

# 1 Module Reference

This chapter references all elements by modules.

## 1.1 GPY2xx APIs

This chapter describes the interface for accessing the GPY2xx.

**Table 1** Module Overview

Name	Description
<a href="#">Init APIs</a>	Group of functional APIs for initialization and cleanup.
<a href="#">MDIO Bus APIs</a>	Group of functional APIs for MDIO Bus access.
<a href="#">Link APIs</a>	Group of functional APIs for auto-negotiation, link status, etc.
<a href="#">Supported Link Mode</a>	Group of macros for supported link mode configuration via gpy2xx_link in Link APIs.
<a href="#">Advertised Link Mode</a>	Group of macros for advertised link mode configuration via gpy2xx_link in Link APIs.
<a href="#">Link Speed</a>	Group of macros for link speed configuration via gpy2xx_link in APIs gpy2xx_setup_forced, gpy2xx_config_aneg and gpy2xx_sgmii_config_aneg.
<a href="#">Duplex</a>	Group of macros for duplex mode configuration via gpy2xx_link in APIs gpy2xx_setup_forced, gpy2xx_config_aneg and gpy2xx_sgmii_config_aneg.
<a href="#">LED Function Config APIs</a>	Group of functional APIs for LED configuration.
<a href="#">External Interrupt APIs</a>	Group of functional APIs for external interrupt configuration.
<a href="#">Diagnosis and Test APIs</a>	Group of functional APIs for diagnosis and test.
<a href="#">Synchronous Ethernet (SyncE) Config APIs.</a>	Group of functional APIs for Synchronous Ethernet (SyncE) configuration. Note: Not supported on GPHY flavours (i.e part numbers): GPY212, GPY2XX.
<a href="#">SGMII Interface Config APIs</a>	Group of functional APIs for SGMII interface configuration.
<a href="#">Wake-on-LAN Flags</a>	Group of Wake-on-LAN flags.
<a href="#">Miscellaneous Config APIs</a>	Group of functional APIs for miscellaneous features.
<a href="#">Firmware Download APIs</a>	Group of functional APIs to download firmware into flash memory.
<a href="#">USXGMII_REACH APIs</a>	Group of functional APIs for USXGMII Reach configuration. Note: USXGMII is not supported for models without USXGMII capability.

### 1.1.1 Init APIs

Group of functional APIs for initialization and cleanup.

**Table 2** Init APIs Structure Overview

Name	Description
<a href="#">gpy2xx_device</a>	Data structure representing the GPHY entity. The user application uses this struct to communicate with GPHY APIs.
<a href="#">gpy2xx_id</a>	Member used by id in gpy2xx_device.

Table 3 Init APIs Enumerator Overview

Name	Description
<a href="#">gpy2xx_fwboot_mode</a>	Macros used by fw_memory in gpy2xx_id.

Table 4 Init APIs Function Overview

Name	Description
<a href="#">gpy2xx_init</a>	Initialization.
<a href="#">gpy2xx_poll_reset</a>	Poll PHY reset status.
<a href="#">gpy2xx_soft_reset</a>	Resets the PHY to its default state.
<a href="#">gpy2xx_uninit</a>	Cleanup.

### 1.1.1.1 gpy2xx\_init

#### Description

Initialization.

This is the first API called by the user application before any other API. This API checks for MDIO access to the given PHY address and verifies the OUI (Operationally Unique ID). It reads and stores other information, such as manufacturer data, firmware/API version, default supported & advertised link configuration & status. This is filled into the [gpy2xx\\_device](#) structure. The user application must provide proper values/function-pointers for mdiobus\_read, mdiobus\_write, mdiobus\_data, and smdio\_addr in the [gpy2xx\\_device](#) structure.

#### Prototype

```
int gpy2xx_init (
    struct gpy2xx_device * phy );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.1.2 gpy2xx\_uninit

#### Description

Cleanup.

This is last API called by the user application for un-initialization purposes. The application has to implement/modify this API to unconfigure/release resources, disable interrupts etc.

### Prototype

```
int gpy2xx_uninit (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> </ul>

#### 1.1.1.3 gpy2xx\_soft\_reset

### Description

Resets the PHY to its default state.

Triggers a soft reset. Active links are terminated.

### Prototype

```
int gpy2xx_soft_reset (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

#### 1.1.1.4 gpy2xx\_poll\_reset

### Description

Poll PHY reset status.

The user application uses [gpy2xx\\_soft\\_reset](#) to trigger a soft reset. Afterwards it uses this API to poll every few milliseconds (e.g. every 50 ms) to determine when the reset has completed, i.e. until a positive value is returned.

### Prototype

```
int gpy2xx_poll_reset (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>&gt;0: reset complete</li> <li>=0: reset incomplete</li> <li>&lt;0: error code</li> </ul>

## 1.1.2 MDIO Bus APIs

Group of functional APIs for MDIO Bus access.

**Table 5 MDIO Bus APIs Function Overview**

Name	Description
<a href="#">gpy2xx_serdes_read</a>	Debug API to read SERDES register via XPCS CR access.
<a href="#">gpy2xx_serdes_write</a>	Debug API to write SERDES register via XPCS CR access.
<a href="#">gpy2xx_xpcs_read</a>	Debug API to read XPCS register.
<a href="#">gpy2xx_xpcs_write</a>	Debug API to write XPCS register.

### 1.1.2.1 gpy2xx\_xpcs\_read

#### Description

Debug API to read XPCS register.

#### Prototype

```
int gpy2xx_xpcs_read (
    struct gpy2xx_device * phy,
    uint32_t regaddr,
    uint16_t * data );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
uint32_t	regaddr	XPCS register address (byte address).	-
uint16_t *	data	Pointer to store read data.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.2.2 gpy2xx\_xpcs\_write

#### Description

Debug API to write XPCS register.

#### Prototype

```
int gpy2xx_xpcs_write (
    struct gpy2xx_device * phy,
    uint32_t regaddr,
    uint16_t data );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
uint32_t	regaddr	XPCS register address (byte address).	-
uint16_t	data	Data to be written to given 'regaddr' address.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.2.3 gpy2xx\_serdes\_read

#### Description

Debug API to read SERDES register via XPCS CR access.

## Prototype

```
int gpy2xx_serdes_read (
    struct gpy2xx_device * phy,
    uint32_t addr,
    uint16_t * data );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
uint32_t	addr		-
uint16_t *	data	Pointer to store read data.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.2.4 gpy2xx\_serdes\_write

## Description

Debug API to write SERDES register via XPCS CR access.

## Prototype

```
int gpy2xx_serdes_write (
    struct gpy2xx_device * phy,
    uint32_t addr,
    uint16_t data );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
uint32_t	addr		-
uint16_t	data	Data to be written to given 'regaddr' address.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.3 Link APIs

Group of functional APIs for auto-negotiation, link status, etc.

**Table 6 Link APIs Structure Overview**

Name	Description
<a href="#">gpy2xx_link</a>	Member used by link in gpy2xx_device and gpy2xx_sgmi.

**Table 7 Link APIs Enumerator Overview**

Name	Description
<a href="#">link_mode_bit_indices</a>	Link mode bit indices.

**Table 8 Link APIs Function Overview**

Name	Description
<a href="#">gpy2xx_aneg_done</a>	Gets the restarted auto-negotiation status.
<a href="#">gpy2xx_config_advert</a>	Configures the link advertisement parameters.
<a href="#">gpy2xx_config_aneg</a>	Enable & restart auto-negotiation or set link speed & duplex forcefully depending on given autoneg of gpy2xx_link.
<a href="#">gpy2xx_read_fw_info</a>	Reads and updates the PHY FW parameters and boot status.
<a href="#">gpy2xx_read_status</a>	Reads and updates the link parameters and status.
<a href="#">gpy2xx_restart_aneg</a>	Enable and restart standard auto-negotiation.
<a href="#">gpy2xx_setup_forced</a>	Forcefully sets the link speed and duplex mode.
<a href="#">gpy2xx_update_link</a>	Updates the current link state of UP or DOWN link of gpy2xx_link link state.

#### 1.1.3.1 gpy2xx\_config\_advert

##### Description

Configures the link advertisement parameters.

It writes MII\_ADVERTISE with the appropriate values, after masking the advertising parameter values to make sure that only the parameters supported by the firmware are advertised. The user must call [gpy2xx\\_restart\\_aneg](#) to make sure the effective paramaters will be advertised to the link partner.

##### Prototype

```
int gpy2xx_config_advert (
    struct gpy2xx_device * phy );
```

##### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>&gt;0: advertisement is changed successfully</li> <li>=0: advertisement is not changed</li> <li>&lt;0: error code</li> </ul>

### 1.1.3.2 gpy2xx\_setup\_forced

#### Description

Forcefully sets the link speed and duplex mode.

It configures the PHY to forcefully set the link speed and duplex. This disables the auto-negotiation at this node and the user application must set the link partner's speed and duplex mode forcefully at the other end of the link node.

#### Prototype

```
int gpy2xx_setup_forced (
    struct gpy2xx_device * phy );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.3.3 gpy2xx\_restart\_aneg

#### Description

Enable and restart standard auto-negotiation.

The user application must call this API to restart standard auto-negotiation, and also to make sure the effective parameters are set in [gpy2xx\\_config\\_advert](#). These will be advertised to the link partner when auto-negotiation is restarted.

#### Prototype

```
int gpy2xx_restart_aneg (
    struct gpy2xx_device * phy );
```



## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.3.4 gpy2xx\_config\_aneg

#### Description

Enable & restart auto-negotiation or set link speed & duplex forcefully depending on given autoneg of [gpy2xx\\_link](#).

If autoneg is 1, this API calls [gpy2xx\\_config\\_advert](#) and [gpy2xx\\_restart\\_aneg](#) to advertise and restart auto-negotiation. If autoneg is 0, [gpy2xx\\_setup\\_forced](#) is used to set link speed and duplex forcefully. In this case, the auto-negotiation is disabled and it is the responsibility of the user application to take care of setting the same speed and duplex for the link partner so that the link is UP and running, otherwise the link will be DOWN.

#### Prototype

```
int gpy2xx_config_aneg (
    struct gpy2xx_device * phy );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.3.5 gpy2xx\_aneg\_done

#### Description

Gets the restarted auto-negotiation status.

The user application must call and check to see if auto-negotiation has completed before reading the link status [gpy2xx\\_read\\_status](#) after auto-negotiation was restarted using [gpy2xx\\_restart\\_aneg](#).

### Prototype

```
int gpy2xx_aneg_done (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=1: auto-negotiation is done</li> <li>=0: auto-negotiation is incomplete or there was an error during negotiation</li> <li>&lt;0: error code</li> </ul>

## 1.1.3.6 gpy2xx\_update\_link

### Description

Updates the current link state of UP or DOWN link of [gpy2xx\\_link](#) link state.

The user application can call this API to only update the link state (UP or DOWN) in the variable 'link' of [gpy2xx\\_link](#).

### Prototype

```
int gpy2xx_update_link (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.3.7 gpy2xx\_read\_status

### Description

Reads and updates the link parameters and status.

The user application can call this API to read and update the link parameters of this and the partner node. It also updates the link state of this node. If autoneg is 0, speed and duplex of [gpy2xx\\_link](#) are retrieved from PHY. If autoneg is 1, link partner information is updated in lp\_advertising, speed, duplex, pause, and asym\_pause of [gpy2xx\\_link](#).

### Prototype

```
int gpy2xx_read_status (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.3.8 gpy2xx\_read\_fw\_info

### Description

Reads and updates the PHY FW parameters and boot status.

The user application can call this API to read and update the most recently updated FW parameters, like Firmware major & minor version, Firmware release indication of test vs engineered and the memory target used for firmware execution.

### Prototype

```
int gpy2xx_read_fw_info (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: Firmware is executed from ROM</li> <li>=1: Firmware is executed from OTP (OneTimeProgram memory)</li> <li>=2: Firmware is executed from FLASH</li> <li>=3: Firmware is executed from SRAM (GPY24x only)</li> <li>&lt;0: error code</li> </ul>

### 1.1.4 Supported Link Mode

Group of macros for supported link mode configuration via gpy2xx\_link in Link APIs.

### 1.1.5 Advertised Link Mode

Group of macros for advertised link mode configuration via gpy2xx\_link in Link APIs.

### 1.1.6 Link Speed

Group of macros for link speed configuration via gpy2xx\_link in APIs gpy2xx\_setup\_forced, gpy2xx\_config\_aneg and gpy2xx\_sgmii\_config\_aneg.

### 1.1.7 Duplex

Group of macros for duplex mode configuration via gpy2xx\_link in APIs gpy2xx\_setup\_forced, gpy2xx\_config\_aneg and gpy2xx\_sgmii\_config\_aneg.

### 1.1.8 LED Function Config APIs

Group of functional APIs for LED configuration.

**Table 9 LED Function Config APIs Structure Overview**

Name	Description
<a href="#">gpy2xx_led_cfg</a>	Data structure for LED configuration.

**Table 10 LED Function Config APIs Enumerator Overview**

Name	Description
<a href="#">gpy2xx_led_bsrc</a>	Macros used by slow_blink_src or fast_blink_src or const_on in gpy2xx_led_cfg.
<a href="#">gpy2xx_led_colormode</a>	Macros used by color_mode in gpy2xx_led_cfg.
<a href="#">gpy2xx_led_pulse</a>	Macros used by pulse in gpy2xx_led_cfg.

### 1.1.9 External Interrupt APIs

Group of functional APIs for external interrupt configuration.

**Table 11 External Interrupt APIs Structure Overview**

Name	Description
<a href="#">gpy2xx_phy_extin</a>	Data structure for external interrupt configuration.

Table 12 External Interrupt APIs Enumerator Overview

Name	Description
<a href="#">gpy2xx_extin_im2_mask</a>	Macros used by ext_imask or ext_istat in gpy2xx_phy_extin.
<a href="#">gpy2xx_extin_phy_event</a>	Macros used by std_imask or std_istat in gpy2xx_phy_extin.

Table 13 External Interrupt APIs Function Overview

Name	Description
<a href="#">gpy2xx_extin_get</a>	Gets configured SoC external interrupt source.
<a href="#">gpy2xx_extin_mask</a>	Enable/disable SoC external interrupt event source.

### 1.1.9.1 gpy2xx\_extin\_mask

#### Description

Enable/disable SoC external interrupt event source.

Use this API to config various events to generate MDINT external interrupt to SoC, including PHY specific, PTP and MACSec events.

#### Prototype

```
int gpy2xx_extin_mask (
    struct gpy2xx_device * phy,
    struct gpy2xx_phy_extin * extin );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_phy_extin</a> *	extin	Pointer to EXT interrupt event source ( <a href="#">gpy2xx_phy_extin</a> ).	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.9.2 gpy2xx\_extin\_get

#### Description

Gets configured SoC external interrupt source.

Use this API to get various events set by [gpy2xx\\_extin\\_mask](#) to generate MDINT external interrupt to SoC, including PHY specific, PTP and MACSec events.

## Prototype

```
int gpy2xx_extin_get (
    struct gpy2xx_device * phy,
    struct gpy2xx_phy_extin * extin );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_phy_extin</a> *	extin	Pointer to EXT interrupt event source <a href="#">gpy2xx_phy_extin</a> .	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.10 Diagnosis and Test APIs

Group of functional APIs for diagnosis and test.

**Table 14** Diagnosis and Test APIs Structure Overview

Name	Description
<a href="#">gpy2xx_abist_pair</a>	Member used by pair in gpy2xx_abist_report.
<a href="#">gpy2xx_abist_report</a>	Analog built-in self-test report.
<a href="#">gpy2xx_cdiag_pair</a>	Member used by pair in gpy2xx_cdiag_report.
<a href="#">gpy2xx_cdiag_report</a>	Cable diagnostic report.
<a href="#">gpy2xx_cdiag_sum</a>	Member used by results in gpy2xx_cdiag_pair.
<a href="#">gpy2xx_pcs_status</a>	PCS status.

**Table 15** Diagnosis and Test APIs Enumerator Overview

Name	Description
<a href="#">gpy2xx_abist_test</a>	Macros used by test in gpy2xx_abist_param.
<a href="#">gpy2xx_cdiag_state</a>	Macros used by state in gpy2xx_cdiag_sum.
<a href="#">gpy2xx_errcnt_event</a>	Error events to be counted.
<a href="#">gpy2xx_test_loop</a>	Test loop modes.
<a href="#">gpy2xx_test_mode</a>	Test modes.

**Table 16** Diagnosis and Test APIs Function Overview

Name	Description
<a href="#">gpy2xx_abist_read</a>	Gets analog built-in self-test (ABIST) report.
<a href="#">gpy2xx_abist_start</a>	Starts analog built-in self-test (ABIST)

**Table 16**    **Diagnosis and Test APIs Function Overview** (cont'd)

Name	Description
<a href="#">gpy2xx_cdiag_abist_stop</a>	Stops cable diagnostics or abist test.
<a href="#">gpy2xx_cdiag_read</a>	Gets cable diagnostics test report.
<a href="#">gpy2xx_cdiag_start</a>	Starts cable diagnostics test.
<a href="#">gpy2xx_errcnt_cfg</a>	Configure errors/events to be counted.
<a href="#">gpy2xx_errcnt_read</a>	Reads error/event counters.
<a href="#">gpy2xx_loopback_cfg</a>	Configures various loopback test modes.
<a href="#">gpy2xx_pcs_status_read</a>	Gets PCS status.
<a href="#">gpy2xx_test_mode_cfg</a>	Sets PHY test mode.

### 1.1.10.1 gpy2xx\_test\_mode\_cfg

#### Description

Sets PHY test mode.

Use this API to test modes of PHY, such as transmit waveform, transmit jitter in master or slave mode, transmitter distortion and AFE test.

#### Prototype

```
int gpy2xx_test_mode_cfg (
    struct gpy2xx_device * phy,
    enum gpy2xx_test_mode mode );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
enum <a href="#">gpy2xx_test_mode</a>	mode	PHY test mode ( <a href="#">gpy2xx_test_mode</a> )	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.2 gpy2xx\_cdiag\_start

#### Description

Starts cable diagnostics test.

Use this API to test cable diagnostics, such as cable open/short detection and cable length estimation.

### Prototype

```
int gpy2xx_cdiag_start (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.10.3 gpy2xx\_cdiag\_read

### Description

Gets cable diagnostics test report.

The user application needs to call this API once after it starts CDIAG using [gpy2xx\\_cdiag\\_start](#), to get cable diagnostics such as cable open/short detection and cable length estimation.

### Prototype

```
int gpy2xx_cdiag_read (
    struct gpy2xx_device * phy,
    struct gpy2xx_cdiag_report * report );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_cdiag_report</a> *	report	Pointer to cable diagnostics report ( <a href="#">gpy2xx_cdiag_report</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>



### 1.1.10.4 gpy2xx\_cdiag\_abist\_stop

#### Description

Stops cable diagnostics or abist test.

#### Prototype

```
int gpy2xx_cdiag_abist_stop (
    struct gpy2xx_device * phy );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.5 gpy2xx\_abist\_start

#### Description

Starts analog built-in self-test (ABIST)

The analog BIST is a feature that enables internal testing & qualification of the analog parts of the device, especially the line drivers (LD), analog gain controls (AGC) and ADC/DAC, without the need for expensive analog production testing equipment.

#### Prototype

```
int gpy2xx_abist_start (
    struct gpy2xx_device * phy,
    enum gpy2xx_abist_test test );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
enum <a href="#">gpy2xx_abist_test</a>	test		-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.6 gpy2xx\_abist\_read

#### Description

Gets analog built-in self-test (ABIST) report.

The user application needs to call this API once after it starts ABIST using [gpy2xx\\_abist\\_start](#), in order to get the ABIST report on, for example, the line drivers(LD), analog gain controls (AGC) and ADC/DAC.

#### Prototype

```
int gpy2xx_abist_read (
    struct gpy2xx_device * phy,
    struct gpy2xx_abist_report * report );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_abist_report</a> *	report	Pointer to ABIST report ( <a href="#">gpy2xx_abist_report</a> ). ( <a href="#">gpy2xx_abist_report</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.7 gpy2xx\_loopback\_cfg

#### Description

Configures various loopback test modes.

Most loopback mode settings only take effect after a link down/up event has taken place.

#### Prototype

```
int gpy2xx_loopback_cfg (
    struct gpy2xx_device * phy,
    enum gpy2xx_test_loop tloop );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
enum <a href="#">gpy2xx_test_loop</a>	tloop	PHY Loopback test mode ( <a href="#">gpy2xx_test_loop</a> )	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.8 gpy2xx\_errcnt\_cfg

#### Description

Configure errors/events to be counted.

The error/event is counted using an 8-bit counter. The counter is cleared every time [gpy2xx\\_errcnt\\_read](#) is called. This counter saturates at the value 255 (0xFF).

#### Prototype

```
int gpy2xx_errcnt_cfg (
    struct gpy2xx_device * phy,
    enum gpy2xx_errcnt_event event );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
enum <a href="#">gpy2xx_errcnt_event</a>	event	Error/event to be counted ( <a href="#">gpy2xx_errcnt_event</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.10.9 gpy2xx\_errcnt\_read

#### Description

Reads error/event counters.

The source is configured with [gpy2xx\\_errcnt\\_cfg](#). The error/event is counted using a 8-bit counter. The counter is cleared every time this API is called. This counter saturates at the value 255 (0xFF).

### Prototype

```
int gpy2xx_errcnt_read (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>255: saturated</li> <li>&gt;=0: successful and value is counter</li> <li>&lt;0: error code</li> </ul>

## 1.1.10.10 gpy2xx\_pcs\_status\_read

### Description

Gets PCS status.

Read MMD registers 3.32, 3.33, 3.44, 3.45.

### Prototype

```
int gpy2xx_pcs_status_read (
    struct gpy2xx_device * phy,
    struct gpy2xx_pcs_status * status );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_pcs_status</a> *	status	Point to get status ( <a href="#">gpy2xx_pcs_status</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.11 Synchronous Ethernet (SyncE) Config APIs.

Group of functional APIs for Synchronous Ethernet (SyncE) configuration. Note: Not supported on GPHY flavours (i.e part numbers): GPY212, GPY2XX.

**Table 17 Synchronous Ethernet (SyncE) Config APIs. Structure Overview**

Name	Description
<a href="#">gpy2xx_synce</a>	SyncE configuration.

**Table 18 Synchronous Ethernet (SyncE) Config APIs. Enumerator Overview**

Name	Description
<a href="#">gpy2xx_data_rate</a>	Macros used by data_rate in gpy2xx_synce.
<a href="#">gpy2xx_gpc_sel</a>	Macros used by gpc_sel in gpy2xx_synce and gpy2xx_pps_ctrl.
<a href="#">gpy2xx_synce_clk</a>	Macros used by synce_refclk in gpy2xx_synce.
<a href="#">gpy2xx_synce_master_mode</a>	Macros used by master_sel in gpy2xx_synce.

**Table 19 Synchronous Ethernet (SyncE) Config APIs. Function Overview**

Name	Description
<a href="#">gpy2xx_synce_cfg</a>	Configures SyncE function.

#### 1.1.11.1 gpy2xx\_synce\_cfg

##### Description

Configures SyncE function.

Synchronous Ethernet interface to support transportation of a source-referable clock from the server to the end-points. Essentially, it consist of a reference clock input and a reference clock output. The GPY2xx works as a reference clock master by accepting an input reference clock and transporting this clock frequency via the signalling on the twisted pair interface. As a reference clock slave, the GPY2xx will recover the clock and derive an output reference clock. The GPY2xx provides a means of detecting the loss of reference clock input. For the SoC integrators: there are two possible connections for this interface:

- If the SoC is capable of generating the reference clocks to be transported, then an internal connection is all that is required.
- If the SoC is intended to also accept or provide reference clocks, then pad connectivity to REFCLKO/REFCLKI should be provided.

Note:The GPY2xx provides both REFCLKO and REFCLKI to meet both reference clock master and reference clock slave requirements. SoC integrators may choose, depending on their system needs, to provide either one or both at the chip-level pinning.

