk_mib_get	
Description:	
Getmibcountervalue	
Input:	
port	port id
counter	mib counter type
Output:	enable or disable
pValue	pointer point to mib counter value
Return:	
RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_stat_port_reset

rtk_api_ret_t rtk_stat_port_reset(rtk_port_t *port*)

Reset per port MIB counter by port, and enable mib counter start to count.

Defined in: rtk_api_ext.h

Parameters

port
port id

Comments

This API can be called to enable mib counter, and reset port's mib counter to run.

Return Codes

RT_ERR_OK	ok
RT_ERR_PORT_ID	invalid port id
RT_ERR_STAT_PORT_CNTR_FAIL	Could not retrieve/reset Port Counter

rtk_eee_portEnable_set

rtk_api_ret_t rtk_eee_portEnable_set(rtk_port_t *port***,**
rtk_enable_t *enable***)**

Set enable ability of 100M EEE function.

Defined in: rtk_api_ext.h

Parameters

port
port id(0

enabled
enable 100M EEE ability

Comments

(1) This API can set 100M EEE function to the specific port.

The configuration of the port is as following:

- DISABLE

- ENABLE

Return Codes

Return:

RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_FAILED	failed
RT_ERR_OK	ok

rtk_eee_portEnable_get

rtk_api_ret_t rtk_eee_portEnable_get(rtk_port_t port**, rtk_enable_t** *pEnable**)***

Get port EEE status

Defined in: rtk_api_ext.h

Parameters

port
port id

* *pEnable*
In EEE state or not

Comments This API can get 100M EEE function to the specific port.

The EEE status of the port is as following:

- DISABLE

- ENABLE

Return Codes

Return:

RT_ERR_PORT_ID	invalid port id
RT_ERR_INPUT	invalid input parameter
RT_ERR_NULL_POINTER	input parameter is null pointer
RT_ERR_FAILED	failed
RT_ERR_OK	ok