##### Prototype

```
int gpy2xx_synce_cfg (
    struct gpy2xx_device * phy,
    struct gpy2xx_synce * cfg );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_synce</a> *	cfg	SyncE configuration ( <a href="#">gpy2xx_synce</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.12 SGMII Interface Config APIs

Group of functional APIs for SGMII interface configuration.

**Table 20 SGMII Interface Config APIs Enumerator Overview**

Name	Description
<a href="#">gpy2xx_sgmi_aneg_mode</a>	Macros used by aneg_mode in gpy2xx_sgmi.
<a href="#">gpy2xx_sgmi_linkcfg_dir</a>	Macros used by linkcfg_dir in gpy2xx_sgmi.
<a href="#">gpy2xx_sgmi_operation</a>	SGMII operation mode.

## 1.1.13 Wake-on-LAN Flags

Group of Wake-on-LAN flags.

## 1.1.14 Miscellaneous Config APIs

Group of functional APIs for miscellaneous features.

**Table 21 Miscellaneous Config APIs Structure Overview**

Name	Description
<a href="#">gpy2xx_ads_ctrl</a>	Link Speed Auto-downspeed Control.
<a href="#">gpy2xx_ads_sta</a>	Auto-downspeed status & config.
<a href="#">gpy2xx_pvt</a>	GPHY temperature information.
<a href="#">gpy2xx_wolinfo</a>	Wake-on-LAN configuration.

**Table 22 Miscellaneous Config APIs Enumerator Overview**

Name	Description
<a href="#">ads_adv_status</a>	Macros used by no_nrg_rst in gpy2xx_ads_ctrl.
<a href="#">ads_force_rst_status</a>	Macros used by force_rst in gpy2xx_ads_ctrl.
<a href="#">ads_nbt_ds_status</a>	Macros used by downshift_en in gpy2xx_ads_ctrl.

Table 23 Miscellaneous Config APIs Function Overview

Name	Description
<a href="#">gpy2xx_ads_cfg</a>	Configures auto-downspeed (ADS) function.
<a href="#">gpy2xx_ads_detected</a>	Gets auto-downspeed (ADS) detected status.
<a href="#">gpy2xx_pvt_get</a>	Gets the GPY2xx's sensor temperature in degrees Celsius.
<a href="#">gpy2xx_wol_cfg</a>	Configures Wake-on-LAN function.

### 1.1.14.1 gpy2xx\_wol\_cfg

#### Description

Configures Wake-on-LAN function.

Wake-on-LAN (WoL) is a feature that is capable of monitoring and detecting WoL packets. The PHY issues a wake-up indication, via the external interrupt sourced by the GPY2xx to the SoC, by activating the MDINT signal to wake a larger SoC from its power-down state. The most commonly used WoL packet is a magic packet that contains the MAC address of the device that is to be woken up. The wolopts of [gpy2xx\\_wolinfo](#) will be updated to the options that have been successfully configured into the hardware after API called.

#### Prototype

```
int gpy2xx_wol_cfg (
    struct gpy2xx_device * phy,
    struct gpy2xx_wolinfo * wol );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_wolinfo</a> *	wol	Pointer to Wake-On-Lan configuration ( <a href="#">gpy2xx_wolinfo</a> ).	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.14.2 gpy2xx\_ads\_cfg

#### Description

Configures auto-downspeed (ADS) function.

The ADS feature ensures maximum interoperability in specific situations, such as, information available about the cabling during ANEG is insufficient, the integrity of received signals is not suitable for link-up due to increased alien noise, or over-length cables. The ADS feature avoids continuous link-up failures in such situations, and the next link-up will be done at the next advertised speed below 1000 Mbit/s.

If the link only supports speeds of 1000 Mbit/s or 2500 Mbit/s, the ADS feature is automatically disabled.

## Prototype

```
int gpy2xx_ads_cfg (
    struct gpy2xx_device * phy,
    struct gpy2xx_ads_ctrl * ads );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_ads_ctrl</a> *	ads	Holds auto-downspeed configuration ( <a href="#">gpy2xx_ads_sta</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.14.3 gpy2xx\_ads\_detected

## Description

Gets auto-downspeed (ADS) detected status.

Gets whether the ADS has happened at all due to harsh or inadequate cable infrastructure environments. The number of times the GPHY2xx decided to downspeed the link is counted and available as statistics via the [gpy2xx\\_errcnt\\_cfg](#) for event = 9.

## Prototype

```
int gpy2xx_ads_detected (
    struct gpy2xx_device * phy );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>&gt;0: successful and auto-downspeed is detected</li> <li>=0: successful but no auto-downspeed is detected</li> <li>&lt;0: error code</li> </ul>



#### 1.1.14.4 gpy2xx\_pvt\_get

##### Description

Gets the GPY2xx's sensor temperature in degrees Celsius.

##### Prototype

```
int gpy2xx_pvt_get (
    struct gpy2xx_device * phy,
    struct gpy2xx_pvt * pvt );
```

##### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_pvt</a> *	pvt	Pointer to on-chip sensor temperature ( <a href="#">gpy2xx_pvt</a> ).	-

##### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

#### 1.1.15 Firmware Download APIs

Group of functional APIs to download firmware into flash memory.

#### 1.1.16 USXGMII\_REACH APIs

Group of functional APIs for USXGMII Reach configuration. Note: USXGMII is not supported for models without USXGMII capability.

**Table 24 USXGMII\_REACH APIs Structure Overview**

Name	Description
<a href="#">gpy2xx_4peye_cfg</a>	USXGMII 4-point eye test parameter. Used by <a href="#">gpy2xx_usxgmii_4peye_start</a> and <a href="#">gpy2xx_usxgmii_4peye_cfg_get</a> .
<a href="#">gpy2xx_4peye_result</a>	USXGMII 4-point eye test result. Used by <a href="#">gpy2xx_usxgmii_4peye_result</a> .

**Table 25 USXGMII\_REACH APIs Function Overview**

Name	Description
<a href="#">gpy2xx_usxgmii_4peye_cancel</a>	Poll running state of 4-point eye test on USXGMII interface.
<a href="#">gpy2xx_usxgmii_4peye_cfg_get</a>	Start 4-point eye test on USXGMII interface.
<a href="#">gpy2xx_usxgmii_4peye_poll</a>	Poll running state of 4-point eye test on USXGMII interface.
<a href="#">gpy2xx_usxgmii_4peye_result</a>	Start 4-point eye test on USXGMII interface.

Table 25 USXGMII\_REACH APIs Function Overview (cont'd)

Name	Description
<a href="#">gpy2xx_usxgmii_4peye_start</a>	Start 4-point eye test on USXGMII interface.
<a href="#">gpy2xx_usxgmii_dbg_enter</a>	Enter or exit USXGMII debug mode.

### 1.1.16.1 gpy2xx\_usxgmii\_dbg\_enter

#### Description

Enter or exit USXGMII debug mode.

This is only applied to models supporting USXGMII EQ tuning function. When entering debug mode, GPY2xx device prepares environment for EQ tuning. This must be first step before running any USXGMII (XPCS/SERDES) register access and related test functions/APIs.

#### Prototype

```
int gpy2xx_usxgmii_dbg_enter (
    struct gpy2xx_device * phy,
    bool is_exit );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
bool	is_exit	Value 'false' to 'enter' and 'true' to leave.	-

#### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.16.2 gpy2xx\_usxgmii\_4peye\_start

#### Description

Start 4-point eye test on USXGMII interface.

This is only applied to models supporting USXGMII EQ tuning function. Start to run 4-point eye test on USXGMII with given parameters. [gpy2xx\\_usxgmii\\_dbg\\_enter](#) must be called to enter debug mode before this API. 4-point eye test takes long time for higher BER target. Use [gpy2xx\\_usxgmii\\_4peye\\_poll](#) to check whether the test is completed. Use [gpy2xx\\_usxgmii\\_4peye\\_cancel](#) to cancel the test if it's too long to wait for. Use [gpy2xx\\_usxgmii\\_4peye\\_result](#) to retrieve the test result.

#### Prototype

```
int gpy2xx_usxgmii_4peye_start (
    struct gpy2xx_device * phy,
```

```
const struct gpy2xx_4peye_cfg * pcfg );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
const struct <a href="#">gpy2xx_4peye_cfg</a> *	pcfg	Refer to <a href="#">gpy2xx_4peye_cfg</a> for parameter details.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.16.3 gpy2xx\_usxgmii\_4peye\_poll

#### Description

Poll running state of 4-point eye test on USXGMII interface.

This is only applied to models supporting USXGMII EQ tuning function. [gpy2xx\\_usxgmii\\_dbg\\_enter](#) must be called to enter debug mode before this API. [gpy2xx\\_usxgmii\\_4peye\\_start](#) is used to start 4-point eye test and this API is used to check whether the test is completed.

#### Prototype

```
int gpy2xx_usxgmii_4peye_poll (
    struct gpy2xx_device * phy );
```

## Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=1: running</li> <li>=0: free (test completed or test not started)</li> <li>&lt;0: error code</li> </ul>

### 1.1.16.4 gpy2xx\_usxgmii\_4peye\_cancel

#### Description

Poll running state of 4-point eye test on USXGMII interface.

This is only applied to models supporting USXGMII EQ tuning function. [gpy2xx\\_usxgmii\\_dbg\\_enter](#) must be called to enter debug mode before this API. This API is used to stop running test started with [gpy2xx\\_usxgmii\\_4peye\\_start](#).

### Prototype

```
int gpy2xx_usxgmii_4peye_cancel (
    struct gpy2xx_device * phy );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-

### Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.1.16.5 gpy2xx\_usxgmii\_4peye\_result

### Description

Start 4-point eye test on USXGMII interface.

This is only applied to models supporting USXGMII EQ tuning function. Start to run 4-point eye test on USXGMII with given parameters. This API is used to retrieve 4-point eye test result.

### Prototype

```
int gpy2xx_usxgmii_4peye_result (
    struct gpy2xx_device * phy,
    struct gpy2xx_4peye_result * presult );
```

### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_4peye_result</a> *	presult	Refer to <a href="#">gpy2xx_4peye_result</a> for test result.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

### 1.1.16.6 gpy2xx\_usxgmii\_4peye\_cfg\_get

#### Description

Start 4-point eye test on USXGMII interface.

This is only applied to models supporting USXGMII EQ tuning function. This API is used to retrieve parameters used to start 4-point eye test with [gpy2xx\\_usxgmii\\_4peye\\_start](#).

#### Prototype

```
int gpy2xx_usxgmii_4peye_cfg_get (
    struct gpy2xx_device * phy,
    struct gpy2xx_4peye_cfg * pcfg );
```

#### Parameters

Data Type	Name	Description	Dir
struct <a href="#">gpy2xx_device</a> *	phy	Pointer to GPHY data ( <a href="#">gpy2xx_device</a> ).	-
struct <a href="#">gpy2xx_4peye_cfg</a> *	pcfg	Refer to <a href="#">gpy2xx_4peye_cfg</a> for parameter details.	-

## Return Values

Data Type	Description
int	<ul style="list-style-type: none"> <li>=0: successful</li> <li>&lt;0: error code</li> </ul>

## 1.2 GSWIP APIs related Data Structs and Enums Type defines

This chapter describes the GSWIP's GSW APIs related Data structures and enums.

**Table 26 GSWIP APIs related Data Structs and Enums Type defines Structure Overview**

Name	Description
<a href="#">GSW_B_TAG_Config_t</a>	B-TAG header definition .GSWIP-3.2 only Used by GSW_PBB_Tunnel_Template_Config_t.
<a href="#">GSW_BRIDGE_alloc_t</a>	Bridge Allocation. Used by GSW_BRIDGE_ALLOC and GSW_BRIDGE_FREE.

Table 26 GSWIP APIs related Data Structs and Enums Type defines Structure Overview (cont'd)

Name	Description
<a href="#">GSW_BRIDGE_config_t</a>	Bridge Configuration. Used by GSW_BRIDGE_CONFIG_SET and GSW_BRIDGE_CONFIG_GET.
<a href="#">GSW_BRIDGE_portAlloc_t</a>	Bridge Port Allocation. Used by GSW_BRIDGE_PORT_ALLOC and GSW_BRIDGE_PORT_FREE.
<a href="#">GSW_BRIDGE_portConfig_t</a>	Bridge Port Configuration. Used by GSW_BRIDGE_PORT_CONFIG_SET and GSW_BRIDGE_PORT_CONFIG_GET.
<a href="#">GSW_cfg_t</a>	Global Switch configuration Attributes. Used by GSW_CFG_SET and GSW_CFG_GET.
<a href="#">GSW_CPU_PortCfg_t</a>	Defines one port that is directly connected to the CPU and its applicable settings. Used by GSW_CPU_PORT_CFG_SET and GSW_CPU_PORT_CFG_GET.
<a href="#">GSW_debug_t</a>	For debugging Purpose only. Used for GSWIP 3.3.
<a href="#">GSW_DSCP2PCP_map_t</a>	DSCP to PCP Mapping. Used by GSW_DSCP2PCP_MAP_GET.
<a href="#">GSW_I_TAG_Config_t</a>	I-TAG header definition .GSWIP-3.2 only Used by GSW_PBB_Tunnel_Template_Config_t.
<a href="#">GSW_MAC_tableAdd_t</a>	MAC Table Entry to be added. Used by GSW_MAC_TABLE_ENTRY_ADD.
<a href="#">GSW_MAC_tableClearCond_t</a>	MAC Table Clear based on given condition. Used by GSW_MAC_TABLE_CLEAR_COND.
<a href="#">GSW_MAC_tableQuery_t</a>	Search for a MAC address entry in the address table. Used by GSW_MAC_TABLE_ENTRY_QUERY.
<a href="#">GSW_MAC_tableRead_t</a>	MAC Table Entry to be read. Used by GSW_MAC_TABLE_ENTRY_READ.
<a href="#">GSW_MAC_tableRemove_t</a>	MAC Table Entry to be removed. Used by GSW_MAC_TABLE_ENTRY_REMOVE.
<a href="#">GSW_MACFILTER_default_t</a>	Default MAC Address Filter. Used by GSW_DEFAULT_MAC_FILTER_SET and GSW_DEFAULT_MAC_FILTER_GET.
<a href="#">GSW_monitorPortCfg_t</a>	Port monitor configuration. Used by GSW_MONITOR_PORT_CFG_GET and GSW_MONITOR_PORT_CFG_SET.
<a href="#">GSW_multicastRouter_t</a>	Add an Ethernet port as router port to the switch hardware multicast table. Used by GSW_MULTICAST_ROUTER_PORT_ADD and GSW_MULTICAST_ROUTER_PORT_REMOVE.
<a href="#">GSW_multicastRouterRead_t</a>	Check if a port has been selected as a router port. Used by GSW_MULTICAST_ROUTER_PORT_READ. Not applicable to GSWIP-3.1.
<a href="#">GSW_multicastSnoopCfg_t</a>	Configure the switch multicast configuration. Used by GSW_MULTICAST_SNOOP_CFG_SET and GSW_MULTICAST_SNOOP_CFG_GET.
<a href="#">GSW_multicastTable_t</a>	Add a host as a member to a multicast group. Used by GSW_MULTICAST_TABLE_ENTRY_ADD and GSW_MULTICAST_TABLE_ENTRY_REMOVE.

Table 26 GSWIP APIs related Data Structs and Enums Type defines Structure Overview (cont'd)

Name	Description
<a href="#">GSW_multicastTableRead_t</a>	Read out the multicast membership table. Used by GSW_MULTICAST_TABLE_ENTRY_READ.
<a href="#">GSW_PBB_Tunnel_Template_Config_t</a>	Tunnel Template Configuration. GSWIP-3.2 only Used by GSW_PBB_TunnelTemplate_Config_Set and GSW_PBB_TunnelTemplate_Config_Get For GSW_PBB_TunnelTemplate_Free, this field should be valid ID returned by GSW_PBB_TunnelTemplate_Alloc.
<a href="#">GSW_PMAPPER_t</a>	P-mapper Configuration Used by GSW_CTP_portConfig_t, GSW_BRIDGE_portConfig_t. In case of LAG, it is user's responsibility to provide the mapped entries in given P-mapper table. In other modes the entries are auto mapped from input packet.
<a href="#">GSW_portCfg_t</a>	Port Configuration Parameters. Used by GSW_PORT_CFG_GET and GSW_PORT_CFG_SET.
<a href="#">GSW_portLinkCfg_t</a>	Ethernet port link, speed status and flow control status. Used by GSW_PORT_LINK_CFG_GET and GSW_PORT_LINK_CFG_SET.
<a href="#">GSW_QoS_ClassPCP_Cfg_t</a>	Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW_QOS_CLASS_PCP_SET and GSW_QOS_CLASS_PCP_GET.
<a href="#">GSW_QoS_colorMarkingEntry_t</a>	Color Marking Table. There are standards to define the marking table. User should use GSW_QOS_COLOR_MARKING_TABLE_SET to initialize the table before color marking happens. GSW_QOS_COLOR_MARKING_TABLE_GET is used to get the marking table, mainly for debug purpose.
<a href="#">GSW_QoS_colorRemarkingEntry_t</a>	Color Remarking Table. There are standards to define the remarking table. User should use GSW_QOS_COLOR_REMARKING_TABLE_SET to initialize the table before color remarking happens. GSW_QOS_COLOR_REMARKING_TABLE_GET is used to get the remarking table, mainly for debug purpose.
<a href="#">GSW_QoS_DSCP_ClassCfg_t</a>	DSCP mapping table. Used by GSW_QOS_DSCP_CLASS_SET and GSW_QOS_DSCP_CLASS_GET.
<a href="#">GSW_QoS_DSCP_DropPrecedenceCfg_t</a>	DSCP to Drop Precedence assignment table configuration. Used by GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_SET and GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_GET.
<a href="#">GSW_QoS_FlowCtrlCfg_t</a>	Configures the global buffer flow control threshold for conforming and non-conforming packets. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_FLOWCTRL_CFG_SET and GSW_QOS_FLOWCTRL_CFG_GET.

Table 26 GSWIP APIs related Data Structs and Enums Type defines Structure Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_FlowCtrlPortCfg_t</a>	Configures the ingress port flow control threshold for used packet segments. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_FLOWCTRL_PORT_CFG_SET and GSW_QOS_FLOWCTRL_PORT_CFG_GET.
<a href="#">GSW_QoS_meterCfg_t</a>	Configures the parameters of a rate meter instance. Used by GSW_QOS_METER_ALLOC, GSW_QOS_METER_FREE, GSW_QOS_METER_CFG_SET and GSW_QOS_METER_CFG_GET.
<a href="#">GSW_QoS_PCP_ClassCfg_t</a>	Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW_QOS_PCP_CLASS_SET and GSW_QOS_PCP_CLASS_GET.
<a href="#">GSW_QoS_portCfg_t</a>	Describes which priority information of ingress packets is used (taken into account) to identify the packet priority and the related egress priority queue. For DSCP, the priority to queue assignment is done using GSW_QOS_DSCP_CLASS_SET. For VLAN, the priority to queue assignment is done using GSW_QOS_PCP_CLASS_SET. Used by GSW_QOS_PORT_CFG_SET and GSW_QOS_PORT_CFG_GET.
<a href="#">GSW_QoS_portRemarkingCfg_t</a>	Port Remarking Configuration. Ingress and Egress remarking options for dedicated packet fields DSCP, CTAG VLAN PCP, STAG VLAN PCP and STAG VLAN DEI. Remarking is done either on the used traffic class or the drop precedence. Packet field specific remarking only applies on a packet if enabled on ingress and egress port. Used by GSW_QOS_PORT_REMARKING_CFG_SET and GSW_QOS_PORT_REMARKING_CFG_GET.
<a href="#">GSW_QoS_QueueBufferReserveCfg_t</a>	Reserved egress queue buffer segments. Used by GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_SET and GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_GET.
<a href="#">GSW_QoS_queuePort_t</a>	Sets the Queue ID for one traffic class of one port. Used by GSW_QOS_QUEUE_PORT_SET and GSW_QOS_QUEUE_PORT_GET.
<a href="#">GSW_QoS_schedulerCfg_t</a>	Configures the egress queues attached to a single port, and that are scheduled to transmit the queued Ethernet packets. Used by GSW_QOS_SCHEDULER_CFG_SET and GSW_QOS_SCHEDULER_CFG_GET.
<a href="#">GSW_QoS_ShaperCfg_t</a>	Configures a rate shaper instance with the rate and the burst size. Used by GSW_QOS_SHAPER_CFG_SET and GSW_QOS_SHAPER_CFG_GET.
<a href="#">GSW_QoS_ShaperQueue_t</a>	Assign one rate shaper instance to a QoS queue. Used by GSW_QOS_SHAPER_QUEUE_ASSIGN and GSW_QOS_SHAPER_QUEUE_DEASSIGN.



Table 26 GSWIP APIs related Data Structs and Enums Type defines Structure Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_ShaperQueueGet_t</a>	Retrieve if a rate shaper instance is assigned to a QoS egress queue. Used by GSW_QOS_SHAPER_QUEUE_GET.
<a href="#">GSW_QoS_stormCfg_t</a>	Assigns one meter instances for storm control. Used by GSW_QOS_STORM_CFG_SET and GSW_QOS_STORM_CFG_GET. Not applicable to GSWIP-3.1.
<a href="#">GSW_QoS_SVLAN_PCP_ClassCfg_t</a>	Traffic class associated with a particular STAG VLAN 802.1P (PCP) priority and Drop Eligible Indicator (DEI) mapping value. This table is global for the entire switch device. Priority map entry structure. The table index value is calculated by 'index=PCP + 8*DEI' Used by GSW_QOS_SVLAN_PCP_CLASS_SET and GSW_QOS_SVLAN_PCP_CLASS_GET.
<a href="#">GSW_QoS_WRED_Cfg_t</a>	Configures the global probability profile of the device. The min. and max. threshold values are given in number of packet buffer segments and required only in case of Manual Mode. The GSWIP-3.0/3.1 supports Auto mode and the threshold values are dynamically computed internally by GSWIP. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_CFG_SET and GSW_QOS_WRED_CFG_GET.
<a href="#">GSW_QoS_WRED_PortCfg_t</a>	Configures the WRED threshold parameter per port. The configured thresholds apply to fill level sum of all egress queues which are assigned to the egress port. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_PORT_CFG_SET and GSW_QOS_WRED_PORT_CFG_GET.
<a href="#">GSW_QoS_WRED_QueueCfg_t</a>	Configures the WRED threshold level values. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_QUEUE_CFG_SET and GSW_QOS_WRED_QUEUE_CFG_GET.
<a href="#">GSW_STP_BPDU_Rule_t</a>	Spanning tree packet detection and forwarding. Used by GSW_STP_BPDU_RULE_SET and GSW_STP_BPDU_RULE_GET.
<a href="#">GSW_STP_portCfg_t</a>	Configures the Spanning Tree Protocol state of an Ethernet port. Used by GSW_STP_PORT_CFG_SET and GSW_STP_PORT_CFG_GET.
<a href="#">GSW_trunkingCfg_t</a>	Global Ethernet trunking configuration. Used by GSW_TRUNKING_CFG_GET and GSW_TRUNKING_CFG_SET.

Table 27 GSWIP APIs related Data Structs and Enums Type defines Enumerator Overview

Name	Description
<a href="#">GSW_8021X_portState_t</a>	Describes the 802.1x port state. Used by GSW_8021X_portCfg_t.
<a href="#">GSW_ageTimer_t</a>	Aging Timer Value. Used by GSW_cfg_t.
<a href="#">GSW_BridgeConfigMask_t</a>	Bridge configuration mask. Used by GSW_BRIDGE_config_t.

Table 27 GSWIP APIs related Data Structs and Enums Type defines Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_BridgeForwardMode_t</a>	Bridge forwarding type of packet. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_BridgePortConfigMask_t</a>	Bridge Port configuration mask. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_BridgePortEgressMeter_t</a>	Meters for various egress traffic type. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_clkMode_t</a>	Ethernet port clock source configuration. Used by GSW_portLinkCfg_t.
<a href="#">GSW_ColorMarkingMode_t</a>	Color Marking Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_ColorRemarkingMode_t</a>	Color Remarking Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	Parser Flags and Offsets Header settings on CPU Port for GSWIP-3.0. Used by GSW_CPU_PortCfg_t.
<a href="#">GSW_CPU_SpecialTagEthType_t</a>	Special tag Ethertype mode.
<a href="#">GSW_direction_t</a>	Specifies the direction for ingress and egress. Used by GSW_QoS_meterPort_t and GSW_QoS_meterPortGet_t.
<a href="#">GSW_FCS_TxOps_t</a>	FCS and Pad Insertion operations for GSWIP 3.1 Used by GSW_CPU_PortCfgSet/Get.
<a href="#">GSW_If_RMON_Mode_t</a>	Interface RMON Counter Mode - (FID, SUBID or FLOWID) Config - GSWIP-3.0 only. Used by GSW_portCfg_t.
<a href="#">GSW_IGMP_MemberMode_t</a>	Defines the multicast group member mode. Used by GSW_multicastTable_t and GSW_multicastTableRead_t.
<a href="#">GSW_MacClearType_t</a>	MAC Table Clear Type Used by GSW_MAC_tableClearCond_t.
<a href="#">GSW_MacFilterType_t</a>	MAC Address Filter Type. Used by GSW_MACFILTER_default_t.
<a href="#">GSW_MII_Mode_t</a>	Ethernet port interface mode. A port might support only a subset of the possible settings. Used by GSW_portLinkCfg_t.
<a href="#">GSW_MII_Type_t</a>	Ethernet port configuration for PHY or MAC mode. Used by GSW_portLinkCfg_t.
<a href="#">GSW_multicastReportSuppression_t</a>	Configure the IGMP report suppression mode. Used by GSW_multicastSnoopCfg_t.
<a href="#">GSW_multicastSnoopMode_t</a>	Configure the IGMP snooping mode. Used by GSW_multicastSnoopCfg_t.
<a href="#">GSW_PmapperMappingMode_t</a>	P-mapper Mapping Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_portDuplex_t</a>	Ethernet port duplex status. Used by GSW_portLinkCfg_t.
<a href="#">GSW_portEnable_t</a>	Port Enable Type Selection. Used by GSW_portCfg_t.
<a href="#">GSW_portFlow_t</a>	Ethernet flow control status. Used by GSW_portCfg_t.
<a href="#">GSW_portForward_t</a>	Packet forwarding. Used by GSW_STP_BPDU_Rule_t and GSW_multicastSnoopCfg_t and GSW_8021X_EAPOL_Rule_t.
<a href="#">GSW_portLink_t</a>	Force the MAC and PHY link modus. Used by GSW_portLinkCfg_t.
<a href="#">GSW_portMonitor_t</a>	Port Mirror Options. Used by GSW_portCfg_t.
<a href="#">GSW_portSpeed_t</a>	Ethernet port speed mode. For certain generations of GSWIP, a port might support only a subset of the possible settings. Used by GSW_portLinkCfg_t.

Table 27 GSWIP APIs related Data Structs and Enums Type defines Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_ClassSelect_t</a>	Selection of the traffic class field. Used by GSW_QoS_portCfg_t. The port default traffic class is assigned in case non of the configured protocol code points given by the packet.
<a href="#">GSW_QoS_DropPrecedence_t</a>	DSCP Drop Precedence to color code assignment. Used by GSW_QoS_DSCP_DropPrecedenceCfg_t.
<a href="#">GSW_QoS_ingressRemarking_t</a>	Ingress DSCP remarking attribute. This attribute defines on the ingress port packets how these will be remarked on the egress port. A packet is only remarked in case its ingress and its egress port have remarking enabled. Used by GSW_QoS_portRemarkingCfg_t.
<a href="#">GSW_QoS_Meter_Type</a>	Meter Type - srTCM or trTCM. Defines the Metering algorithm Type. Used by GSW_QoS_meterCfg_t.
<a href="#">GSW_QoS_qMapMode_t</a>	Describes the QoS Queue Mapping Mode. GSWIP-3.1 only. Used by GSW_QoS_queuePort_t.
<a href="#">GSW_QoS_Scheduler_t</a>	Select the type of the Egress Queue Scheduler. Used by GSW_QoS_schedulerCfg_t.
<a href="#">GSW_QoS_WRED_Mode_t</a>	WRED Cfg Type - Automatic (Adaptive) or Manual. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_Profile_t</a>	Drop Probability Profile. Defines the drop probability profile. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_ThreshMode_t</a>	WRED Thresholds Mode Type. - GSWIP-3.0/3.1 only Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_WATERMARK_t</a>	Egress Queue Congestion Notification Watermark. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_return_t</a>	Enumeration for function status return. The upper four bits are reserved for error classification.
<a href="#">GSW_STP_PortState_t</a>	Spanning Tree Protocol port states. Used by GSW_STP_portCfg_t.
<a href="#">GSW_VlanFilterTciMask_t</a>	VLAN Filter TCI Mask. Used by GSW_VLANFILTER_config_t.

### 1.3 GSWIP CPT APIs related Data Structs and Enums Type defines

This chapter describes the interface for configuring the MXL GSW GPT Port.

Table 28 GSWIP CPT APIs related Data Structs and Enums Type defines Structure Overview

Name	Description
<a href="#">GSW_CTP_portAssignment_t</a>	CTP Port Assignment/association with logical port. Used by GSW_CTP_PORT_ASSIGNMENT_ALLOC, GSW_CTP_PORT_ASSIGNMENT_SET and GSW_CTP_PORT_ASSIGNMENT_GET.
<a href="#">GSW_CTP_portConfig_t</a>	CTP Port Configuration. Used by GSW_CTP_PORT_CONFIG_SET and GSW_CTP_PORT_CONFIG_GET.

Table 29 GSWIP CPT APIs related Data Structs and Enums Type defines Enumerator Overview

Name	Description
<a href="#">GSW_CtpPortConfigMask_t</a>	CTP Port configuration mask. Used by GSW_CTP_portConfig_t.
<a href="#">GSW_LogicalPortMode_t</a>	Logical port mode. Used by GSW_CTP_portAssignment_t.

## 1.4 GSWIP Packet Classification Engine (PCE) related APIs

This chapter describes the GSWIP's PCE and other low level related APIs.

Table 30 GSWIP Packet Classification Engine (PCE) related APIs Structure Overview

Name	Description
<a href="#">GSW_PCE_pattern_t</a>	Packet Classification Engine Pattern Configuration. GSWIP-3.0 has additional patterns such as Inner IP, Inner DSCP, Inner Protocol, Exclude Mode etc. Used by GSW_PCE_rule_t.
<a href="#">GSW_PCE_rule_t</a>	Parameter to add/read a rule to/from the packet classification engine. Used by GSW_PCE_RULE_WRITE and GSW_PCE_RULE_READ.
<a href="#">GSW_PCE_ruleDelete_t</a>	Parameter to delete a rule from the packet classification engine. Used by GSW_PCE_RULE_DELETE.
<a href="#">GSW_register_mod_t</a>	Register access parameter to directly modify internal registers. Used by GSW_REGISTER_MOD.
<a href="#">GSW_register_t</a>	Register access parameter to directly read or write switch internal registers. Used by GSW_REGISTER_SET and GSW_REGISTER_GET.

Table 31 GSWIP Packet Classification Engine (PCE) related APIs Enumerator Overview

Name	Description
<a href="#">GSW_PCE_ActionColorFrame_t</a>	Color Frame Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCriticalFrame_t</a>	Critical Frame Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCrossState_t</a>	Cross State Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCrossVLAN_t</a>	Cross VLAN Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionIGMP_Snoop_t</a>	IGMP Snooping Control. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionIrq_t</a>	Interrupt Control Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionLearning_t</a>	MAC Address Learning control. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionMeter_t</a>	Flow Meter Assignment control. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionPortmap_t</a>	Forwarding Group Action Selector. This flow table action and the 'bFlowID_Action' action can be used exclusively. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionTimestamp_t</a>	Timestamp Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionTrafficClass_t</a>	Traffic Class Action Selector. Used by GSW_PCE_action_t.

Table 31 GSWIP Packet Classification Engine (PCE) related APIs Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_PCE_ActionVLAN_t</a>	VLAN Group Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_IP_t</a>	Rule selection for IPv4/IPv6. Used by GSW_PCE_pattern_t.
<a href="#">GSW_PCE_PortFilterAction_t</a>	Port Filter Action-1/2/3/4/5/6 Selector - used for GSWIP-3.0 only. This can be used only along with PortMember config. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ProcessingPathAction_t</a>	MPE Processing Path Assignment Selector - used for GSWIP-3.0 only. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_RuleRegion_t</a>	Traffic Flow Table Mangement. Used by GSW_PCE_rule_t.
<a href="#">GSW_PCE_SUBIFID_TYPE_t</a>	Select Mode of Sub-Interface ID Field. Used by GSW_PCE_pattern_t.

## 1.5 GSWIP PMAC related APIs

This chapter describes the GSWIP's PMAC and other low level related APIs.

Table 32 GSWIP PMAC related APIs Structure Overview

Name	Description
<a href="#">GSW_PMAC_BM_Cfg_t</a>	Configure the Backpressure mapping for egress Queues Congestion or ingress (receiving) ports to DMA channel. Used by GSW_PMAC_BM_CFG_SET and GSW_PMAC_BM_CFG_GET.
<a href="#">GSW_PMAC_Cnt_t</a>	PMAC Counters available for specified DMA Channel. Used by GSW_PMAC_COUNT_GET.
<a href="#">GSW_PMAC_Eg_Cfg_t</a>	Configure the PMAC Egress Configuration. (Upto 1024 entries) This Egress PMAC table is addressed through combination of following fields (Bit0 - Bit 9). nDestPortId (Bits 0-3) + Combination of [bMpe1Flag (Bit 4) + bMpe2Flag (Bit 5) + bEncFlag (Bit 6) + bDecFlag (Bit 7) ] or TrafficClass Value (Bits 4-7) + nFlowIdMSB (Bits 8-9). The bits 4-7 of index option is either based upon TC (default) or combination of Processing flags is decided through bProcFlagsEgPMACena. It is expected to pass the correct value in bProcFlagsSelect same as global bProcFlagsEgPMACena; Used by GSW_PMAC_EG_CFG_SET and GSW_PMAC_EG_CFG_GET.
<a href="#">GSW_PMAC_Glbl_Cfg_t</a>	Configure the global settings of PMAC for GSWIP-3.x. This includes settings such as Jumbo frame, Checksum handling, Padding and Egress PMAC Selector Config. Used by GSW_PMAC_GLBL_CFG_SET and GSW_PMAC_GLBL_CFG_GET.
<a href="#">GSW_PMAC_Ig_Cfg_t</a>	Configure the PMAC Ingress Configuration on a given Tx DMA channel to PMAC. (Upto 16 entries). This Ingress PMAC table is addressed through Transmit DMA Channel Identifier. Used by GSW_PMAC_IG_CFG_SET and GSW_PMAC_IG_CFG_GET.

Table 33 GSWIP PMAC related APIs Enumerator Overview

Name	Description
<a href="#">GSW_PMAC_Ig_Cfg_Src_t</a>	PMAC Ingress Configuration Source Source of the corresponding field.
<a href="#">GSW_PMAC_Proc_Flags_Eg_Cfg_t</a>	Egress PMAC Config Table Selector.
<a href="#">GSW_PMAC_Short_Frame_Chk_t</a>	Short Length Received Frame Check Type for PMAC. Used by PMAC structure GSW_PMAC_Glbl_Cfg_t.

## 1.6 GSWIP RMON Counter related APIs

This chapter describes the GSWIP's RMON Counter and other low level related APIs.

Table 34 GSWIP RMON Counter related APIs Structure Overview

Name	Description
<a href="#">GSW_Debug_RMON_Port_cnt_t</a>	For debugging Purpose only. Used for GSWIP 3.1.
<a href="#">GSW_RMON_clear_t</a>	RMON Counters Data Structure for clearance of values. Used by GSW_RMON_CLEAR.
<a href="#">GSW_RMON_flowGet_t</a>	Hardware platform extended RMON Counters. GSWIP-3.1 only. This structure contains additional RMON counters. These counters can be used by the packet classification engine and can be freely assigned to dedicated packet rules and flows. Used by GSW_RMON_FLOW_GET.
<a href="#">GSW_RMON_Meter_cnt_t</a>	RMON Counters for Meter - Type (GSWIP-3.0 only). This structure contains the RMON counters of one Meter Instance. Used by GSW_RMON_METER_GET.
<a href="#">GSW_RMON_mode_t</a>	RMON Counters Mode for different Elements. This structure takes RMON Counter Element Name and mode config.
<a href="#">GSW_RMON_Port_cnt_t</a>	RMON Counters for individual Port. This structure contains the RMON counters of an Ethernet Switch Port. Used by GSW_RMON_PORT_GET.
<a href="#">GSW_TflowCmodeConf_t</a>	Hardware platform TFLOW counter mode. Supported modes include, Global (default), Logical, CTP, Bridge port mode. The number of counters that can be assigned varies based these mode type. Used by GSW_TFLOW_COUNT_MODE_SET and GSW_TFLOW_COUNT_MODE_GET.

Table 35 GSWIP RMON Counter related APIs Enumerator Overview

Name	Description
<a href="#">GSW_portType_t</a>	Port Type - GSWIP-3.1 only. Used by GSW_portCfg_t.
<a href="#">GSW_RMON_CountMode_t</a>	RMON Counters Mode Enumeration. This enumeration defines Counters mode - Packets based or Bytes based counting. Metering and Routing Sessions RMON counting support either Byte based or packets based only.



**Table 35 GSWIP RMON Counter related APIs Enumerator Overview (cont'd)**

Name	Description
<a href="#">GSW_RMON_type_t</a>	RMON Counters Type enumeration. Used by GSW_RMON_clear_t and GSW_RMON_mode_t.
<a href="#">GSW_RmonMeterColor_t</a>	Used for getting metering RMON counters. Used by GSW_RMON_METER_GET.

## 1.7 MDIO\_RELAY\_APIS

This chapter describes the interface for PHY access via the MDIO BUS.

**Table 36 MDIO\_RELAY\_APIS Structure Overview**

Name	Description
<a href="#">mdio_relay_data</a>	Struct defined for MDIO Relay Data Structure
<a href="#">mdio_relay_mod_data</a>	Struct defined for MDIO Relay Mode Data Structure

**Table 37 MDIO\_RELAY\_APIS Function Overview**

Name	Description
<a href="#">ext_mdio_mod</a>	modify external GPHY MDIO/MMD registers via MDIO bus
<a href="#">ext_mdio_read</a>	read external GPHY MDIO/MMD registers via MDIO bus
<a href="#">ext_mdio_write</a>	write external GPHY MDIO/MMD registers via MDIO bus
<a href="#">int_gphy_mod</a>	modify internal GPHY MDIO/MMD registers
<a href="#">int_gphy_read</a>	read internal GPHY MDIO/MMD registers.
<a href="#">int_gphy_write</a>	write internal GPHY MDIO/MMD registers.

### 1.7.1 int\_gphy\_read

#### Description

read internal GPHY MDIO/MMD registers.

#### Prototype

```
void int_gphy_read (
    const GSW_Device_t * dev,
    struct mdio_relay_data * pdata );
```

#### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_data</a> *	pdata	The Pointer of MDIO Relay Data structure.	-

## 1.7.2 int\_gphy\_write

### Description

write internal GPHY MDIO/MMD registers.

### Prototype

```
void int_gphy_write (
    const GSW_Device_t * dev,
    struct mdio_relay_data * pdata );
```

### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_data</a> *	pdata	The Pointer of MDIO Relay Data structure.	-

## 1.7.3 int\_gphy\_mod

### Description

modify internal GPHY MDIO/MMD registers

### Prototype

```
void int_gphy_mod (
    const GSW_Device_t * dev,
    struct mdio_relay_mod_data * pdata );
```

### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_mod_data</a> *	pdata	The Pointer of MDIO Relay Mode Data structure.	-

## 1.7.4 ext\_mdio\_read

### Description

read external GPHY MDIO/MMD registers via MDIO bus

### Prototype

```
void ext_mdio_read (
    const GSW_Device_t * dev,
```



```
struct mdio_relay_data * pdata );
```

### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_data</a> *	pdata	The Pointer of MDIO Relay Mode Data structure.	-

## 1.7.5 ext\_mdio\_write

### Description

write external GPHY MDIO/MMD registers via MDIO bus

### Prototype

```
void ext_mdio_write (
    const GSW_Device_t * dev,
    struct mdio_relay_data * pdata );
```

### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_data</a> *	pdata	The Pointer of MDIO Relay Data structure.	-

## 1.7.6 ext\_mdio\_mod

### Description

modify external GPHY MDIO/MMD registers via MDIO bus

### Prototype

```
void ext_mdio_mod (
    const GSW_Device_t * dev,
    struct mdio_relay_mod_data * pdata );
```

### Parameters

Data Type	Name	Description	Dir
const GSW_Device_t *	dev	The GSW Device Pointer.	-
struct <a href="#">mdio_relay_mod_data</a> *	pdata	The Pointer of MDIO Relay Data Mode structure.	-

## 1.8 GPY211\_MDIO

## 1.9 GPY2XX\_GPIO\_FLAG

## 1.10 Data Types and Structure Reference

This chapter contains the reference of data types and structures of all modules.

### 1.10.1 Structure Reference

This chapter contains the Structure reference.

**Table 38 Structure Overview**

Name	Description
<a href="#">gpy2xx_4peye_cfg</a>	USXGMII 4-point eye test parameter. Used by gpy2xx_usxgmii_4peye_start and gpy2xx_usxgmii_4peye_cfg_get.
<a href="#">gpy2xx_4peye_result</a>	USXGMII 4-point eye test result. Used by gpy2xx_usxgmii_4peye_result.
<a href="#">gpy2xx_abist_pair</a>	Member used by pair in gpy2xx_abist_report.
<a href="#">gpy2xx_abist_report</a>	Analog built-in self-test report.
<a href="#">gpy2xx_ads_ctrl</a>	Link Speed Auto-downspeed Control.
<a href="#">gpy2xx_ads_sta</a>	Auto-downspeed status & config.
<a href="#">gpy2xx_cdiag_pair</a>	Member used by pair in gpy2xx_cdiag_report.
<a href="#">gpy2xx_cdiag_report</a>	Cable diagnostic report.
<a href="#">gpy2xx_cdiag_sum</a>	Member used by results in gpy2xx_cdiag_pair.
<a href="#">gpy2xx_device</a>	Data structure representing the GPHY entity. The user application uses this struct to communicate with GPHY APIs.
<a href="#">gpy2xx_id</a>	Member used by id in gpy2xx_device.
<a href="#">gpy2xx_led_cfg</a>	Data structure for LED configuration.
<a href="#">gpy2xx_link</a>	Member used by link in gpy2xx_device and gpy2xx_sgmi.
<a href="#">gpy2xx_pcs_status</a>	PCS status.
<a href="#">gpy2xx_phy_extin</a>	Data structure for external interrupt configuration.
<a href="#">gpy2xx_pvt</a>	GPHY temperature information.
<a href="#">gpy2xx_synce</a>	SyncE configuration.
<a href="#">gpy2xx_wolinfo</a>	Wake-on-LAN configuration.
<a href="#">GSW_B_TAG_Config_t</a>	B-TAG header definition .GSWIP-3.2 only Used by GSW_PBB_Tunnel_Template_Config_t.
<a href="#">GSW_BRIDGE_alloc_t</a>	Bridge Allocation. Used by GSW_BRIDGE_ALLOC and GSW_BRIDGE_FREE.
<a href="#">GSW_BRIDGE_config_t</a>	Bridge Configuration. Used by GSW_BRIDGE_CONFIG_SET and GSW_BRIDGE_CONFIG_GET.
<a href="#">GSW_BRIDGE_portAlloc_t</a>	Bridge Port Allocation. Used by GSW_BRIDGE_PORT_ALLOC and GSW_BRIDGE_PORT_FREE.

Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_BRIDGE_portConfig_t</a>	Bridge Port Configuration. Used by GSW_BRIDGE_PORT_CONFIG_SET and GSW_BRIDGE_PORT_CONFIG_GET.
<a href="#">GSW_cfg_t</a>	Global Switch configuration Attributes. Used by GSW_CFG_SET and GSW_CFG_GET.
<a href="#">GSW_CPU_PortCfg_t</a>	Defines one port that is directly connected to the CPU and its applicable settings. Used by GSW_CPU_PORT_CFG_SET and GSW_CPU_PORT_CFG_GET.
<a href="#">GSW_CTP_portAssignment_t</a>	CTP Port Assignment/association with logical port. Used by GSW_CTP_PORT_ASSIGNMENT_ALLOC, GSW_CTP_PORT_ASSIGNMENT_SET and GSW_CTP_PORT_ASSIGNMENT_GET.
<a href="#">GSW_CTP_portConfig_t</a>	CTP Port Configuration. Used by GSW_CTP_PORT_CONFIG_SET and GSW_CTP_PORT_CONFIG_GET.
<a href="#">GSW_Debug_RMON_Port_cnt_t</a>	For debugging Purpose only. Used for GSWIP 3.1.
<a href="#">GSW_debug_t</a>	For debugging Purpose only. Used for GSWIP 3.3.
<a href="#">GSW_DSCP2PCP_map_t</a>	DSCP to PCP Mapping. Used by GSW_DSCP2PCP_MAP_GET.
<a href="#">GSW_I_TAG_Config_t</a>	I-TAG header definition .GSWIP-3.2 only Used by GSW_PBB_Tunnel_Template_Config_t.
<a href="#">GSW_MAC_tableAdd_t</a>	MAC Table Entry to be added. Used by GSW_MAC_TABLE_ENTRY_ADD.
<a href="#">GSW_MAC_tableClearCond_t</a>	MAC Table Clear based on given condition. Used by GSW_MAC_TABLE_CLEAR_COND.
<a href="#">GSW_MAC_tableQuery_t</a>	Search for a MAC address entry in the address table. Used by GSW_MAC_TABLE_ENTRY_QUERY.
<a href="#">GSW_MAC_tableRead_t</a>	MAC Table Entry to be read. Used by GSW_MAC_TABLE_ENTRY_READ.
<a href="#">GSW_MAC_tableRemove_t</a>	MAC Table Entry to be removed. Used by GSW_MAC_TABLE_ENTRY_REMOVE.
<a href="#">GSW_MACFILTER_default_t</a>	Default MAC Address Filter. Used by GSW_DEFAULT_MAC_FILTER_SET and GSW_DEFAULT_MAC_FILTER_GET.
<a href="#">GSW_monitorPortCfg_t</a>	Port monitor configuration. Used by GSW_MONITOR_PORT_CFG_GET and GSW_MONITOR_PORT_CFG_SET.
<a href="#">GSW_multicastRouter_t</a>	Add an Ethernet port as router port to the switch hardware multicast table. Used by GSW_MULTICAST_ROUTER_PORT_ADD and GSW_MULTICAST_ROUTER_PORT_REMOVE.
<a href="#">GSW_multicastRouterRead_t</a>	Check if a port has been selected as a router port. Used by GSW_MULTICAST_ROUTER_PORT_READ. Not applicable to GSWIP-3.1.

Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_multicastSnoopCfg_t</a>	Configure the switch multicast configuration. Used by GSW_MULTICAST_SNOOP_CFG_SET and GSW_MULTICAST_SNOOP_CFG_GET.
<a href="#">GSW_multicastTable_t</a>	Add a host as a member to a multicast group. Used by GSW_MULTICAST_TABLE_ENTRY_ADD and GSW_MULTICAST_TABLE_ENTRY_REMOVE.
<a href="#">GSW_multicastTableRead_t</a>	Read out the multicast membership table. Used by GSW_MULTICAST_TABLE_ENTRY_READ.
<a href="#">GSW_PBB_Tunnel_Template_Config_t</a>	Tunnel Template Configuration. GSWIP-3.2 only Used by GSW_PBB_TunnelTemplate_Config_Set and GSW_PBB_TunnelTemplate_Config_Get For GSW_PBB_TunnelTemplate_Free, this field should be valid ID returned by GSW_PBB_TunnelTemplate_Alloc.
<a href="#">GSW_PCE_pattern_t</a>	Packet Classification Engine Pattern Configuration. GSWIP-3.0 has additional patterns such as Inner IP, Inner DSCP, Inner Protocol, Exclude Mode etc. Used by GSW_PCE_rule_t.
<a href="#">GSW_PCE_rule_t</a>	Parameter to add/read a rule to/from the packet classification engine. Used by GSW_PCE_RULE_WRITE and GSW_PCE_RULE_READ.
<a href="#">GSW_PCE_ruleDelete_t</a>	Parameter to delete a rule from the packet classification engine. Used by GSW_PCE_RULE_DELETE.
<a href="#">GSW_PMAC_BM_Cfg_t</a>	Configure the Backpressure mapping for egress Queues Congestion or ingress (receiving) ports to DMA channel. Used by GSW_PMAC_BM_CFG_SET and GSW_PMAC_BM_CFG_GET.
<a href="#">GSW_PMAC_Cnt_t</a>	PMAC Counters available for specified DMA Channel. Used by GSW_PMAC_COUNT_GET.
<a href="#">GSW_PMAC_Eg_Cfg_t</a>	Configure the PMAC Egress Configuration. (Upto 1024 entries) This Egress PMAC table is addressed through combination of following fields (Bit0 - Bit 9). nDestPortId (Bits 0-3) + Combination of [bMpe1Flag (Bit 4) + bMpe2Flag (Bit 5) + bEncFlag (Bit 6) + bDecFlag (Bit 7) ] or TrafficClass Value (Bits 4-7) + nFlowIdMSB (Bits 8-9). The bits 4-7 of index option is either based upon TC (default) or combination of Processing flags is decided through bProcFlagsEgPMACena. It is expected to pass the correct value in bProcFlagsSelect same as global bProcFlagsEgPMACena; Used by GSW_PMAC_EG_CFG_SET and GSW_PMAC_EG_CFG_GET.
<a href="#">GSW_PMAC_Glbl_Cfg_t</a>	Configure the global settings of PMAC for GSWIP-3.x. This includes settings such as Jumbo frame, Checksum handling, Padding and Engress PMAC Selector Config. Used by GSW_PMAC_GLBL_CFG_SET and GSW_PMAC_GLBL_CFG_GET.
<a href="#">GSW_PMAC_Ig_Cfg_t</a>	Configure the PMAC Ingress Configuration on a given Tx DMA channel to PMAC. (Upto 16 entries). This Ingress PMAC table is addressed through Transmit DMA Channel Identifier. Used by GSW_PMAC_IG_CFG_SET and GSW_PMAC_IG_CFG_GET.

Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_PMAPPER_t</a>	P-mapper Configuration Used by GSW_CTP_portConfig_t, GSW_BRIDGE_portConfig_t. In case of LAG, it is user's responsibility to provide the mapped entries in given P-mapper table. In other modes the entries are auto mapped from input packet.
<a href="#">GSW_portCfg_t</a>	Port Configuration Parameters. Used by GSW_PORT_CFG_GET and GSW_PORT_CFG_SET.
<a href="#">GSW_portLinkCfg_t</a>	Ethernet port link, speed status and flow control status. Used by GSW_PORT_LINK_CFG_GET and GSW_PORT_LINK_CFG_SET.
<a href="#">GSW_QoS_ClassPCP_Cfg_t</a>	Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW_QOS_CLASS_PCP_SET and GSW_QOS_CLASS_PCP_GET.
<a href="#">GSW_QoS_colorMarkingEntry_t</a>	Color Marking Table. There are standards to define the marking table. User should use GSW_QOS_COLOR_MARKING_TABLE_SET to initialize the table before color marking happens. GSW_QOS_COLOR_MARKING_TABLE_GET is used to get the marking table, mainly for debug purpose.
<a href="#">GSW_QoS_colorRemarkingEntry_t</a>	Color Remarking Table. There are standards to define the remarking table. User should use GSW_QOS_COLOR_REMARKING_TABLE_SET to initialize the table before color remarking happens. GSW_QOS_COLOR_REMARKING_TABLE_GET is used to get the remarking table, mainly for debug purpose.
<a href="#">GSW_QoS_DSCP_ClassCfg_t</a>	DSCP mapping table. Used by GSW_QOS_DSCP_CLASS_SET and GSW_QOS_DSCP_CLASS_GET.
<a href="#">GSW_QoS_DSCP_DropPrecedenceCfg_t</a>	DSCP to Drop Precedence assignment table configuration. Used by GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_SET and GSW_QOS_DSCP_DROP_PRECEDENCE_CFG_GET.
<a href="#">GSW_QoS_FlowCtrlCfg_t</a>	Configures the global buffer flow control threshold for conforming and non-conforming packets. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_FLOWCTRL_CFG_SET and GSW_QOS_FLOWCTRL_CFG_GET.
<a href="#">GSW_QoS_FlowCtrlPortCfg_t</a>	Configures the ingress port flow control threshold for used packet segments. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_FLOWCTRL_PORT_CFG_SET and GSW_QOS_FLOWCTRL_PORT_CFG_GET.

Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_meterCfg_t</a>	Configures the parameters of a rate meter instance. Used by GSW_QOS_METER_ALLOC, GSW_QOS_METER_FREE, GSW_QOS_METER_CFG_SET and GSW_QOS_METER_CFG_GET.
<a href="#">GSW_QoS_PCP_ClassCfg_t</a>	Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW_QOS_PCP_CLASS_SET and GSW_QOS_PCP_CLASS_GET.
<a href="#">GSW_QoS_portCfg_t</a>	Describes which priority information of ingress packets is used (taken into account) to identify the packet priority and the related egress priority queue. For DSCP, the priority to queue assignment is done using GSW_QOS_DSCP_CLASS_SET. For VLAN, the priority to queue assignment is done using GSW_QOS_PCP_CLASS_SET. Used by GSW_QOS_PORT_CFG_SET and GSW_QOS_PORT_CFG_GET.
<a href="#">GSW_QoS_portRemarkingCfg_t</a>	Port Remarking Configuration. Ingress and Egress remarking options for dedicated packet fields DSCP, CTAG VLAN PCP, STAG VLAN PCP and STAG VLAN DEI. Remarking is done either on the used traffic class or the drop precedence. Packet field specific remarking only applies on a packet if enabled on ingress and egress port. Used by GSW_QOS_PORT_REMARKING_CFG_SET and GSW_QOS_PORT_REMARKING_CFG_GET.
<a href="#">GSW_QoS_QueueBufferReserveCfg_t</a>	Reserved egress queue buffer segments. Used by GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_SET and GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_GET.
<a href="#">GSW_QoS_queuePort_t</a>	Sets the Queue ID for one traffic class of one port. Used by GSW_QOS_QUEUE_PORT_SET and GSW_QOS_QUEUE_PORT_GET.
<a href="#">GSW_QoS_schedulerCfg_t</a>	Configures the egress queues attached to a single port, and that are scheduled to transmit the queued Ethernet packets. Used by GSW_QOS_SCHEDULER_CFG_SET and GSW_QOS_SCHEDULER_CFG_GET.
<a href="#">GSW_QoS_ShaperCfg_t</a>	Configures a rate shaper instance with the rate and the burst size. Used by GSW_QOS_SHAPER_CFG_SET and GSW_QOS_SHAPER_CFG_GET.
<a href="#">GSW_QoS_ShaperQueue_t</a>	Assign one rate shaper instance to a QoS queue. Used by GSW_QOS_SHAPER_QUEUE_ASSIGN and GSW_QOS_SHAPER_QUEUE_DEASSIGN.
<a href="#">GSW_QoS_ShaperQueueGet_t</a>	Retrieve if a rate shaper instance is assigned to a QoS egress queue. Used by GSW_QOS_SHAPER_QUEUE_GET.
<a href="#">GSW_QoS_stormCfg_t</a>	Assigns one meter instances for storm control. Used by GSW_QOS_STORM_CFG_SET and GSW_QOS_STORM_CFG_GET. Not applicable to GSWIP-3.1.

Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_SVLAN_PCP_ClassCfg_t</a>	Traffic class associated with a particular STAG VLAN 802.1P (PCP) priority and Drop Eligible Indicator (DEI) mapping value. This table is global for the entire switch device. Priority map entry structure. The table index value is calculated by 'index=PCP + 8*DEI' Used by GSW_QOS_SVLAN_PCP_CLASS_SET and GSW_QOS_SVLAN_PCP_CLASS_GET.
<a href="#">GSW_QoS_WRED_Cfg_t</a>	Configures the global probability profile of the device. The min. and max. threshold values are given in number of packet buffer segments and required only in case of Manual Mode. The GSWIP-3.0/3.1 supports Auto mode and the threshold values are dynamically computed internally by GSWIP. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_CFG_SET and GSW_QOS_WRED_CFG_GET.
<a href="#">GSW_QoS_WRED_PortCfg_t</a>	Configures the WRED threshold parameter per port. The configured thresholds apply to fill level sum of all egress queues which are assigned to the egress port. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_PORT_CFG_SET and GSW_QOS_WRED_PORT_CFG_GET.
<a href="#">GSW_QoS_WRED_QueueCfg_t</a>	Configures the WRED threshold level values. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW_CAP_GET. Used by GSW_QOS_WRED_QUEUE_CFG_SET and GSW_QOS_WRED_QUEUE_CFG_GET.
<a href="#">GSW_register_mod_t</a>	Register access parameter to directly modify internal registers. Used by GSW_REGISTER_MOD.
<a href="#">GSW_register_t</a>	Register access parameter to directly read or write switch internal registers. Used by GSW_REGISTER_SET and GSW_REGISTER_GET.
<a href="#">GSW_RMON_clear_t</a>	RMON Counters Data Structure for clearance of values. Used by GSW_RMON_CLEAR.
<a href="#">GSW_RMON_flowGet_t</a>	Hardware platform extended RMON Counters. GSWIP-3.1 only. This structure contains additional RMON counters. These counters can be used by the packet classification engine and can be freely assigned to dedicated packet rules and flows. Used by GSW_RMON_FLOW_GET.
<a href="#">GSW_RMON_Meter_cnt_t</a>	RMON Counters for Meter - Type (GSWIP-3.0 only). This structure contains the RMON counters of one Meter Instance. Used by GSW_RMON_METER_GET.
<a href="#">GSW_RMON_mode_t</a>	RMON Counters Mode for different Elements. This structure takes RMON Counter Element Name and mode config.
<a href="#">GSW_RMON_Port_cnt_t</a>	RMON Counters for individual Port. This structure contains the RMON counters of an Ethernet Switch Port. Used by GSW_RMON_PORT_GET.



Table 38 Structure Overview (cont'd)

Name	Description
<a href="#">GSW_STP_BPDU_Rule_t</a>	Spanning tree packet detection and forwarding. Used by GSW_STP_BPDU_RULE_SET and GSW_STP_BPDU_RULE_GET.
<a href="#">GSW_STP_portCfg_t</a>	Configures the Spanning Tree Protocol state of an Ethernet port. Used by GSW_STP_PORT_CFG_SET and GSW_STP_PORT_CFG_GET.
<a href="#">GSW_TflowCmodeConf_t</a>	Hardware platform TFLOW counter mode. Supported modes include, Global (default), Logical, CTP, Bridge port mode. The number of counters that can be assigned varies based these mode type. Used by GSW_TFLOW_COUNT_MODE_SET and GSW_TFLOW_COUNT_MODE_GET.
<a href="#">GSW_trunkingCfg_t</a>	Global Ethernet trunking configuration. Used by GSW_TRUNKING_CFG_GET and GSW_TRUNKING_CFG_SET.
<a href="#">mdio_relay_data</a>	Struct defined for MDIO Relay Data Structure
<a href="#">mdio_relay_mod_data</a>	Struct defined for MDIO Relay Mode Data Structure

### 1.10.1.1 gpy2xx\_4peye\_cfg

#### Description

USXGMII 4-point eye test parameter. Used by [gpy2xx\\_usxgmii\\_4peye\\_start](#) and [gpy2xx\\_usxgmii\\_4peye\\_cfg\\_get](#).

#### Prototype

```
struct
{
    uint16_t ber;
    uint16_t errbit;
    bool is_opt;
    uint32_t mtime;
} gpy2xx_4peye_cfg;
```

#### Parameters

Data Type	Name	Description
uint16_t	ber	Bit Error Rate (BER) target in exp. For example BERT target 1E-9 if ber = 9. 9~12 is supported in current implementation.
uint16_t	errbit	Number of error bits in unit of 5. For example 15 error bits if errbit = 3.



Data Type	Name	Description
bool	is_opt	Optimization mode. Value 'true' to do 1E-9 with same number of error bits to get result range, then sweep the smaller range with give BER target.
uint32_t	mtime	Measurement time (optional). Measurement time for each data point. If this is 0, a default time calculated based BER target and number of error bits is used.

### 1.10.1.2 gpy2xx\_4peye\_result

#### Description

USXGMII 4-point eye test result. Used by [gpy2xx\\_usxgmii\\_4peye\\_result](#).

#### Prototype

```
struct
{
    int ret;
    int left;
    int right;
    int top;
    int bottom;
    uint32_t time;
} gpy2xx_4peye_result;
```

#### Parameters

Data Type	Name	Description
int	ret	Running state. < 0 - error code = 0 - not started = 1 - running = 2 - completed with valid result.
int	left	Running state. left of 4-point eye in UI.
int	right	Running state. right of 4-point eye in UI.
int	top	Running state. top of 4-point eye in 3.125mV.
int	bottom	Running state. bottom of 4-point eye in 3.125mV.
uint32_t	time	Measurement time in milliseconds (ms).

### 1.10.1.3 gpy2xx\_abist\_pair

#### Description

Member used by pair in [gpy2xx\\_abist\\_report](#).

Analog built-in self-test report of one pair

#### Prototype

```
struct
```

```

{
    uint8_t mag_max;
    uint8_t mag_min;
    uint8_t mag_avg;
    uint8_t mag_dc;
    struct gpy2xx_abist_pair::@0 icn_gmax;
    struct gpy2xx_abist_pair::@0 icn_gmin;
    uint8_t agc_pwr[6];
    uint8_t agc_mean;
    uint8_t agc_std;
    struct gpy2xx_abist_pair::@1 agc_hyb;
    struct gpy2xx_abist_pair::@1 agc_nohyb;
    uint8_t dc_mag;
    uint8_t nyq_mag;
    uint8_t k1_mag;
    uint8_t k2_mag;
    uint8_t k3_mag;
    uint8_t k4_mag;
    struct gpy2xx_abist_pair::@2 nohyb_10bt;
    struct gpy2xx_abist_pair::@2 nohyb_100bt;
    struct gpy2xx_abist_pair::@2 hyb_1000bt;
    struct gpy2xx_abist_pair::@2 nohyb_1000bt;
    struct gpy2xx_abist_pair::@2 hyb_2500bt;
    struct gpy2xx_abist_pair::@2 nohyb_2500bt;
} gpy2xx_abist_pair;

```

## Parameters

Data Type	Name	Description
uint8_t	mag_max	Max ADC noise injection magnitude.
uint8_t	mag_min	Min ADC noise injection magnitude.
uint8_t	mag_avg	Average ADC noise injection magnitude.
uint8_t	mag_dc	DC ADC noise injection magnitude.
struct gpy2xx_abist_pair::@0	icn_gmax	ADC noise (ICN_GMAX)
struct gpy2xx_abist_pair::@0	icn_gmin	DAC + LD noise (ICN_GMIN)
uint8_t	agc_pwr[6]	Measured power at +9/+6/+2/-1/-5/-8 dB gain respectively.
uint8_t	agc_mean	Mean of measured power at +9/+6/+2/-1/-5/-8 dB gain.
uint8_t	agc_std	Std measured power at +9/+6/+2/-1/-5/-8 dB gain.
struct gpy2xx_abist_pair::@1	agc_hyb	Measured power when hybrid ON.
struct gpy2xx_abist_pair::@1	agc_nohyb	Measured power when hybrid OFF.
uint8_t	dc_mag	FFT magnitude at DC.
uint8_t	nyq_mag	FFT magnitude at nyquist frequency 100 MHz.
uint8_t	k1_mag	FFT magnitude at fundamental frequency.

Data Type	Name	Description
uint8_t	k2_mag	Harmonics K2.
uint8_t	k3_mag	Harmonics K3.
uint8_t	k4_mag	Harmonics K4.
struct gpy2xx_abist_pair::@2	nohyb_10bt	Measured for 10bt mode with hybrid OFF.
struct gpy2xx_abist_pair::@2	nohyb_100bt	Measured for 100bt mode with hybrid OFF.
struct gpy2xx_abist_pair::@2	hyb_1000bt	Measured for 1000bt mode with hybrid ON.
struct gpy2xx_abist_pair::@2	nohyb_1000bt	Measured for 1000bt mode with hybrid OFF.
struct gpy2xx_abist_pair::@2	hyb_2500bt	Measured for 2500bt mode with hybrid ON.
struct gpy2xx_abist_pair::@2	nohyb_2500bt	Measured for 2500bt mode with hybrid OFF.

#### 1.10.1.4 gpy2xx\_abist\_report

##### Description

Analog built-in self-test report.

##### Prototype

```
struct
{
    struct gpy2xx_abist_pair pair[4];
} gpy2xx_abist_report;
```

##### Parameters

Data Type	Name	Description
struct <a href="#">gpy2xx_abist_pair</a>	pair[4]	Report of each pair: 0 - DSPA, 1 - DSPB, 2 - DSPC, 3 - DSPD.

#### 1.10.1.5 gpy2xx\_ads\_ctrl

##### Description

Link Speed Auto-downspeed Control.

##### Prototype

```
struct
{
    enum ads_adv_status no_nrg_rst;
    enum ads_nbt_ds_status downshift_en;
    uint16_t downshift_thr;
    enum ads_force_rst_status force_rst;
    uint8_t nrg_rst_cnt;
} gpy2xx_ads_ctrl;
```

## Parameters

Data Type	Name	Description
enum <a href="#">ads_adv_status</a>	no_nrg_rst	Auto-downspeed configuration.
enum <a href="#">ads_nbt_ds_status</a>	downshift_en	Auto-downspeed status.
uint16_t	downshift_thr	NBASE-T Downshift Training Counter Threshold, 0 - 15.
enum <a href="#">ads_force_rst_status</a>	force_rst	Force Reset of Downshift Process, 0-disable /1-enable.
uint8_t	nrg_rst_cnt	Timer to Reset the Downshift process, 0 - 255

### 1.10.1.6 gpy2xx\_ads\_sta

#### Description

Auto-downspeed status & config.

#### Prototype

```
struct
{
    uint8_t downshift_cnt;
    uint8_t downshift_2G5;
    uint8_t downshift_1G;
    struct gpy2xx_ads_ctrl ads_ctrl;
} gpy2xx_ads_sta;
```

## Parameters

Data Type	Name	Description
uint8_t	downshift_cnt	Training attempt counter, the number of attempt, when hit downshift_thr, downshift speed.
uint8_t	downshift_2G5	Downshift from 2.5 G to lower speed, 0-no downshift / 1-downshift.
uint8_t	downshift_1G	Downshift from 1 G to lower speed, 0-no downshift / 1-downshift.
struct <a href="#">gpy2xx_ads_ctrl</a>	ads_ctrl	ads control config

### 1.10.1.7 gpy2xx\_cdiag\_pair

#### Description

Member used by pair in [gpy2xx\\_cdiag\\_report](#).

Cable diagnostic report of one pair

#### Prototype

```
struct
{
```

```
uint16_t num_valid_result;
struct gpy2xx_cdiag_sum results[5];
int16_t xc_pwr[3];
} gpy2xx_cdiag_pair;
```

### Parameters

Data Type	Name	Description
uint16_t	num_valid_result	Number of valid results in results of gpy2xx_cdiag_pair.
struct <a href="#">gpy2xx_cdiag_sum</a>	results[5]	Up to 5 non-trivial echos are reported.
int16_t	xc_pwr[3]	16-bit signed short integer representing the sum-of-square of all XC(0~2) coefficients. This can be used to detect whether the cross-talk level is non-trivial.

### 1.10.1.8 gpy2xx\_cdiag\_report

#### Description

Cable diagnostic report.

#### Prototype

```
struct
{
    struct gpy2xx_cdiag_pair pair[4];
} gpy2xx_cdiag_report;
```

### Parameters

Data Type	Name	Description
struct <a href="#">gpy2xx_cdiag_pair</a>	pair[4]	Report of each pair: 0 - DSPA, 1 - DSPB, 2 - DSPC, 3 - DSPD.

### 1.10.1.9 gpy2xx\_cdiag\_sum

#### Description

Member used by results in [gpy2xx\\_cdiag\\_pair](#).

Cable diagnostic report of non-trivial echo

#### Prototype

```
struct
{
    uint8_t distance;
    uint8_t state;
    int16_t peak;
} gpy2xx_cdiag_sum;
```

## Parameters

Data Type	Name	Description
uint8_t	distance	Distance in meters.
uint8_t	state	Pair state (gpy2xx_cdiag_state)
int16_t	peak	16-bit signed short integer of the echo coefficient of the first detected peak

### 1.10.1.10 gpy2xx\_device

#### Description

Data structure representing the GPHY entity. The user application uses this struct to communicate with GPHY APIs.

#### Prototype

```

struct
{
    void(* lock)(void *lock_data);
    void(* unlock)(void *lock_data);
    void * lock_data;
    int(* mdiobus_read)(void *mdiobus_data, uint16_t addr, uint32_t regnum);
    int(* mdiobus_write)(void *mdiobus_data, uint16_t addr, uint32_t regnum,
uint16_t val);
    void * mdiobus_data;
    uint8_t smdio_addr;
    uint8_t phy_addr;
    void * priv_data;
    struct gpy2xx_id id;
    struct gpy2xx_link link;
    uint32_t wol_supported;
    unsigned int synce_supported: 1;
    unsigned int macsec_supported: 1;
    unsigned int mstr_slave: 1;
} gpy2xx_device;

```

## Parameters

Data Type	Name	Description
void(*	lock)(void *lock_data)	Function called by API when entering an API function. The user application should implement this function for resource protection in a multi-threaded application, or NULL for a single-threaded application.
void(*	unlock)(void *lock_data)	Function called by API when leaving an API function. The user application should implement this function for resource protection in a multi-threaded application, or NULL for a single-threaded application.

## Module Reference

Data Type	Name	Description
void *	lock_data	User data for lock, unlock of gpy2xx_device. The user application should provide proper value before calling any GPHY APIs.
int(*)	mdiobus_read)(void *mdiobus_data, uint16_t addr, uint32_t regnum)	Function for MDIO read operation on MDIO bus. The user application must implement this function for a GPHY API accessing the MDIO device. Both Clause 22 and Clause 45 addressing should be supported. If the MDIO master does not support Clause 45 addressing, the user application should use MMD indirect access registers at 0x13 and 0x14 for the implementation.
int(*)	mdiobus_write)(void *mdiobus_data, uint16_t addr, uint32_t regnum, uint16_t val)	Function for MDIO write operation on MDIO bus. The user application must implement this function for a GPHY API accessing the MDIO device. Both Clause 22 and Clause 45 addressing should be supported. If the MDIO master does not support Clause 45 addressing, the user application should use MMD indirect access registers at 0x13 and 0x14 for the implementation.
void *	mdiobus_data	User data for mdiobus_read and mdiobus_write of gpy2xx_device. The user application must provide proper value before calling any GPHY APIs.
uint8_t	smdio_addr	Slave MDIO address for internal register access. Default value is 0x1F. The user application must provide proper value before calling any GPHY APIs. The value must be the same as SMDIO_PDI_SMDIO_REGISTERS_SMDIO_CFG.ADDR.
uint8_t	phy_addr	This is the GPHY address for Standard MDIO and MMD register access. This address is from PMU_PDI_REGISTERS_GPHY_GPS1.MDIO_PHY_ADDR which is pin-strapped. The user application must call gpy2xx_init before any other APIs for initialization.
void *	priv_data	This is private data for GPHY APIs. The user application must call gpy2xx_init before any other APIs for initialization. And call gpy2xx_uninit to free the device when cleaning up.
struct <a href="#">gpy2xx_id</a>	id	PHY ID.

Data Type	Name	Description
struct <a href="#">gpy2xx_link</a>	link	Link configuration and status. The user application uses APIs defined in Link APIs, such as <code>gpy2xx_config_advert</code> , <code>gpy2xx_setup_forced</code> , <code>gpy2xx_restart_aneg</code> , <code>gpy2xx_config_aneg</code> , <code>gpy2xx_aneg_done</code> , <code>gpy2xx_update_link</code> , and <code>gpy2xx_read_status</code> for configuration purposes or to read out information.
uint32_t	wol_supported	Flags for supported Wake-on-LAN modes. Values (Wake-on-LAN Flags) can be combined with "or".
unsigned int	synce_supported: 1	SyncE capable (value 1) or not capable (value 0)
unsigned int	macsec_supported: 1	MACsec capable (value 1) or not capable (value 0)
unsigned int	mstr_slave: 1	Allows forcing of master or slave mode manually. Applicable when choosing forced link speed of 2.5G or 1G only.

### 1.10.1.11 gpy2xx\_id

#### Description

Member used by id in [gpy2xx\\_device](#).

GPHY ID information such as manufacturer data, firmware or API version

#### Prototype

```
struct
{
    uint32_t OUI;
    uint8_t model_no;
    uint8_t family;
    uint8_t revision;
    unsigned int fw_release: 1;
    unsigned int fw_major: 7;
    uint8_t fw_minor;
    enum gpy2xx_fwboot_mode fw_memory;
    uint16_t drv_major;
    uint16_t drv_minor;
    uint16_t drv_release;
    uint16_t drv_patch;
} gpy2xx_id;
```



## Parameters

Data Type	Name	Description
uint32_t	OUI	PHY organizationally unique identifier.
uint8_t	model_no	PHY manufacturer's model number.
uint8_t	family	PHY manufacturer's model family.
uint8_t	revision	PHY revision number.
unsigned int	fw_release: 1	The most recently read firmware release indication: 0 - test version, 1 - released version.
unsigned int	fw_major: 7	The most recently read firmware major version number.
uint8_t	fw_minor	The most recently read firmware minor version number.
enum <a href="#">gpy2xx_fwboot_mode</a>	fw_memory	The memory target used for firmware execution, valid from FW 8747 onwards only. Valid values are defined in <a href="#">gpy2xx_fwboot_mode</a> enum.
uint16_t	drv_major	API major version number.
uint16_t	drv_minor	API minor version number.
uint16_t	drv_release	API release indication: 0 - test version, 1 - general available (GA) release, >= 2 - maintenance release (MR)
uint16_t	drv_patch	The most recently read firmware patch indication: 1 - GA/MR release, >= 2 - patch release.

### 1.10.1.12 gpy2xx\_led\_cfg

#### Description

Data structure for LED configuration.

#### Prototype

```
struct
{
    int id;
    enum gpy2xx_led_colormode color_mode;
    enum gpy2xx_led_bsrc slow_blink_src;
    enum gpy2xx_led_bsrc fast_blink_src;
    enum gpy2xx_led_bsrc const_on;
    uint32_t pulse;
} gpy2xx_led_cfg;
```

## Parameters

Data Type	Name	Description
int	id	LED ID (0~3) (LED Function Config APIs)
enum <a href="#">gpy2xx_led_colormode</a>	color_mode	LED color mode. Valid values are gpy2xx_led_colormode. NOTE: This is for internal use only.
enum <a href="#">gpy2xx_led_bsrc</a>	slow_blink_src	Select in which PHY state the LED blinks with slow frequency. Valid values are defined in gpy2xx_led_bsrc enum.
enum <a href="#">gpy2xx_led_bsrc</a>	fast_blink_src	Select in which PHY state the LED blinks with fast frequency. Valid values are defined in gpy2xx_led_bsrc enum.
enum <a href="#">gpy2xx_led_bsrc</a>	const_on	Select in which PHY status the LED is constantly on. Valid values are defined in gpy2xx_led_bsrc enum.
uint32_t	pulse	Pulsing configuration. Values (gpy2xx_led_pulse) can be combined with "or".

### 1.10.1.13 gpy2xx\_link

#### Description

Member used by link in [gpy2xx\\_device](#) and gpy2xx\_sgmii.

Data structure representing GPHY link configuration and status

#### Prototype

```
struct
{
    int16_t speed;
    unsigned int duplex: 2;
    unsigned int pause: 1;
    unsigned int asym_pause: 1;
    unsigned int link: 1;
    unsigned int fixed2g5: 1;
    unsigned int autoneg: 1;
    uint64_t supported;
    uint64_t advertising;
    uint64_t lp_advertising;
} gpy2xx_link;
```

#### Parameters

Data Type	Name	Description
int16_t	speed	Link speed (forced) or partner link speed (auto-negotiation) defined by Link Speed macros.
unsigned int	duplex: 2	Duplex (forced) or partner duplex (auto-negotiation) defined by Duplex macros.

Data Type	Name	Description
unsigned int	pause: 1	Partner pause (auto-negotiation)
unsigned int	asym_pause: 1	Partner asym-pause (auto-negotiation)
unsigned int	link: 1	The most recently read link state.
unsigned int	fixed2g5: 1	TPI speed or forced 2.5G.
unsigned int	autoneg: 1	Enable auto-negotiation. Value 1 to enable auto-negotiation, value 0 to force link.
uint64_t	supported	Union of GPHY supported modes listed in Supported Link Mode macros. This is updated when gpy2xx_init is called and should not be changed by the user application.
uint64_t	advertising	Union of GPHY advertising modes listed in Advertised Link Mode macros.
uint64_t	lp_advertising	Union of partner advertising modes listed in Advertised Link Mode macros.

#### 1.10.1.14 gpy2xx\_pcs\_status

##### Description

PCS status.

##### Prototype

```
struct
{
    uint32_t ber;
    uint32_t errored_block;
    uint8_t high_ber;
    uint8_t block_lock;
    uint8_t rcv_link_up;
} gpy2xx_pcs_status;
```

##### Parameters

Data Type	Name	Description
uint32_t	ber	Bit error rate (BER)
uint32_t	errored_block	Number of errored blocks.
uint8_t	high_ber	1 indicates high bit error rate (BER)
uint8_t	block_lock	0 indicates loss of block lock
uint8_t	rcv_link_up	1 indicates PCS receive link up

#### 1.10.1.15 gpy2xx\_phy\_extin

##### Description

Data structure for external interrupt configuration.

### Prototype

```
struct
{
    enum gpy2xx_extin_phy_event std_imask;
    enum gpy2xx_extin_phy_event std_istat;
    enum gpy2xx_extin_im2_mask ext_imask;
    enum gpy2xx_extin_im2_mask ext_istat;
} gpy2xx_phy_extin;
```

### Parameters

Data Type	Name	Description
enum <a href="#">gpy2xx_extin_phy_event</a>	std_imask	Standard interrupt mask. Valid values are defined in gpy2xx_extin_phy_event enum.
enum <a href="#">gpy2xx_extin_phy_event</a>	std_istat	Standard interrupt status. Valid values are defined in gpy2xx_extin_phy_event enum.
enum <a href="#">gpy2xx_extin_im2_mask</a>	ext_imask	Extended interrupt mask. Valid values are defined in gpy2xx_extin_im2_mask enum.
enum <a href="#">gpy2xx_extin_im2_mask</a>	ext_istat	Extended interrupt status. Valid values are defined in gpy2xx_extin_im2_mask enum.

## 1.10.1.16 gpy2xx\_pvt

### Description

GPHY temperature information.

### Prototype

```
struct
{
    int temperature;
} gpy2xx_pvt;
```

### Parameters

Data Type	Name	Description
int	temperature	GPHY temperature in degree Celsius scale.

## 1.10.1.17 gpy2xx\_synce

### Description

SyncE configuration.

### Prototype

```
struct
{
    char synce_enable;
    enum gpy2xx_synce_clk synce_refclk;
```

```
enum gpy2xx_synce_master_mode master_sel;
enum gpy2xx_data_rate data_rate;
enum gpy2xx_gpc_sel gpc_sel;
} gpy2xx_synce;
```

### Parameters

Data Type	Name	Description
char	synce_enable	Enable SyncE. Value 1 to enable, value 0 to disable.
enum <a href="#">gpy2xx_synce_clk</a>	synce_refclk	Select input frequency of reference clock. Valid values are defined in gpy2xx_synce_clk enum.
enum <a href="#">gpy2xx_synce_master_mode</a>	master_sel	Master/slave select. Value 0 to select slave mode, else to select master mode.
enum <a href="#">gpy2xx_data_rate</a>	data_rate	Data rate (1G or 2.5G)
enum <a href="#">gpy2xx_gpc_sel</a>	gpc_sel	GPC select. Configure GPC and GPIOs.

### 1.10.1.18 gpy2xx\_wolinfo

#### Description

Wake-on-LAN configuration.

#### Prototype

```
struct
{
    uint32_t wolopts;
    uint8_t mac[6];
    uint8_t sopass[6];
} gpy2xx_wolinfo;
```

### Parameters

Data Type	Name	Description
uint32_t	wolopts	Flags for enabled Wake-on-LAN modes. Values (Wake-on-LAN Flags) can be combined with "or". If this is 0, Wake-on-LAN is disabled.
uint8_t	mac[6]	Wake-on-LAN designated MAC.
uint8_t	sopass[6]	Wake-on-LAN SecureON password. This is only meaningful if WAKE_MAGICSECURE is set in wolopts of gpy2xx_wolinfo.

### 1.10.1.19 GSW\_B\_TAG\_Config\_t

#### Description

B-TAG header definition .GSWIP-3.2 only Used by [GSW\\_PBB\\_Tunnel\\_Template\\_Config\\_t](#).

### Prototype

```
struct
{
    gsw_bool_t bTpidEnable;
    u16 nTpid;
    gsw_bool_t bPcpEnable;
    u8 nPcp;
    gsw_bool_t bDeiEnable;
    u8 nDei;
    gsw_bool_t bVidEnable;
    u16 nVid;
} GSW_B_TAG_Config_t;
```

### Parameters

Data Type	Name	Description
gsw_bool_t	bTpidEnable	B-TAG TPID -2 bytes field
u16	nTpid	
gsw_bool_t	bPcpEnable	B-TAG PCP -3 Bit field
u8	nPcp	
gsw_bool_t	bDeiEnable	B-TAG DEI -1 Bit field
u8	nDei	
gsw_bool_t	bVidEnable	B-TAG VID -12 Bit field
u16	nVid	

#### 1.10.1.20 GSW\_BRIDGE\_alloc\_t

### Description

Bridge Allocation. Used by GSW\_BRIDGE\_ALLOC and GSW\_BRIDGE\_FREE.

### Prototype

```
struct
{
    u16 nBridgeId;
} GSW_BRIDGE_alloc_t;
```

### Parameters

Data Type	Name	Description
u16	nBridgeId	If GSW_BRIDGE_ALLOC is successful, a valid ID will be returned in this field. Otherwise, INVALID_HANDLE is returned in this field. For GSW_BRIDGE_FREE, this field should be valid ID returned by GSW_BRIDGE_ALLOC. ID 0 is special Bridge created during initialization.

### 1.10.1.21 GSW\_BRIDGE\_config\_t

#### Description

Bridge Configuration. Used by GSW\_BRIDGE\_CONFIG\_SET and GSW\_BRIDGE\_CONFIG\_GET.

#### Prototype

```
struct
{
    u16 nBridgeId;
    GSW_BridgeConfigMask_t eMask;
    gsw_bool_t bMacLearningLimitEnable;
    u16 nMacLearningLimit;
    u16 nMacLearningCount;
    u32 nLearningDiscardEvent;
    gsw_bool_t bSubMeteringEnable[GSW_BRIDGE_PORT_EGRESS_METER_MAX];
    u16 nTrafficSubMeterId[GSW_BRIDGE_PORT_EGRESS_METER_MAX];
    GSW_BridgeForwardMode_t eForwardBroadcast;
    GSW_BridgeForwardMode_t eForwardUnknownMulticastIp;
    GSW_BridgeForwardMode_t eForwardUnknownMulticastNonIp;
    GSW_BridgeForwardMode_t eForwardUnknownUnicast;
} GSW_BRIDGE_config_t;
```

#### Parameters

Data Type	Name	Description
u16	nBridgeId	Bridge ID (FID) allocated by GSW_BRIDGE_ALLOC.  <i>Note: If <a href="#">GSW_BRIDGE_config_t</a> has <a href="#">GSW_BridgeConfigMask_t</a>, this field is absolute index of Bridge (FID) in hardware for debug purpose, bypassing any check.</i>
<a href="#">GSW_BridgeConfigMask_t</a>	eMask	Mask for updating/retrieving fields.
gsw_bool_t	bMacLearningLimitEnable	Enable MAC learning limitation.
u16	nMacLearningLimit	Max number of MAC can be learned in this bridge (all bridge ports).
u16	nMacLearningCount	Get number of MAC address learned from this bridge port.
u32	nLearningDiscardEvent	Number of learning discard event due to hardware resource not available.  <i>Note: This is discard event due to either MAC table full or Hash collision. Discard due to nMacLearningCount reached is not counted in this field.</i>
gsw_bool_t	bSubMeteringEnable[GSW_BRIDGE_PORT_EGRESS_METER_MAX]	Traffic metering on type of traffic (such as broadcast, multicast, unknown unicast, etc) applies.

Data Type	Name	Description
u16	nTrafficSubMeterId[GSW_BRIDGE_PORT_EGRESS_METER_MAX]	Meter for bridge process with specific type (such as broadcast, multicast, unknown unicast, etc). Need pre-allocated for each type.
<a href="#">GSW_BridgeForwardMode_t</a>	eForwardBroadcast	Forwarding mode of broadcast traffic.
<a href="#">GSW_BridgeForwardMode_t</a>	eForwardUnknownMulticastIpl	Forwarding mode of unknown multicast IP traffic.
<a href="#">GSW_BridgeForwardMode_t</a>	eForwardUnknownMulticastNonIpl	Forwarding mode of unknown multicast non-IP traffic.
<a href="#">GSW_BridgeForwardMode_t</a>	eForwardUnknownUnicast	Forwarding mode of unknown unicast traffic.

### 1.10.1.22 GSW\_BRIDGE\_portAlloc\_t

#### Description

Bridge Port Allocation. Used by GSW\_BRIDGE\_PORT\_ALLOC and GSW\_BRIDGE\_PORT\_FREE.

#### Prototype

```
struct
{
    u16 nBridgePortId;
} GSW_BRIDGE_portAlloc_t;
```

#### Parameters

Data Type	Name	Description
u16	nBridgePortId	If GSW_BRIDGE_PORT_ALLOC is successful, a valid ID will be returned in this field. Otherwise, INVALID_HANDLE is returned in this field. For GSW_BRIDGE_PORT_FREE, this field should be valid ID returned by GSW_BRIDGE_PORT_ALLOC. ID 0 is special for CPU port in PRX300 by mapping to CTP 0 (Logical Port 0 with Sub-interface ID 0), and pre-allocated during initialization.

### 1.10.1.23 GSW\_BRIDGE\_portConfig\_t

#### Description

Bridge Port Configuration. Used by GSW\_BRIDGE\_PORT\_CONFIG\_SET and GSW\_BRIDGE\_PORT\_CONFIG\_GET.

#### Prototype

```
struct
{
    u16 nBridgePortId;
    GSW_BridgePortConfigMask_t eMask;
```



```
ul6 nBridgeId;
gsw_bool_t bIngressExtendedVlanEnable;
ul6 nIngressExtendedVlanBlockId;
ul6 nIngressExtendedVlanBlockSize;
gsw_bool_t bEgressExtendedVlanEnable;
ul6 nEgressExtendedVlanBlockId;
ul6 nEgressExtendedVlanBlockSize;
GSW_ColorMarkingMode_t eIngressMarkingMode;
GSW_ColorRemarkingMode_t eEgressRemarkingMode;
gsw_bool_t bIngressMeteringEnable;
ul6 nIngressTrafficMeterId;
gsw_bool_t bEgressSubMeteringEnable[GSW_BRIDGE_PORT_EGRESS_METER_MAX];
ul6 nEgressTrafficSubMeterId[GSW_BRIDGE_PORT_EGRESS_METER_MAX];
u8 nDestLogicalPortId;
gsw_bool_t bPmapperEnable;
ul6 nDestSubIfIdGroup;
GSW_PmapperMappingMode_t ePmapperMappingMode;
gsw_bool_t bPmapperIdValid;
GSW_PMAPPER_t sPmapper;
ul6 nBridgePortMap[8];
gsw_bool_t bMcDestIpLookupDisable;
gsw_bool_t bMcSrcIpLookupEnable;
gsw_bool_t bDestMacLookupDisable;
gsw_bool_t bSrcMacLearningDisable;
gsw_bool_t bMacSpoofingDetectEnable;
gsw_bool_t bPortLockEnable;
gsw_bool_t bMacLearningLimitEnable;
ul6 nMacLearningLimit;
ul6 nLoopViolationCount;
ul6 nMacLearningCount;
gsw_bool_t bIngressVlanFilterEnable;
ul6 nIngressVlanFilterBlockId;
ul6 nIngressVlanFilterBlockSize;
gsw_bool_t bBypassEgressVlanFilter1;
gsw_bool_t bEgressVlanFilter1Enable;
ul6 nEgressVlanFilter1BlockId;
ul6 nEgressVlanFilter1BlockSize;
gsw_bool_t bEgressVlanFilter2Enable;
ul6 nEgressVlanFilter2BlockId;
ul6 nEgressVlanFilter2BlockSize;
gsw_bool_t bIngressVlanBasedMacLearningEnable;
gsw_bool_t bVlanTagSelection;
gsw_bool_t bVlanSrcMacPriorityEnable;
gsw_bool_t bVlanSrcMacDEIEnable;
gsw_bool_t bVlanSrcMacVidEnable;
gsw_bool_t bVlanDstMacPriorityEnable;
gsw_bool_t bVlanDstMacDEIEnable;
gsw_bool_t bVlanDstMacVidEnable;
gsw_bool_t bVlanBasedMultiCastLookup;
gsw_bool_t bVlanMulticastPriorityEnable;
gsw_bool_t bVlanMulticastDEIEnable;
```

```

    gsw_bool_t bVlanMulticastVidEnable;
} GSW_BRIDGE_portConfig_t;

```

## Parameters

Data Type	Name	Description
u16	nBridgePortId	Bridge Port ID allocated by GSW_BRIDGE_PORT_ALLOC. <i>Note: If <a href="#">GSW_BRIDGE_portConfig_t</a> has <a href="#">GSW_BridgePortConfigMask_t</a>, this field is absolute index of Bridge Port in hardware for debug purpose, bypassing any check.</i>
<a href="#">GSW_BridgePortConfigMask_t</a>	eMask	Mask for updating/retrieving fields.
u16	nBridgeId	Bridge ID (FID) to which this bridge port is associated. A default bridge (ID 0) should be always available.
gsw_bool_t	bIngressExtendedVlanEnable	Enable extended VLAN processing for ingress non-IGMP traffic.
u16	nIngressExtendedVlanBlockId	Extended VLAN block allocated for ingress non-IGMP traffic. It defines extended VLAN process for ingress non-IGMP traffic. Valid when bIngressExtendedVlanEnable is TRUE.
u16	nIngressExtendedVlanBlockSize	Extended VLAN block size for ingress non-IGMP traffic. This is optional. If it is 0, the block size of nIngressExtendedVlanBlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bEgressExtendedVlanEnable	Enable extended VLAN processing enabled for egress non-IGMP traffic.
u16	nEgressExtendedVlanBlockId	Extended VLAN block allocated for egress non-IGMP traffic. It defines extended VLAN process for egress non-IGMP traffic. Valid when bEgressExtendedVlanEnable is TRUE.
u16	nEgressExtendedVlanBlockSize	Extended VLAN block size for egress non-IGMP traffic. This is optional. If it is 0, the block size of nEgressExtendedVlanBlockId will be used. Otherwise, this field will be used.
<a href="#">GSW_ColorMarkingMode_t</a>	eIngressMarkingMode	Ingress color marking mode for ingress traffic.
<a href="#">GSW_ColorRemarkingMode_t</a>	eEgressRemarkingMode	Color remarking for egress traffic.
gsw_bool_t	bIngressMeteringEnable	Traffic metering on ingress traffic applies.

## Module Reference

Data Type	Name	Description
u16	nIngressTrafficMeterId	Meter for ingress Bridge Port process. <i>Note: Meter should be allocated with GSW_QOS_METER_ALLOC before Bridge port configuration. If this Bridge port is re-set, the last used meter should be released.</i>
gsw_bool_t	bEgressSubMeteringEnable[GSW_BRIDGE_PORT_EGRESS_METER_MAX]	Traffic metering on various types of egress traffic (such as broadcast, multicast, unknown unicast, etc) applies.
u16	nEgressTrafficSubMeterId[GSW_BRIDGE_PORT_EGRESS_METER_MAX]	Meter for egress Bridge Port process with specific type (such as broadcast, multicast, unknown unicast, etc). Need pre-allocated for each type.
u8	nDestLogicalPortId	This field defines destination logical port.
gsw_bool_t	bPmapperEnable	This field indicates whether to enable P-mapper.
u16	nDestSubIfIdGroup	When bPmapperEnable is FALSE, this field defines destination sub interface ID group.
<b>GSW_PmapperMappingMode_t</b>	ePmapperMappingMode	When bPmapperEnable is TRUE, this field selects either DSCP or PCP to derive sub interface ID.
gsw_bool_t	bPmapperIdValid	When bPmapperEnable is TRUE, P-mapper is used. This field determines whether sPmapper.nPmapperId is valid. If this field is TRUE, the P-mapper is re-used and no allocation of new P-mapper or value change in the P-mapper. If this field is FALSE, allocation is taken care in the API implementation.
<b>GSW_PMAPPER_t</b>	sPmapper	When bPmapperEnable is TRUE, P-mapper is used. if bPmapperIdValid is FALSE, API implementation need take care of P-mapper allocation, and maintain the reference counter of P-mapper used multiple times. If bPmapperIdValid is TRUE, only sPmapper.nPmapperId is used to associate the P-mapper, and there is no allocation of new P-mapper or value change in the P-mapper.
u16	nBridgePortMap[8]	Port map define broadcast domain. <i>Note: Each bit is one bridge port. Bridge port ID is index * 16 + bit offset. For example, bit 1 of nBridgePortMap[1] is bridge port ID 17.</i>
gsw_bool_t	bMcDestIpLookupDisable	Multicast IP table is searched if this field is FALSE and traffic is IP multicast.
gsw_bool_t	bMcSrcIpLookupEnable	Multicast IP table is searched if this field is TRUE and traffic is IP multicast.

## Module Reference

Data Type	Name	Description
gsw_bool_t	bDestMacLookupDisable	Default is FALSE. Packet is treated as "unknown" if it's not broadcast/multicast packet.
gsw_bool_t	bSrcMacLearningDisable	Default is FALSE. Source MAC address is learned.
gsw_bool_t	bMacSpoofingDetectEnable	If this field is TRUE and MAC address which is already learned in another bridge port appears on this bridge port, port locking violation is detected.
gsw_bool_t	bPortLockEnable	If this field is TRUE and MAC address which is already learned in this bridge port appears on another bridge port, port locking violation is detected.
gsw_bool_t	bMacLearningLimitEnable	Enable MAC learning limitation.
u16	nMacLearningLimit	Max number of MAC can be learned from this bridge port.
u16	nLoopViolationCount	Get number of Loop violation counter from this bridge port.
u16	nMacLearningCount	Get number of MAC address learned from this bridge port.
gsw_bool_t	bIngressVlanFilterEnable	Enable ingress VLAN filter
u16	nIngressVlanFilterBlockId	VLAN filter block of ingress traffic if <a href="#">GSW_BRIDGE_portConfig_t</a> is TRUE.
u16	nIngressVlanFilterBlockSize	VLAN filter block size. This is optional. If it is 0, the block size of nIngressVlanFilterBlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bBypassEgressVlanFilter1	For ingress traffic, bypass VLAN filter 1 at egress bridge port processing.
gsw_bool_t	bEgressVlanFilter1Enable	Enable egress VLAN filter 1
u16	nEgressVlanFilter1BlockId	VLAN filter block 1 of egress traffic if <a href="#">GSW_BRIDGE_portConfig_t</a> is TRUE.
u16	nEgressVlanFilter1BlockSize	VLAN filter block 1 size. This is optional. If it is 0, the block size of nEgressVlanFilter1BlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bEgressVlanFilter2Enable	Enable egress VLAN filter 2
u16	nEgressVlanFilter2BlockId	VLAN filter block 2 of egress traffic if <a href="#">GSW_BRIDGE_portConfig_t</a> is TRUE.
u16	nEgressVlanFilter2BlockSize	VLAN filter block 2 size. This is optional. If it is 0, the block size of nEgressVlanFilter2BlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bIngressVlanBasedMacLearningEnable	Enable Ingress VLAN Based Mac Learning

Module Reference

Data Type	Name	Description
gsw_bool_t	bVlanTagSelection	0 - Intermediate outer VLAN tag is used for MAC address/multicast learning, lookup and filtering. 1 - Original outer VLAN tag is used for MAC address/multicast learning, lookup and filtering.
gsw_bool_t	bVlanSrcMacPriorityEnable	0 - Disable, VLAN Priority field is not used and value 0 is used for source MAC address learning and filtering. 1 - Enable, VLAN Priority field is used for source MAC address learning and filtering.
gsw_bool_t	bVlanSrcMacDEIEnable	0 - Disable, VLAN DEI/CFI field is not used and value 0 is used for source MAC address learning and filtering. 1 - Enable, VLAN DEI/CFI field is used for source MAC address learning and filtering.
gsw_bool_t	bVlanSrcMacVidEnable	0 - Disable, VLAN ID field is not used and value 0 is used for source MAC address learning and filtering. 1 - Enable, VLAN ID field is used for source MAC address learning and filtering.
gsw_bool_t	bVlanDstMacPriorityEnable	0 - Disable, VLAN Priority field is not used and value 0 is used for destination MAC address look up and filtering. 1 - Enable, VLAN Priority field is used for destination MAC address look up and filtering.
gsw_bool_t	bVlanDstMacDEIEnable	0 - Disable, VLAN CFI/DEI field is not used and value 0 is used for destination MAC address lookup and filtering. 1 - Enable, VLAN CFI/DEI field is used for destination MAC address look up and filtering.
gsw_bool_t	bVlanDstMacVidEnable	0 - Disable, VLAN ID field is not used and value 0 is used for destination MAC address look up and filtering. 1 - Enable, VLAN ID field is destination for destination MAC address look up and filtering.
gsw_bool_t	bVlanBasedMultiCastLookup	Enable, VLAN Based Multicast Lookup
gsw_bool_t	bVlanMulticastPriorityEnable	0 - Disable, VLAN Priority field is not used and value 0 is used for IP multicast lookup. 1 - Enable, VLAN Priority field is used for IP multicast lookup.
gsw_bool_t	bVlanMulticastDEIEnable	0 - Disable, VLAN CFI/DEI field is not used and value 0 is used for IP multicast lookup. 1 - Enable, VLAN CFI/DEI field is used for IP multicast lookup.
gsw_bool_t	bVlanMulticastVidEnable	0 - Disable, VLAN ID field is not used and value 0 is used for IP multicast lookup. 1 - Enable, VLAN ID field is destination for IP multicast lookup.

### 1.10.1.24 GSW\_cfg\_t

#### Description

Global Switch configuration Attributes. Used by GSW\_CFG\_SET and GSW\_CFG\_GET.

#### Prototype

```
struct
{
    GSW_ageTimer_t eMAC_TableAgeTimer;
    u32 nAgeTimer;
    u16 nMaxPacketLen;
    gsw_bool_t bLearningLimitAction;
    gsw_bool_t bMAC_LockingAction;
    gsw_bool_t bMAC_SpoofingAction;
    gsw_bool_t bPauseMAC_ModeSrc;
    u8 nPauseMAC_Src[GSW_MAC_ADDR_LEN];
} GSW_cfg_t;
```

#### Parameters

Data Type	Name	Description
<a href="#">GSW_ageTimer_t</a>	eMAC_TableAgeTimer	MAC table aging timer. After this timer expires the MAC table entry is aged out.
u32	nAgeTimer	If eMAC_TableAgeTimer = GSW_AGETIMER_CUSTOM, this variable defines MAC table aging timer in seconds.
u16	nMaxPacketLen	Maximum Ethernet packet length.
gsw_bool_t	bLearningLimitAction	Automatic MAC address table learning limitation consecutive action. These frame addresses are not learned, but there exists control as to whether the frame is still forwarded or dropped. <ul style="list-style-type: none"> <li>False: Drop</li> <li>True: Forward</li> </ul>
gsw_bool_t	bMAC_LockingAction	Accept or discard MAC port locking violation packets. MAC spoofing detection features identifies ingress packets that carry a MAC source address which was previously learned on a different ingress port (learned by MAC bridging table). This also applies to static added entries. MAC address port locking is configured on port level by 'bLearningMAC_PortLock'. <ul style="list-style-type: none"> <li>False: Drop</li> <li>True: Forward</li> </ul>

Data Type	Name	Description
gsw_bool_t	bMAC_SpoofingAction	Accept or discard MAC spoofing and port MAC locking violation packets. MAC spoofing detection features identifies ingress packets that carry a MAC source address which was previously learned on a different ingress port (learned by MAC bridging table). This also applies to static added entries. MAC spoofing detection is enabled on port level by 'bMAC_SpoofingDetection'. <ul style="list-style-type: none"> <li>False: Drop</li> <li>True: Forward</li> </ul>
gsw_bool_t	bPauseMAC_ModeSrc	Pause frame MAC source address mode. If enabled, use the alternative address specified with 'nMAC'.
u8	nPauseMAC_Src[GSW_MAC_ADDR_LEN]	Pause frame MAC source address.

### 1.10.1.25 GSW\_CPU\_PortCfg\_t

#### Description

Defines one port that is directly connected to the CPU and its applicable settings. Used by GSW\_CPU\_PORT\_CFG\_SET and GSW\_CPU\_PORT\_CFG\_GET.

#### Prototype

```

struct
{
    u16 nPortId;
    gsw_bool_t bCPU_PortValid;
    gsw_bool_t bSpecialTagIngress;
    gsw_bool_t bSpecialTagEgress;
    gsw_bool_t bFcsCheck;
    gsw_bool_t bFcsGenerate;
    GSW_CPU_SpecialTagEthType_t bSpecialTagEthType;
    GSW_CPU_ParserHeaderCfg_t eNoMPEParserCfg;
    GSW_CPU_ParserHeaderCfg_t eMPE1ParserCfg;
    GSW_CPU_ParserHeaderCfg_t eMPE2ParserCfg;
    GSW_CPU_ParserHeaderCfg_t eMPE1MPE2ParserCfg;
    GSW_FCS_TxOps_t bFcsTxOps;
    gsw_bool_t bTsPtp;
    gsw_bool_t bTsNonptp;
} GSW_CPU_PortCfg_t;

```

## Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting) set to CPU Port. The valid number is hardware dependent. (E.g. Port number 0 for GSWIP-3.0 or 6 for GSWIP-2.x). An error code is delivered if the selected port is not available.
gsw_bool_t	bCPU_PortValid	CPU port validity. Set command: set true to define a CPU port, set false to undo the setting. Get command: true if defined as CPU, false if not defined as CPU port.
gsw_bool_t	bSpecialTagIngress	Special tag enable in ingress direction.
gsw_bool_t	bSpecialTagEgress	Special tag enable in egress direction.
gsw_bool_t	bFcsCheck	Enable FCS check <ul style="list-style-type: none"> <li>false: No check, forward all frames</li> <li>1: Check FCS, drop frames with errors</li> </ul>
gsw_bool_t	bFcsGenerate	Enable FCS generation <ul style="list-style-type: none"> <li>false: Forward packets without FCS</li> <li>1: Generate FCS for all frames</li> </ul>
<a href="#">GSW_CPU_SpecialTagEthType_t</a>	bSpecialTagEthType	Special tag Ethertype mode. Not applicable to GSWIP-3.1.
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	eNoMPEParserCfg	GSWIP-3.0 specific Parser Header Config for no MPE flags (i.e. MPE1=0, MPE2=0).
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	eMPE1ParserCfg	GSWIP-3.0 specific Parser Header Config for MPE-1 set flag (i.e. MPE1=1, MPE2=0).
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	eMPE2ParserCfg	GSWIP-3.0 specific Parser Header Config for MPE-2 set flag (i.e. MPE1=0, MPE2=1).
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	eMPE1MPE2ParserCfg	GSWIP-3.0 specific Parser Header Config for both MPE-1 and MPE-2 set flag (i.e. MPE1=1, MPE2=1).
<a href="#">GSW_FCS_TxOps_t</a>	bFcsTxOps	GSWIP-3.1 FCS tx Operations.
gsw_bool_t	bTsPtp	GSWIP-3.2 Time Stamp Field Removal for PTP Packet 0 - DIS Removal is disabled 1 - EN Removal is enabled
gsw_bool_t	bTsNonptp	GSWIP-3.2 Time Stamp Field Removal for Non-PTP Packet 0 - DIS Removal is disabled 1 - EN Removal is enabled



### 1.10.1.26 GSW\_CTP\_portAssignment\_t

#### Description

CTP Port Assignment/association with logical port. Used by GSW\_CTP\_PORT\_ASSIGNMENT\_ALLOC, GSW\_CTP\_PORT\_ASSIGNMENT\_SET and GSW\_CTP\_PORT\_ASSIGNMENT\_GET.

#### Prototype

```
struct
{
    u8 nLogicalPortId;
    u16 nFirstCtpPortId;
    u16 nNumberOfCtpPort;
    GSW_LogicalPortMode_t eMode;
    u16 nBridgePortId;
} GSW_CTP_portAssignment_t;
```

#### Parameters

Data Type	Name	Description
u8	nLogicalPortId	Logical Port Id. The valid range is hardware dependent.
u16	nFirstCtpPortId	First CTP Port ID mapped to above logical port ID. <i>Note: For GSW_CTP_PORT_ASSIGNMENT_ALLOC, this is output when CTP allocation is successful. For other APIs, this is input.</i>
u16	nNumberOfCtpPort	Total number of CTP Ports mapped above logical port ID.
<a href="#">GSW_LogicalPortMode_t</a>	eMode	Logical port mode to define sub interface ID format.
u16	nBridgePortId	Bridge ID (FID) <i>Note: For GSW_CTP_PORT_ASSIGNMENT_ALLOC, this is input. Each CTP allocated is mapped to Bridge Port given by this field. The Bridge Port will be configured to use first CTP (<a href="#">GSW_CTP_portAssignment_t</a>) as egress CTP. For other APIs, this is ignored.</i>

### 1.10.1.27 GSW\_CTP\_portConfig\_t

#### Description

CTP Port Configuration. Used by GSW\_CTP\_PORT\_CONFIG\_SET and GSW\_CTP\_PORT\_CONFIG\_GET.

## Prototype

```
struct
{
    u8 nLogicalPortId;
    u16 nSubIfIdGroup;
    GSW_CtpPortConfigMask_t eMask;
    u16 nBridgePortId;
    gsw_bool_t bForcedTrafficClass;
    u8 nDefaultTrafficClass;
    gsw_bool_t bIngressExtendedVlanEnable;
    u16 nIngressExtendedVlanBlockId;
    u16 nIngressExtendedVlanBlockSize;
    gsw_bool_t bIngressExtendedVlanIgmpEnable;
    u16 nIngressExtendedVlanBlockIdIgmp;
    u16 nIngressExtendedVlanBlockSizeIgmp;
    gsw_bool_t bEgressExtendedVlanEnable;
    u16 nEgressExtendedVlanBlockId;
    u16 nEgressExtendedVlanBlockSize;
    gsw_bool_t bEgressExtendedVlanIgmpEnable;
    u16 nEgressExtendedVlanBlockIdIgmp;
    u16 nEgressExtendedVlanBlockSizeIgmp;
    gsw_bool_t bIngressNtolVlanEnable;
    gsw_bool_t bEgressNtolVlanEnable;
    GSW_ColorMarkingMode_t eIngressMarkingMode;
    GSW_ColorMarkingMode_t eEgressMarkingMode;
    gsw_bool_t bEgressMarkingOverrideEnable;
    GSW_ColorMarkingMode_t eEgressMarkingModeOverride;
    GSW_ColorRemarkingMode_t eEgressRemarkingMode;
    gsw_bool_t bIngressMeteringEnable;
    u16 nIngressTrafficMeterId;
    gsw_bool_t bEgressMeteringEnable;
    u16 nEgressTrafficMeterId;
    gsw_bool_t bBridgingBypass;
    u8 nDestLogicalPortId;
    gsw_bool_t bPmapperEnable;
    u16 nDestSubIfIdGroup;
    GSW_PmapperMappingMode_t ePmapperMappingMode;
    gsw_bool_t bPmapperIdValid;
    GSW_PMAPPER_t sPmapper;
    u16 nFirstFlowEntryIndex;
    u16 nNumberOfFlowEntries;
    gsw_bool_t bIngressLoopbackEnable;
    gsw_bool_t bIngressDaSaSwapEnable;
    gsw_bool_t bEgressLoopbackEnable;
    gsw_bool_t bEgressDaSaSwapEnable;
    gsw_bool_t bIngressMirrorEnable;
    gsw_bool_t bEgressMirrorEnable;
} GSW_CTP_portConfig_t;
```

## Parameters

Data Type	Name	Description
u8	nLogicalPortId	Logical Port Id. The valid range is hardware dependent. If <a href="#">GSW_CTP_portConfig_t</a> has <a href="#">GSW_CtpPortConfigMask_t</a> , this field is ignored.
u16	nSubIfIdGroup	Sub interface ID group. The valid range is hardware/protocol dependent. <i>Note: Sub interface ID group is defined for each of <a href="#">GSW_LogicalPortMode_t</a>. For both <a href="#">GSW_LOGICAL_PORT_8BIT_WLAN</a> and <a href="#">GSW_LOGICAL_PORT_9BIT_WLAN</a>, this field is VAP. For <a href="#">GSW_LOGICAL_PORT_GPON</a>, this field is GEM index. For <a href="#">GSW_LOGICAL_PORT_EPON</a>, this field is stream index. For <a href="#">GSW_LOGICAL_PORT_GINT</a>, this field is LLID. For others, this field is 0. If <a href="#">GSW_CTP_portConfig_t</a> has <a href="#">GSW_CtpPortConfigMask_t</a>, this field is absolute index of CTP in hardware for debug purpose, bypassing any check.</i>
<a href="#">GSW_CtpPortConfigMask_t</a>	eMask	Mask for updating/retrieving fields.
u16	nBridgePortId	Ingress Bridge Port ID to which this CTP port is associated for ingress traffic.
gsw_bool_t	bForcedTrafficClass	Default traffic class can not be overridden by other rules (except traffic flow table and special tag) in processing stages.
u8	nDefaultTrafficClass	Default traffic class associated with all ingress traffic from this CTP Port.
gsw_bool_t	bIngressExtendedVlanEnable	Enable Extended VLAN processing for ingress non-IGMP traffic.
u16	nIngressExtendedVlanBlockId	Extended VLAN block allocated for ingress non-IGMP traffic. It defines extended VLAN process for ingress non-IGMP traffic. Valid when bIngressExtendedVlanEnable is TRUE.
u16	nIngressExtendedVlanBlockSize	Extended VLAN block size for ingress non-IGMP traffic. This is optional. If it is 0, the block size of nIngressExtendedVlanBlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bIngressExtendedVlanIgmpEnable	Enable extended VLAN processing for ingress IGMP traffic.

Module Reference

Data Type	Name	Description
u16	nIngressExtendedVlanBlockIdIgmp	Extended VLAN block allocated for ingress IGMP traffic. It defines extended VLAN process for ingress IGMP traffic. Valid when bIngressExtendedVlanIgmpEnable is TRUE.
u16	nIngressExtendedVlanBlockSizeIgmp	Extended VLAN block size for ingress IGMP traffic. This is optional. If it is 0, the block size of nIngressExtendedVlanBlockIdIgmp will be used. Otherwise, this field will be used.
gsw_bool_t	bEgressExtendedVlanEnable	Enable extended VLAN processing for egress non-IGMP traffic.
u16	nEgressExtendedVlanBlockId	Extended VLAN block allocated for egress non-IGMP traffic. It defines extended VLAN process for egress non-IGMP traffic. Valid when bEgressExtendedVlanEnable is TRUE.
u16	nEgressExtendedVlanBlockSize	Extended VLAN block size for egress non-IGMP traffic. This is optional. If it is 0, the block size of nEgressExtendedVlanBlockId will be used. Otherwise, this field will be used.
gsw_bool_t	bEgressExtendedVlanIgmpEnable	Enable extended VLAN processing for egress IGMP traffic.
u16	nEgressExtendedVlanBlockIdIgmp	Extended VLAN block allocated for egress IGMP traffic. It defines extended VLAN process for egress IGMP traffic. Valid when bEgressExtendedVlanIgmpEnable is TRUE.
u16	nEgressExtendedVlanBlockSizeIgmp	Extended VLAN block size for egress IGMP traffic. This is optional. If it is 0, the block size of nEgressExtendedVlanBlockIdIgmp will be used. Otherwise, this field will be used.
gsw_bool_t	bIngressNto1VlanEnable	For WLAN type logical port, this should be FALSE. For other types, if enabled and ingress packet is VLAN tagged, outer VLAN ID is used for "nSubIfId" field in MAC table, otherwise, 0 is used for "nSubIfId".
gsw_bool_t	bEgressNto1VlanEnable	For WLAN type logical port, this should be FALSE. For other types, if enabled and egress packet is known unicast, outer VLAN ID is from "nSubIfId" field in MAC table.
<a href="#">GSW_ColorMarkingMode_t</a>	eIngressMarkingMode	Ingress color marking mode for ingress traffic.
<a href="#">GSW_ColorMarkingMode_t</a>	eEgressMarkingMode	Egress color marking mode for ingress traffic at egress priority queue color marking stage
gsw_bool_t	bEgressMarkingOverrideEnable	Egress color marking mode override color marking mode from last stage.
<a href="#">GSW_ColorMarkingMode_t</a>	eEgressMarkingModeOverride	Egress color marking mode for egress traffic. Valid only when bEgressMarkingOverride is TRUE.

## Module Reference

Data Type	Name	Description
<a href="#">GSW_ColorRemarkingMode_t</a>	eEgressRemarkingMode	Color remarking for egress traffic.
gsw_bool_t	bIngressMeteringEnable	Traffic metering on ingress traffic applies.
u16	nIngressTrafficMeterId	Meter for ingress CTP process. <i>Note: Meter should be allocated with GSW_QOS_METER_ALLOC before CTP port configuration. If this CTP port is re-set, the last used meter should be released.</i>
gsw_bool_t	bEgressMeteringEnable	Traffic metering on egress traffic applies.
u16	nEgressTrafficMeterId	Meter for egress CTP process. <i>Note: Meter should be allocated with GSW_QOS_METER_ALLOC before CTP port configuration. If this CTP port is re-set, the last used meter should be released.</i>
gsw_bool_t	bBridgingBypass	Ingress traffic bypass bridging/multicast processing. Following parameters are used to determine destination. Traffic flow table is not bypassed.
u8	nDestLogicalPortId	When bBridgingBypass is TRUE, this field defines destination logical port.
gsw_bool_t	bPmapperEnable	When bBridgingBypass is TRUE, this field indicates whether to use <a href="#">GSW_CTP_portConfig_t</a> or <a href="#">GSW_CTP_portConfig_t/</a> <a href="#">GSW_CTP_portConfig_t</a> .
u16	nDestSubIfIdGroup	When bBridgingBypass is TRUE and bPmapperEnable is FALSE, this field defines destination sub interface ID group.
<a href="#">GSW_PmapperMappingMode_t</a>	ePmapperMappingMode	When bBridgingBypass is TRUE and bPmapperEnable is TRUE, this field selects either DSCP or PCP to derive sub interface ID.
gsw_bool_t	bPmapperIdValid	When bPmapperEnable is TRUE, P-mapper is used. This field determines whether sPmapper.nPmapperId is valid. If this field is TRUE, the P-mapper is re-used and no allocation of new P-mapper or value change in the P-mapper. If this field is FALSE, allocation is taken care in the API implementation.

## Module Reference

Data Type	Name	Description
<b>GSW_PMAPPER_t</b>	sPmapper	When bBridgingBypass is TRUE and bPmapperEnable is TRUE, P-mapper is used. If bPmapperIdValid is FALSE, API implementation need take care of P-mapper allocation, and maintain the reference counter of P-mapper used multiple times. If bPmapperIdValid is TRUE, only sPmapper.nPmapperId is used to associate the P-mapper, and there is no allocation of new P-mapper or value change in the P-mapper.
u16	nFirstFlowEntryIndex	First traffic flow table entry is associated to this CTP port. Ingress traffic from this CTP port will go through traffic flow table search starting from nFirstFlowEntryIndex. Should be times of 4.
u16	nNumberOfFlowEntries	Number of traffic flow table entries are associated to this CTP port. Ingress traffic from this CTP port will go through PCE rules search ending at (nFirstFlowEntryIndex+nNumberOfFlowEntries)-1. Should be times of 4.
gsw_bool_t	bIngressLoopbackEnable	Ingress traffic from this CTP port will be redirected to ingress logical port of this CTP port with source sub interface ID used as destination sub interface ID. Following processing except traffic flow table search is bypassed if loopback enabled.
gsw_bool_t	bIngressDaSaSwapEnable	Destination/Source MAC address of ingress traffic is swapped before transmitted (not swapped during PCE processing stages). If destination is multicast, there is no swap, but source MAC address is replaced with global configurable value.
gsw_bool_t	bEgressLoopbackEnable	Egress traffic to this CTP port will be redirected to ingress logical port with same sub interface ID as ingress.
gsw_bool_t	bEgressDaSaSwapEnable	Destination/Source MAC address of egress traffic is swapped before transmitted.
gsw_bool_t	bIngressMirrorEnable	If enabled, ingress traffic is mirrored to the monitoring port. <i>Note: This should be used exclusive with bIngressLoopbackEnable.</i>
gsw_bool_t	bEgressMirrorEnable	If enabled, egress traffic is mirrored to the monitoring port. <i>Note: This should be used exclusive with bEgressLoopbackEnable.</i>

### 1.10.1.28 GSW\_Debug\_RMON\_Port\_cnt\_t

#### Description

For debugging Purpose only. Used for GSWIP 3.1.

#### Prototype

```
struct
{
    u16 nPortId;
    ePortType;
    gsw_bool_t b64BitMode;
    u32 nRxExtendedVlanDiscardPkts;
    u32 nMtuExceedDiscardPkts;
    u32 nTxUnderSizeGoodPkts;
    u32 nTxOversizeGoodPkts;
    u32 nRxGoodPkts;
    u32 nRxUnicastPkts;
    u32 nRxBroadcastPkts;
    u32 nRxMulticastPkts;
    u32 nRxFCSErrorPkts;
    u32 nRxUnderSizeGoodPkts;
    u32 nRxOversizeGoodPkts;
    u32 nRxUnderSizeErrorPkts;
    u32 nRxGoodPausePkts;
    u32 nRxOversizeErrorPkts;
    u32 nRxAlignErrorPkts;
    u32 nRxFilteredPkts;
    u32 nRx64BytePkts;
    u32 nRx127BytePkts;
    u32 nRx255BytePkts;
    u32 nRx511BytePkts;
    u32 nRx1023BytePkts;
    u32 nRxMaxBytePkts;
    u32 nTxGoodPkts;
    u32 nTxUnicastPkts;
    u32 nTxBroadcastPkts;
    u32 nTxMulticastPkts;
    u32 nTxSingleCollCount;
    u32 nTxMultCollCount;
    u32 nTxLateCollCount;
    u32 nTxExcessCollCount;
    u32 nTxCollCount;
    u32 nTxPauseCount;
    u32 nTx64BytePkts;
    u32 nTx127BytePkts;
    u32 nTx255BytePkts;
    u32 nTx511BytePkts;
    u32 nTx1023BytePkts;
    u32 nTxMaxBytePkts;
    u32 nTxDroppedPkts;
```

```

u32 nTxAcmdroppedPkts;
u32 nRxDroppedPkts;
u64 nRxGoodBytes;
u64 nRxBadBytes;
u64 nTxGoodBytes;
u32 nRxUnicastPktsYellowRed;
u32 nRxBroadcastPktsYellowRed;
u32 nRxMulticastPktsYellowRed;
u64 nRxGoodBytesYellowRed;
u32 nRxGoodPktsYellowRed;
u32 nTxUnicastPktsYellowRed;
u32 nTxBroadcastPktsYellowRed;
u32 nTxMulticastPktsYellowRed;
u64 nTxGoodBytesYellowRed;
u32 nTxGoodPktsYellowRed;
} GSW_Debug_RMON_Port_cnt_t;

```

## Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available. This parameter specifies for which MAC port the RMON counter is read. It has to be set by the application before calling GSW_RMON_PORT_GET.
	ePortType	Table address selection based on port type Applicable only for GSWIP 3.1
gsw_bool_t	b64BitMode	
u32	nRxExtendedVlanDiscardPkts	
u32	nMtuExceedDiscardPkts	
u32	nTxUnderSizeGoodPkts	
u32	nTxOversizeGoodPkts	
u32	nRxGoodPkts	Receive Packet Count (only packets that are accepted and not discarded).
u32	nRxUnicastPkts	Receive Unicast Packet Count.
u32	nRxBroadcastPkts	Receive Broadcast Packet Count.
u32	nRxMulticastPkts	Receive Multicast Packet Count.
u32	nRxFCSErrorPkts	Receive FCS Error Packet Count.
u32	nRxUnderSizeGoodPkts	Receive Undersize Good Packet Count.
u32	nRxOversizeGoodPkts	Receive Oversize Good Packet Count.
u32	nRxUnderSizeErrorPkts	Receive Undersize Error Packet Count.
u32	nRxGoodPausePkts	Receive Good Pause Packet Count.
u32	nRxOversizeErrorPkts	Receive Oversize Error Packet Count.
u32	nRxAlignErrorPkts	Receive Align Error Packet Count.



Module Reference

Data Type	Name	Description
u32	nRxFilteredPkts	Filtered Packet Count.
u32	nRx64BytePkts	Receive Size 64 Bytes Packet Count.
u32	nRx127BytePkts	Receive Size 65-127 Bytes Packet Count.
u32	nRx255BytePkts	Receive Size 128-255 Bytes Packet Count.
u32	nRx511BytePkts	Receive Size 256-511 Bytes Packet Count.
u32	nRx1023BytePkts	Receive Size 512-1023 Bytes Packet Count.
u32	nRxMaxBytePkts	Receive Size 1024-1522 Bytes (or more, if configured) Packet Count.
u32	nTxGoodPkts	Overall Transmit Good Packets Count.
u32	nTxUnicastPkts	Transmit Unicast Packet Count.
u32	nTxBroadcastPkts	Transmit Broadcast Packet Count.
u32	nTxMulticastPkts	Transmit Multicast Packet Count.
u32	nTxSingleCollCount	Transmit Single Collision Count.
u32	nTxMultCollCount	Transmit Multiple Collision Count.
u32	nTxLateCollCount	Transmit Late Collision Count.
u32	nTxExcessCollCount	Transmit Excessive Collision Count.
u32	nTxCollCount	Transmit Collision Count.
u32	nTxPauseCount	Transmit Pause Packet Count.
u32	nTx64BytePkts	Transmit Size 64 Bytes Packet Count.
u32	nTx127BytePkts	Transmit Size 65-127 Bytes Packet Count.
u32	nTx255BytePkts	Transmit Size 128-255 Bytes Packet Count.
u32	nTx511BytePkts	Transmit Size 256-511 Bytes Packet Count.
u32	nTx1023BytePkts	Transmit Size 512-1023 Bytes Packet Count.
u32	nTxMaxBytePkts	Transmit Size 1024-1522 Bytes (or more, if configured) Packet Count.
u32	nTxDroppedPkts	Transmit Drop Packet Count.
u32	nTxAcuDroppedPkts	Transmit Dropped Packet Count, based on Congestion Management.
u32	nRxDroppedPkts	Receive Dropped Packet Count.
u64	nRxGoodBytes	Receive Good Byte Count (64 bit).
u64	nRxBadBytes	Receive Bad Byte Count (64 bit).
u64	nTxGoodBytes	Transmit Good Byte Count (64 bit).
u32	nRxUnicastPktsYellowRed	For GSWIP V32 only Receive Unicast Packet Count for Yellow & Red packet.
u32	nRxBroadcastPktsYellowRed	Receive Broadcast Packet Count for Yellow & Red packet.
u32	nRxMulticastPktsYellowRed	Receive Multicast Packet Count for Yellow & Red packet.
u64	nRxGoodBytesYellowRed	Receive Good Byte Count (64 bit) for Yellow & Red packet.
u32	nRxGoodPktsYellowRed	Receive Packet Count for Yellow & Red packet.

Data Type	Name	Description
u32	nTxUnicastPktsYellowRed	Transmit Unicast Packet Count for Yellow & Red packet.
u32	nTxBroadcastPktsYellowRed	Transmit Broadcast Packet Count for Yellow & Red packet.
u32	nTxMulticastPktsYellowRed	Transmit Multicast Packet Count for Yellow & Red packet.
u64	nTxGoodBytesYellowRed	Transmit Good Byte Count (64 bit) for Yellow & Red packet.
u32	nTxGoodPktsYellowRed	Transmit Packet Count for Yellow & Red packet.

### 1.10.1.29 GSW\_debug\_t

#### Description

For debugging Purpose only. Used for GSWIP 3.3.

#### Prototype

```
struct
{
    u16 nTableIndex;
    u8 nForceSet;
    gsw_bool_t nCheckIndexInUse;
    u16 nblockid;
    u8 nDiscardUntagged;
    u8 nDiscardUnMatchedTagged;
    u8 nPmacId;
    u8 nDestPort;
} GSW_debug_t;
```

#### Parameters

Data Type	Name	Description
u16	nTableIndex	Table Index to get status of the Table Index only for GSWIP 3.1
u8	nForceSet	Force table Index In USE only for GSWIP 3.1
gsw_bool_t	nCheckIndexInUse	To check display index which are In USE for GSWIP 3.1
u16	nblockid	Vlan Filter or Exvlan BlockID
u8	nDiscardUntagged	Vlan Filter debugging usage
u8	nDiscardUnMatchedTagged	Vlan Filter debugging usage
u8	nPmacId	Pmac debugging purpose
u8	nDestPort	Pmac debugging purpose

### 1.10.1.30 GSW\_DSCP2PCP\_map\_t

#### Description

DSCP to PCP Mapping. Used by GSW\_DSCP2PCP\_MAP\_GET.

#### Prototype

```
struct
{
    u16 nIndex;
    u8 nMap[64];
} GSW_DSCP2PCP_map_t;
```

#### Parameters

Data Type	Name	Description
u16	nIndex	Index of entry in mapping table.
u8	nMap[64]	The index of array stands for DSCP value. Each byte of the array is 3-bit PCP value.

### 1.10.1.31 GSW\_I\_TAG\_Config\_t

#### Description

I-TAG header definition .GSWIP-3.2 only Used by [GSW\\_PBB\\_Tunnel\\_Template\\_Config\\_t](#).

#### Prototype

```
struct
{
    gsw_bool_t bTpidEnable;
    u16 nTpid;
    gsw_bool_t bPcpEnable;
    u8 nPcp;
    gsw_bool_t bDeiEnable;
    u8 nDei;
    gsw_bool_t bUacEnable;
    u8 nUac;
    gsw_bool_t bResEnable;
    u8 nRes;
    gsw_bool_t bSidEnable;
    u32 nSid;
} GSW_I_TAG_Config_t;
```

#### Parameters

Data Type	Name	Description
gsw_bool_t	bTpidEnable	I-TAG TPID -2 bytes field
u16	nTpid	
gsw_bool_t	bPcpEnable	I-TAG PCP -3 Bit field

Data Type	Name	Description
u8	nPcp	
gsw_bool_t	bDeiEnable	I-TAG DEI -1 Bit field
u8	nDei	
gsw_bool_t	bUacEnable	I-TAG UAC -1 Bit field
u8	nUac	
gsw_bool_t	bResEnable	I-TAG RES -3 Bit field
u8	nRes	
gsw_bool_t	bSidEnable	I-TAG SID -24 Bit field
u32	nSid	

### 1.10.1.32 GSW\_MAC\_tableAdd\_t

#### Description

MAC Table Entry to be added. Used by GSW\_MAC\_TABLE\_ENTRY\_ADD.

#### Prototype

```
struct
{
    u16 nFid;
    u32 nPortId;
    u16 nPortMap[8];
    u16 nSubIfId;
    int nAgeTimer;
    u16 nSVLAN_Id;
    gsw_bool_t bStaticEntry;
    u8 nTrafficClass;
    u8 nMAC[GSW_MAC_ADDR_LEN];
    u8 nFilterFlag;
    gsw_bool_t bIcmpControlled;
    u8 nAssociatedMAC[GSW_MAC_ADDR_LEN];
    u16 nTci;
} GSW_MAC_tableAdd_t;
```

## Parameters

Data Type	Name	Description
u16	nFId	Filtering Identifier (FID) (not supported by all switches)
u32	nPortId	<p>Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge Port ID. The valid range is hardware dependent.</p> <p><i>Note: In GSWIP-2.1/2.2/3.0, this field is used as portmap field, when the MSB bit is set. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port. The macro <a href="#">GSW_PORTMAP_FLAG_SET</a> allows to set the MSB bit, marking it as portmap variable. Checking the portmap flag can be done by using the <a href="#">GSW_PORTMAP_FLAG_GET</a> macro. From GSWIP3.1, if MSB is set, other bits in this field are ignored. array <a href="#">GSW_MAC_tableRead_t</a> is used for bit map.</i></p>
u16	nPortMap[8]	Bridge Port Map - to support GSWIP-3.1, following field is added for port map in static entry. It's valid only when MSB of <a href="#">GSW_MAC_tableRead_t</a> is set. Each bit stands for 1 bridge port.
u16	nSubIfId	<p>Sub-Interface Identifier Destination (supported in GSWIP-3.0/3.1 only).</p> <p><i>Note: In GSWIP-3.1, this field is sub interface ID for WLAN logical port. For Other types, either outer VLAN ID if Nto1Vlan enabled or 0.</i></p>
int	nAgeTimer	Aging Time, given in multiples of 1 second in a range from 1 s to 1,000,000 s. The configured value might be rounded that it fits to the given hardware platform.
u16	nSVLAN_Id	STAG VLAN Id. Only applicable in case SVLAN support is enabled on the device.
gsw_bool_t	bStaticEntry	Static Entry (value will be aged out if the entry is not set to static). The switch API implementation uses the maximum age timer in case the entry is not static.
u8	nTrafficClass	Egress queue traffic class. The queue index starts counting from zero.

Data Type	Name	Description
u8	nMAC[GSW_MAC_ADDR_LEN]	MAC Address to add to the table.
u8	nFilterFlag	Source/Destination MAC address filtering flag (GSWIP-3.1 only) Value 0 - not filter, 1 - source address filter, 2 - destination address filter, 3 - both source and destination filter. <i>Note: Please refer to "GSWIP Hardware Architecture Spec" chapter 3.4.4.6 "Source MAC Address Filtering and Destination MAC Address Filtering" for more detail.</i>
gsw_bool_t	blgmpControlled	Packet is marked as IGMP controlled if destination MAC address matches MAC in this entry. (GSWIP-3.1 only)
u8	nAssociatedMAC[GSW_MAC_ADDR_LEN]	Associated Mac address -(GSWIP-3.2)
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI

### 1.10.1.33 GSW\_MAC\_tableClearCond\_t

#### Description

MAC Table Clear based on given condition. Used by GSW\_MAC\_TABLE\_CLEAR\_COND.

#### Prototype

```
struct
{
    u8 eType;
    u8 nPortId;
    union GSW_MAC_tableClearCond_t::@3 @4;
} GSW_MAC_tableClearCond_t;
```

#### Parameters

Data Type	Name	Description
u8	eType	MAC table clear type <a href="#">GSW_MacClearType_t</a>
u8	nPortId	Physical port id (0~16) if is ref GSW_MAC_CLEAR_PHY_PORT.
union GSW_MAC_tableClearCond_t: :@3	@4	

### 1.10.1.34 GSW\_MAC\_tableQuery\_t

#### Description

Search for a MAC address entry in the address table. Used by GSW\_MAC\_TABLE\_ENTRY\_QUERY.

#### Prototype

```
struct
{
    u8 nMAC[GSW_MAC_ADDR_LEN];
    u16 nFId;
    gsw_bool_t bFound;
    u32 nPortId;
    u16 nPortMap[8];
    u16 nSubIfId;
    int nAgeTimer;
    u16 nSVLAN_Id;
    gsw_bool_t bStaticEntry;
    u8 nFilterFlag;
    gsw_bool_t bIcmpControlled;
    gsw_bool_t bEntryChanged;
    u8 nAssociatedMAC[GSW_MAC_ADDR_LEN];
    gsw_bool_t hitstatus;
    u16 nTci;
    u16 nFirstBridgePortId;
} GSW_MAC_tableQuery_t;
```

#### Parameters

Data Type	Name	Description
u8	nMAC[GSW_MAC_ADDR_LEN]	MAC Address. This parameter needs to be provided for the search operation. This is an input parameter.
u16	nFId	Get the MAC table entry belonging to the given Filtering Identifier (not supported by all switches). This is an input parameter.
gsw_bool_t	bFound	MAC Address Found. Switch API sets this boolean variable in case the requested MAC address 'nMAC' is found inside the address table, otherwise it is set to FALSE. This is an output parameter.

## Module Reference

Data Type	Name	Description
u32	nPortId	<p>Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge port ID. The valid range is hardware dependent.</p> <p><i>Note: In GSWIP-2.1/2.2/3.0, this field is used as portmap field, when the MSB bit is set. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port. The macro <a href="#">GSW_PORTMAP_FLAG_SET</a> allows to set the MSB bit, marking it as portmap variable. Checking the portmap flag can be done by using the <a href="#">GSW_PORTMAP_FLAG_GET</a> macro. From GSWIP3.1, if MSB is set, other bits in this field are ignored. array <a href="#">GSW_MAC_tableRead_t</a> is used for bit map.</i></p>
u16	nPortMap[8]	<p>Bridge Port Map - to support GSWIP-3.1, following field is added for port map in static entry. It's valid only when MSB of <a href="#">GSW_MAC_tableRead_t</a> is set. Each bit stands for 1 bridge port.</p>
u16	nSubIfId	<p>Sub-Interface Identifier Destination (supported in GSWIP-3.0/3.1 only).</p>
int	nAgeTimer	<p>Aging Time, given in multiples of 1 second in a range from 1 s to 1,000,000 s. The value read back in a GET command might differ slightly from the value given in the SET command due to limited hardware timing resolution. Filled out by the switch API implementation. This is an output parameter.</p>
u16	nSVLAN_Id	<p>STAG VLAN Id. Only applicable in case SVLAN support is enabled on the device.</p>
gsw_bool_t	bStaticEntry	<p>Static Entry (value will be aged out after 'nAgeTimer' if the entry is not set to static). This is an output parameter.</p>
u8	nFilterFlag	<p>Source/Destination MAC address filtering flag (GSWIP-3.1 only) Value 0 - not filter, 1 - source address filter, 2 - destination address filter, 3 - both source and destination filter.</p> <p><i>Note: Please refer to "GSWIP Hardware Architecture Spec" chapter 3.4.4.6 "Source MAC Address Filtering and Destination MAC Address Filtering" for more detail.</i></p>



Data Type	Name	Description
gsw_bool_t	blgmpControlled	Packet is marked as IGMP controlled if destination MAC address matches MAC in this entry. (GSWIP-3.1 only)
gsw_bool_t	bEntryChanged	Changed 0: the entry is not changed 1: the entry is changed and not accessed yet
u8	nAssociatedMAC[GSW_MAC_ADDR_LEN]	Associated Mac address -(GSWIP-3.2)
gsw_bool_t	hitstatus	
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI
u16	nFirstBridgePortId	first bridge port ID (supported in GSWIP-3.3only)

### 1.10.1.35 GSW\_MAC\_tableRead\_t

#### Description

MAC Table Entry to be read. Used by GSW\_MAC\_TABLE\_ENTRY\_READ.

#### Prototype

```
struct
{
    gsw_bool_t bInitial;
    gsw_bool_t bLast;
    u16 nFid;
    u32 nPortId;
    u16 nPortMap[8];
    int nAgeTimer;
    u16 nSVLAN_Id;
    gsw_bool_t bStaticEntry;
    u16 nSubIfId;
    u8 nMAC[GSW_MAC_ADDR_LEN];
    u8 nFilterFlag;
    gsw_bool_t bIgmpControlled;
    gsw_bool_t bEntryChanged;
    u8 nAssociatedMAC[GSW_MAC_ADDR_LEN];
    gsw_bool_t hitstatus;
    u16 nTci;
    u16 nFirstBridgePortId;
} GSW_MAC_tableRead_t;
```

## Parameters

Data Type	Name	Description
gsw_bool_t	bInitial	Restart the get operation from the beginning of the table. Otherwise return the next table entry (next to the entry that was returned during the previous get operation). This boolean parameter is set by the calling application.
gsw_bool_t	bLast	Indicates that the read operation got all last valid entries of the table. This boolean parameter is set by the switch API when the Switch API is called after the last valid one was returned already.
u16	nFId	Get the MAC table entry belonging to the given Filtering Identifier (not supported by all switches).
u32	nPortId	<p>Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge Port ID. The valid range is hardware dependent.</p> <p><i>Note: In GSWIP-2.1/2.2/3.0, this field is used as portmap field, when the MSB bit is set. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port. The macro <a href="#">GSW_PORTMAP_FLAG_SET</a> allows to set the MSB bit, marking it as portmap variable. Checking the portmap flag can be done by using the <a href="#">GSW_PORTMAP_FLAG_GET</a> macro. From GSWIP3.1, if MSB is set, other bits in this field are ignored. array <a href="#">GSW_MAC_tableRead_t</a> is used for bit map.</i></p>
u16	nPortMap[8]	Bridge Port Map - to support GSWIP-3.1, following field is added for port map in static entry. It's valid only when MSB of <a href="#">GSW_MAC_tableRead_t</a> is set. Each bit stands for 1 bridge port.
int	nAgeTimer	Aging Time, given in multiples of 1 second in a range from 1 s to 1,000,000 s. The value read back in a GET command might differ slightly from the value given in the SET command due to limited hardware timing resolution. Filled out by the switch API implementation.
u16	nSVLAN_Id	STAG VLAN Id. Only applicable in case SVLAN support is enabled on the device.

## Module Reference

Data Type	Name	Description
gsw_bool_t	bStaticEntry	Static Entry (value will be aged out after 'nAgeTimer' if the entry is not set to static).
u16	nSubIfId	Sub-Interface Identifier Destination (supported in GSWIP-3.0/3.1 only).
u8	nMAC[GSW_MAC_ADDR_LEN]	MAC Address. Filled out by the switch API implementation.
u8	nFilterFlag	Source/Destination MAC address filtering flag (GSWIP-3.1 only) Value 0 - not filter, 1 - source address filter, 2 - destination address filter, 3 - both source and destination filter. <i>Note: Please refer to "GSWIP Hardware Architecture Spec" chapter 3.4.4.6 "Source MAC Address Filtering and Destination MAC Address Filtering" for more detail.</i>
gsw_bool_t	blgmpControlled	Packet is marked as IGMP controlled if destination MAC address matches MAC in this entry. (GSWIP-3.1 only)
gsw_bool_t	bEntryChanged	Changed 0: the entry is not changed 1: the entry is changed and not accessed yet
u8	nAssociatedMAC[GSW_MAC_ADDR_LEN]	Associated Mac address -(GSWIP-3.2)
gsw_bool_t	hitstatus	
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI
u16	nFirstBridgePortId	

### 1.10.1.36 GSW\_MAC\_tableRemove\_t

#### Description

MAC Table Entry to be removed. Used by GSW\_MAC\_TABLE\_ENTRY\_REMOVE.

#### Prototype

```
struct
{
    u16 nFid;
    u8 nMAC[GSW_MAC_ADDR_LEN];
    u8 nFilterFlag;
    u16 nTci;
} GSW_MAC_tableRemove_t;
```

## Parameters

Data Type	Name	Description
u16	nFid	Filtering Identifier (FID) (not supported by all switches)
u8	nMAC[GSW_MAC_ADDR_LEN]	MAC Address to be removed from the table.
u8	nFilterFlag	Source/Destination MAC address filtering flag (GSWIP-3.1 only) Value 0 - not filter, 1 - source address filter, 2 - destination address filter, 3 - both source and destination filter. <i>Note: Please refer to "GSWIP Hardware Architecture Spec" chapter 3.4.4.6 "Source MAC Address Filtering and Destination MAC Address Filtering" for more detail.</i>
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI

### 1.10.1.37 GSW\_MACFILTER\_default\_t

#### Description

Default MAC Address Filter. Used by GSW\_DEFAULT\_MAC\_FILTER\_SET and GSW\_DEFAULT\_MAC\_FILTER\_GET.

#### Prototype

```
struct
{
    GSW_MacFilterType_t eType;
    u16 nPortmap[8];
} GSW_MACFILTER_default_t;
```

#### Parameters

Data Type	Name	Description
<a href="#">GSW_MacFilterType_t</a>	eType	MAC Filter Type
u16	nPortmap[8]	Destination bridge port map. For GSWIP-3.1 only. <i>Note: Each bit stands for 1 bridge port. For PRX300 (GSWIP-3.1 integrated), only index 0-7 is valid.</i>

### 1.10.1.38 GSW\_monitorPortCfg\_t

#### Description

Port monitor configuration. Used by GSW\_MONITOR\_PORT\_CFG\_GET and GSW\_MONITOR\_PORT\_CFG\_SET.

#### Prototype

```
struct
{
    u8 nPortId;
    u16 nSubIfId;
    gsw_bool_t bMonitorPort;
} GSW_monitorPortCfg_t;
```

#### Parameters

Data Type	Name	Description
u8	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.
u16	nSubIfId	
gsw_bool_t	bMonitorPort	

### 1.10.1.39 GSW\_multicastRouter\_t

#### Description

Add an Ethernet port as router port to the switch hardware multicast table. Used by GSW\_MULTICAST\_ROUTER\_PORT\_ADD and GSW\_MULTICAST\_ROUTER\_PORT\_REMOVE.

#### Prototype

```
struct
{
    u16 nPortId;
} GSW_multicastRouter_t;
```

## Parameters

Data Type	Name	Description
u16	nPortId	<p>Bridge Port ID. The valid range is hardware dependent. An error code is delivered if the selected port is not available.</p> <p><i>Note: This field is used as portmap field, when the MSB bit is set. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port. The macro <a href="#">GSW_PORTMAP_FLAG_SET</a> allows to set the MSB bit, marking it as portmap variable. Checking the portmap flag can be done by using the <a href="#">GSW_PORTMAP_FLAG_GET</a> macro.</i></p>

### 1.10.1.40 GSW\_multicastRouterRead\_t

#### Description

Check if a port has been selected as a router port. Used by GSW\_MULTICAST\_ROUTER\_PORT\_READ. Not applicable to GSWIP-3.1.

#### Prototype

```
struct
{
    gsw_bool_t bInitial;
    gsw_bool_t bLast;
    u16 nPortId;
} GSW_multicastRouterRead_t;
```

## Parameters

Data Type	Name	Description
gsw_bool_t	bInitial	Restart the get operation from the start of the table. Otherwise return the next table entry (next to the entry that was returned during the previous get operation). This parameter is always reset during the read operation. This boolean parameter is set by the calling application.

Data Type	Name	Description
gsw_bool_t	bLast	Indicates that the read operation got all last valid entries of the table. This boolean parameter is set by the switch API when the Switch API is called after the last valid one was returned already.
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.

### 1.10.1.41 GSW\_multicastSnoopCfg\_t

#### Description

Configure the switch multicast configuration. Used by GSW\_MULTICAST\_SNOOP\_CFG\_SET and GSW\_MULTICAST\_SNOOP\_CFG\_GET.

#### Prototype

```
struct
{
    GSW_multicastSnoopMode_t eIGMP_Mode;
    gsw_bool_t bCrossVLAN;
    GSW_portForward_t eForwardPort;
    u8 nForwardPortId;
    u8 nClassOfService;
} GSW_multicastSnoopCfg_t;
```

#### Parameters

Data Type	Name	Description
<a href="#">GSW_multicastSnoopMode_t</a>	eIGMP_Mode	Enables and configures the IGMP/MLD snooping feature. Select autolearning or management packet forwarding mode. Packet forwarding is done to the port selected in 'eForwardPort'.
gsw_bool_t	bCrossVLAN	Enables snooped IGMP control packets treated as cross-CTAG VLAN packets. This parameter is used for hardware auto-learning and snooping packets forwarded to a dedicated port. This dedicated port can be selected over 'eForwardPort'.
<a href="#">GSW_portForward_t</a>	eForwardPort	Forward snooped packet, only used if forwarded mode is selected by 'eIGMP_Mode = GSW_MULTICAST_SNOOP_MODE_SNOOP_FORWARD'.

Data Type	Name	Description
u8	nForwardPortId	Target Bridge Port ID for forwarded packets, only used if selected by 'eForwardPort = GSW_PORT_FORWARD_PORT'.
u8	nClassOfService	Snooping control class of service. Snooping control packet can be forwarded to the 'nForwardPortId' when selected in 'eIGMP_Mode'. The class of service of this port can be selected for the snooped control packets, starting from zero. The maximum possible service class depends on the hardware platform used. The value GSW_TRAFFIC_CLASS_DISABLE disables overwriting the given class assignment.

### 1.10.1.42 GSW\_multicastTable\_t

#### Description

Add a host as a member to a multicast group. Used by GSW\_MULTICAST\_TABLE\_ENTRY\_ADD and GSW\_MULTICAST\_TABLE\_ENTRY\_REMOVE.

#### Prototype

```
struct
{
    u32 nPortId;
    u16 nSubIfId;
    GSW_IP_Select_t eIPVersion;
    GSW_IP_t uIP_Gda;
    GSW_IP_t uIP_Gsa;
    u8 nFID;
    gsw_bool_t bExclSrcIP;
    GSW_IGMP_MemberMode_t eModeMember;
    u16 nTci;
} GSW_multicastTable_t;
```

#### Parameters

Data Type	Name	Description
u32	nPortId	Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge Port ID. The valid range is hardware dependent. An error code is delivered if the selected port is not available.
u16	nSubIfId	Sub-Interface Id - valid for GSWIP 3.0/3.1 only
GSW_IP_Select_t	eIPVersion	Select the IP version of the 'uIP_Gda' and 'uIP_Gsa' fields. Both fields support either IPv4 or IPv6.
<a href="#">GSW_IP_t</a>	uIP_Gda	Group Destination IP address (GDA).



Data Type	Name	Description
<b>GSW_IP_t</b>	uIP_Gsa	Group Source IP address. Only used in case IGMPv3 support is enabled and 'eModeMember != GSW_IGMP_MEMBER_DONT_CARE'.
u8	nFID	FID - valid for GSWIP 3.0 only subject to Global FID for MC is enabled. always valid in GSWIP-3.1.
gsw_bool_t	bExclSrcIP	Exclude Mode - valid for GSWIP 3.0 only - Includes or Excludes Source IP - uIP_Gsa
<b>GSW_IGMP_MemberMode_t</b>	eModeMember	Group member filter mode. This is valid for GSWIP-3.0/3.1 to replaces bExclSrcIP. This parameter is ignored when deleting a multicast membership table entry. The configurations 'GSW_IGMP_MEMBER_EXCLUDE' and 'GSW_IGMP_MEMBER_INCLUDE' are only supported if IGMPv3 is used.
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI

### 1.10.1.43 GSW\_multicastTableRead\_t

#### Description

Read out the multicast membership table. Used by GSW\_MULTICAST\_TABLE\_ENTRY\_READ.

#### Prototype

```
struct
{
    gsw_bool_t bInitial;
    gsw_bool_t bLast;
    u32 nPortId;
    u16 nPortMap[8];
    u16 nSubIfId;
    GSW_IP_Select_t eIPVersion;
    GSW_IP_t uIP_Gda;
    GSW_IP_t uIP_Gsa;
    u8 nFID;
    gsw_bool_t bExclSrcIP;
    GSW_IGMP_MemberMode_t eModeMember;
    gsw_bool_t hitstatus;
    u16 nTci;
} GSW_multicastTableRead_t;
```

## Parameters

Data Type	Name	Description
gsw_bool_t	bInitial	Restart the get operation from the beginning of the table. Otherwise return the next table entry (next to the entry that was returned during the previous get operation). This parameter is always reset during the read operation. This boolean parameter is set by the calling application.
gsw_bool_t	bLast	Indicates that the read operation got all last valid entries of the table. This boolean parameter is set by the switch API when the Switch API is called after the last valid one was returned already.
u32	nPortId	Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge Port ID. The valid range is hardware dependent. An error code is delivered if the selected port is not available.  <i>Note: This field is used as portmap field, when the MSB bit is set. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port. The macro <a href="#">GSW_PORTMAP_FLAG_SET</a> allows to set the MSB bit, marking it as portmap variable. Checking the portmap flag can be done by using the <a href="#">GSW_PORTMAP_FLAG_GET</a> macro.</i>
u16	nPortMap[8]	Ethernet Port Map - to support GSWIP-3.1, following field is added for port map in static entry. It's valid only when MSB of nPortId is set. Each bit stands for 1 bridge port.
u16	nSubIfId	Sub-Interface Id - valid for GSWIP 3.0 only
GSW_IP_Select_t	eIPVersion	Select the IP version of the 'uIP_Gda' and 'uIP_Gsa' fields. Both fields support either IPv4 or IPv6.
<a href="#">GSW_IP_t</a>	uIP_Gda	Group Destination IP address (GDA).
<a href="#">GSW_IP_t</a>	uIP_Gsa	Group Source IP address. Only used in case IGMPv3 support is enabled.
u8	nFID	FID - valid for GSWIP 3.0 only subject to Global FID for MC is enabled
gsw_bool_t	bExclSrcIP	Exclude Mode - valid for GSWIP 3.0 only - Includes or Excludes Source IP - uIP_Gsa

Data Type	Name	Description
<a href="#">GSW_IGMP_MemberMode_t</a>	eModeMember	Group member filter mode. This parameter is ignored when deleting a multicast membership table entry. The configurations 'GSW_IGMP_MEMBER_EXCLUDE' and 'GSW_IGMP_MEMBER_INCLUDE' are only supported if IGMPv3 is used.
gsw_bool_t	hitstatus	
u16	nTci	TCI for (GSWIP-3.2) B-Step Bit [0:11] - VLAN ID Bit [12] - VLAN CFI/DEI Bit [13:15] - VLAN PRI

#### 1.10.1.44 GSW\_PBB\_Tunnel\_Template\_Config\_t

##### Description

Tunnel Template Configuration.GSWIP-3.2 only Used by GSW\_PBB\_TunnelTemplate\_Config\_Set and GSW\_PBB\_TunnelTemplate\_Config\_Get For GSW\_PBB\_TunnelTemplate\_Free, this field should be valid ID returned by GSW\_PBB\_TunnelTemplate\_Alloc.

##### Prototype

```
struct
{
    u16 nTunnelTemplateId;
    gsw_bool_t bIheaderDstMACEnable;
    u8 nIheaderDstMAC[GSW_MAC_ADDR_LEN];
    gsw_bool_t bIheaderSrcMACEnable;
    u8 nIheaderSrcMAC[GSW_MAC_ADDR_LEN];
    gsw_bool_t bItagEnable;
    GSW_I_TAG_Config_t sItag;
    gsw_bool_t bBtagEnable;
    GSW_B_TAG_Config_t sBtag;
} GSW_PBB_Tunnel_Template_Config_t;
```

##### Parameters

Data Type	Name	Description
u16	nTunnelTemplateId	
gsw_bool_t	blheaderDstMACEnable	I-Header Destination Address
u8	nlheaderDstMAC[GSW_MAC_ADDR_LEN]	
gsw_bool_t	blheaderSrcMACEnable	I-Header source Address
u8	nlheaderSrcMAC[GSW_MAC_ADDR_LEN]	
gsw_bool_t	bltagEnable	I-Tag
<a href="#">GSW_I_TAG_Config_t</a>	sItag	

Data Type	Name	Description
gsw_bool_t	bBtagEnable	B-Tag
<a href="#">GSW_B_TAG_Config_t</a>	sBtag	

### 1.10.1.45 GSW\_PCE\_pattern\_t

#### Description

Packet Classification Engine Pattern Configuration. GSWIP-3.0 has additional patterns such as Inner IP, Inner DSCP, Inner Protocol, Exclude Mode etc. Used by [GSW\\_PCE\\_rule\\_t](#).

#### Prototype

```
struct
{
    u16 nIndex;
    u32 nDstIP_Mask;
    u32 nInnerDstIP_Mask;
    u32 nSrcIP_Mask;
    u32 nInnerSrcIP_Mask;
    u16 nSubIfId;
    u16 nPktLng;
    u16 nPktLngRange;
    u16 nMAC_DstMask;
    u16 nMAC_SrcMask;
    u16 nAppDataMSB;
    u16 nAppMaskRangeMSB;
    u16 nEtherType;
    u16 nEtherTypeMask;
    u16 nSessionId;
    u16 nPPP_Protocol;
    u16 nPPP_ProtocolMask;
    u16 nVid;
    u16 nSLAN_Vid;
    u16 nFlexibleField4_Value;
    u16 nFlexibleField4_MaskOrRange;
    u16 nFlexibleField3_Value;
    u16 nFlexibleField3_MaskOrRange;
    u16 nFlexibleField1_Value;
    u16 nFlexibleField1_MaskOrRange;
    u16 nOuterVidRange;
    u16 nPayload1;
    u16 nPayload1_Mask;
    u16 nPayload2;
    u16 nPayload2_Mask;
    u16 nParserFlagLSB;
    u16 nParserFlagLSB_Mask;
    u16 nParserFlagMSB;
    u16 nParserFlagMSB_Mask;
    u16 nParserFlag1LSB;
    u16 nParserFlag1LSB_Mask;
    u16 nParserFlag1MSB;
```

```
ul6 nParserFlag1MSB_Mask;
ul6 nAppDataLSB;
ul6 nAppMaskRangeLSB;
ul6 nInsertionFlag;
ul6 nVidRange;
ul6 nFlexibleField2_MaskOrRange;
ul6 nFlexibleField2_Value;
u8 nPortId;
u8 nDSCP;
u8 nInnerDSCP;
u8 nPCP;
u8 nSTAG_PCP_DEI;
u8 nMAC_Dst[6];
u8 nMAC_Src[6];
u8 nProtocol;
u8 nProtocolMask;
u8 nInnerProtocol;
u8 nInnerProtocolMask;
u8 nFlexibleField4_ParserIndex;
u8 nFlexibleField3_ParserIndex;
u8 nFlexibleField1_ParserIndex;
u8 nFlexibleField2_ParserIndex;
gsw_bool_t bEnable;
gsw_bool_t bPortIdEnable;
gsw_bool_t bPortId_Exclude;
GSW_PCE_SUBIFID_TYPE_t eSubIfIdType;
gsw_bool_t bSubIfIdEnable;
gsw_bool_t bSubIfId_Exclude;
gsw_bool_t bDSCP_Enable;
gsw_bool_t bDSCP_Exclude;
gsw_bool_t bInner_DSCP_Enable;
gsw_bool_t bInnerDSCP_Exclude;
gsw_bool_t bPCP_Enable;
gsw_bool_t bCTAG_PCP_DEI_Exclude;
gsw_bool_t bSTAG_PCP_DEI_Enable;
gsw_bool_t bSTAG_PCP_DEI_Exclude;
gsw_bool_t bPktLngEnable;
gsw_bool_t bPktLng_Exclude;
gsw_bool_t bMAC_DstEnable;
gsw_bool_t bDstMAC_Exclude;
gsw_bool_t bMAC_SrcEnable;
gsw_bool_t bSrcMAC_Exclude;
gsw_bool_t bAppDataMSB_Enable;
gsw_bool_t bAppMaskRangeMSB_Select;
gsw_bool_t bAppMSB_Exclude;
gsw_bool_t bAppDataLSB_Enable;
gsw_bool_t bAppMaskRangeLSB_Select;
gsw_bool_t bAppLSB_Exclude;
GSW_PCE_IP_t eDstIP_Select;
GSW_IP_t nDstIP;
gsw_bool_t bDstIP_Exclude;
```

```
GSW_PCE_IP_t eInnerDstIP_Select;
GSW_IP_t nInnerDstIP;
gsw_bool_t bInnerDstIP_Exclude;
GSW_PCE_IP_t eSrcIP_Select;
GSW_IP_t nSrcIP;
gsw_bool_t bSrcIP_Exclude;
GSW_PCE_IP_t eInnerSrcIP_Select;
GSW_IP_t nInnerSrcIP;
gsw_bool_t bInnerSrcIP_Exclude;
gsw_bool_t bEtherTypeEnable;
gsw_bool_t bEtherType_Exclude;
gsw_bool_t bProtocolEnable;
gsw_bool_t bProtocol_Exclude;
gsw_bool_t bInnerProtocolEnable;
gsw_bool_t bInnerProtocol_Exclude;
gsw_bool_t bSessionIdEnable;
gsw_bool_t bSessionId_Exclude;
gsw_bool_t bPPP_ProtocolEnable;
gsw_bool_t bPPP_Protocol_Exclude;
gsw_bool_t bVid;
gsw_bool_t bVidRange_Select;
gsw_bool_t bVid_Exclude;
gsw_bool_t bVid_Original;
gsw_bool_t bSLAN_Vid;
gsw_bool_t bSLANVid_Exclude;
gsw_bool_t bSVidRange_Select;
gsw_bool_t bOuterVid_Original;
gsw_bool_t bPayload1_SrcEnable;
gsw_bool_t bPayload1MaskRange_Select;
gsw_bool_t bPayload1_Exclude;
gsw_bool_t bPayload2_SrcEnable;
gsw_bool_t bPayload2MaskRange_Select;
gsw_bool_t bPayload2_Exclude;
gsw_bool_t bParserFlagLSB_Enable;
gsw_bool_t bParserFlagLSB_Exclude;
gsw_bool_t bParserFlagMSB_Enable;
gsw_bool_t bParserFlagMSB_Exclude;
gsw_bool_t bParserFlag1LSB_Enable;
gsw_bool_t bParserFlag1LSB_Exclude;
gsw_bool_t bParserFlag1MSB_Enable;
gsw_bool_t bParserFlag1MSB_Exclude;
gsw_bool_t bInsertionFlag_Enable;
gsw_bool_t bFlexibleField4Enable;
gsw_bool_t bFlexibleField4_ExcludeEnable;
gsw_bool_t bFlexibleField4_RangeEnable;
gsw_bool_t bFlexibleField3Enable;
gsw_bool_t bFlexibleField3_ExcludeEnable;
gsw_bool_t bFlexibleField3_RangeEnable;
gsw_bool_t bFlexibleField2Enable;
gsw_bool_t bFlexibleField2_ExcludeEnable;
gsw_bool_t bFlexibleField2_RangeEnable;
```

```

gsw_bool_t bFlexibleField1Enable;
gsw_bool_t bFlexibleField1_ExcludeEnable;
gsw_bool_t bFlexibleField1_RangeEnable;
} GSW_PCE_pattern_t;

```

## Parameters

Data Type	Name	Description
u16	nIndex	PCE Rule Index (Upto 512 rules supported in GSWIP-3.0, whereas 64 rules supported in GSWIP-2.x)
u32	nDstIP_Mask	Destination IP Nibble Mask. 1 bit represents 1 nibble mask of the 'nDstIP' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u32	nInnerDstIP_Mask	Inner Destination IP Nibble Mask - for GSWIP-3.0 only. 1 bit represents 1 nibble mask of the 'nInnerDstIP' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u32	nSrcIP_Mask	Source IP Nibble Mask (Outer for GSWIP-3.0). 1 bit represents 1 nibble mask of the 'nSrcIP' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u32	nInnerSrcIP_Mask	Inner Src IP Nibble Mask - for GSWIP-3.0 only. 1 bit represents 1 nibble mask of the 'nInnerSrcIP' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u16	nSubIfId	Incoming Sub-Interface ID value - used for GSWIP-3.0 only
u16	nPktLng	Packet length in bytes
u16	nPktLngRange	Packet length Range (from nPktLng to nPktLngRange)
u16	nMAC_DstMask	Destination MAC address nibble mask. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.

Module Reference

Data Type	Name	Description
u16	nMAC_SrcMask	Source MAC address nibble mask. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u16	nAppDataMSB	MSB Application field. The first 2 bytes of the packet content following the IP header for TCP/UDP packets (source port field), or the first 2 bytes of packet content following the Ethertype for non-IP packets. Any part of this content can be masked-out by a programmable bit mask 'nAppMaskRangeMSB'.
u16	nAppMaskRangeMSB	MSB Application mask/range. When used as a range parameter, 1 bit represents 1 nibble mask of the 'nAppDataMSB' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u16	nEtherType	Ethertype
u16	nEtherTypeMask	Ethertype Mask. 1 bit represents 1 nibble mask of the 'nEtherType' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u16	nSessionId	PPPoE Session Id
u16	nPPP_Protocol	PPP Protocol Value - used for GSWIP-3.0 only
u16	nPPP_ProtocolMask	PPP protocol Bit Mask (Positional bit 1 signifies masking of corresponding bit value in nPPP_Protocol) - for GSWIP-3.0 only.
u16	nVid	VLAN ID (CVID)
u16	nSLAN_Vid	STAG VLAN ID
u16	nFlexibleField4_Value	Flexible Field 4 value. 16 bit value for pattern match
u16	nFlexibleField4_MaskOrRange	Flexible Field 4 mask or range value. If bFlexibleField4_MaskEnable is 1 then this 16 bit feid will be Mask or it will be Range
u16	nFlexibleField3_Value	Flexible Field 3 value. 16 bit value for pattern match
u16	nFlexibleField3_MaskOrRange	Flexible Field 3 mask or range value. If bFlexibleField4_MaskEnable is 1 then this 16 bit feid will be Mask or it will be Range
u16	nFlexibleField1_Value	Flexible Field 1 value. 16 bit value for pattern match



Module Reference

Data Type	Name	Description
u16	nFlexibleField1_MaskOrRange	Flexible Field 1 mask or range value. If bFlexibleField4_MaskEnable is 1 then this 16 bit field will be Mask or it will be Range
u16	nOuterVidRange	MSB Application Data Exclude - for GSWIP-3.0 only VLAN ID Range for outer VLAN tag. Used for GSWIP-3.1 only.
u16	nPayload1	Payload-1 Value (16-bits) - for GSWIP-3.0 PAE only
u16	nPayload1_Mask	Payload-1 Bit mask - for GSWIP-3.0 PAE only
u16	nPayload2	Payload-2 Value (16-bits) - for GSWIP-3.0 PAE only
u16	nPayload2_Mask	Payload-2 Bit mask - for GSWIP-3.0 PAE only
u16	nParserFlagLSB	Parser Flag LSW Value - each bit indicates specific parsed result
u16	nParserFlagLSB_Mask	Corresponding LSW Parser Flag Mask - when the bit is set to 1 corresponding flag gets masked out (ignored).
u16	nParserFlagMSB	Parser Flag MSW Value - each bit indicates specific parsed result
u16	nParserFlagMSB_Mask	Corresponding Parser Flag MSW Mask - when the bit is set to 1 corresponding flag gets masked out (ignored).
u16	nParserFlag1LSB	Parser Flag LSW Value - each bit indicates specific parsed result
u16	nParserFlag1LSB_Mask	Corresponding LSW Parser Flag Mask - when the bit is set to 1 corresponding flag gets masked out (ignored).
u16	nParserFlag1MSB	Parser Flag MSW Value - each bit indicates specific parsed result
u16	nParserFlag1MSB_Mask	Corresponding Parser Flag MSW Mask - when the bit is set to 1 corresponding flag gets masked out (ignored).
u16	nAppDataLSB	LSB Application field. The following 2 bytes of the packet behind the 'nAppDataMSB' field. This is the destination port field for TCP/UDP packets, or byte 3 and byte 4 of the packet content following the Ethertype for non-IP packets. Any part of this content can be masked-out by a programmable bit mask 'nAppMaskRangeLSB'.

Module Reference

Data Type	Name	Description
u16	nAppMaskRangeLSB	LSB Application mask/range. When used as a range parameter, 1 bit represents 1 nibble mask of the 'nAppDataLSB' field. Please clear the bits of the nibbles that are not marked out and set all other bits. The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u16	nInsertionFlag	Inserted packet by CPU to data path. For GSWIP-3.1 only
u16	nVidRange	VLAN ID Range (CVID). Gets used as mask to nVid in case bVidRange_Select is set to 0
u16	nFlexibleField2_MaskOrRange	Flexible Field 2 mask or range value. If bFlexibleField4_MaskEnable is 1 then this 16 bit feid will be Mask or it will be Range
u16	nFlexibleField2_Value	Flexible Field 2 value. 16 bit value for pattern match
u8	nPortId	Port ID value of incoming packets used for classification
u8	nDSCP	DSCP value (Outer for GSWIP-3.0)
u8	nInnerDSCP	Inner DSCP value for GSWIP-3.0 only
u8	nPCP	CTAG VLAN PCP n DEI value
u8	nSTAG_PCP_DEI	STAG VLAN PCP value
u8	nMAC_Dst[6]	Destination MAC address
u8	nMAC_Src[6]	Source MAC address
u8	nProtocol	IP protocol Value
u8	nProtocolMask	IP protocol Mask. 1 bit represents 1 nibble mask of the 'nProtocol' field. Please clear the bits of the nibbles that are not marked out and set all other bits i.e. a set bit 1 indicates that bit is masked out (not compared). The LSB bit represents the lowest data nibble, the next bit the next nibble, and so on.
u8	nInnerProtocol	Inner IP protocol Value - for GSWIP-3.0 only.
u8	nInnerProtocolMask	Inner IP protocol Bit Mask - for GSWIP-3.0 only.
u8	nFlexibleField4_ParserIndex	Flexible Field 4 parser out put index 0-127
u8	nFlexibleField3_ParserIndex	Flexible Field 3 parser out put index 0-127
u8	nFlexibleField1_ParserIndex	Flexible Field 1 parser out put index 0-127
u8	nFlexibleField2_ParserIndex	Flexible Field 2 parser out put index 0-127
gsw_bool_t	bEnable	Index is used (enabled) or set to unused (disabled)
gsw_bool_t	bPortIdEnable	Port ID used for ingress packet classification

Module Reference

Data Type	Name	Description
gsw_bool_t	bPortId_Exclude	Exclude Port Id Value - When set exclusion of specified nPortId takes effect. Available for GSWIP-3.0 only
<b>GSW_PCE_SUBIFID_TYPE_t</b>	eSubIfIdType	Select mode of sub-interface ID field
gsw_bool_t	bSubIfIdEnable	Incoming Sub-Interface ID Enable - used for GSWIP-3.0 only
gsw_bool_t	bSubIfId_Exclude	Exclude of specified Sub-Interface Id value in nSubIfId - used for GSWIP-3.0 only
gsw_bool_t	bDSCP_Enable	DSCP value used (Outer for GSWIP-3.0)
gsw_bool_t	bDSCP_Exclude	Exclude (Outer) DSCP value used for GSWIP-3.0 only
gsw_bool_t	bInner_DSCP_Enable	Inner DSCP value used for GSWIP-3.0 only
gsw_bool_t	bInnerDSCP_Exclude	Exclude of Inner DSCP (nInnerDSCP) value used for GSWIP-3.0 only
gsw_bool_t	bPCP_Enable	CTAG VLAN PCP n DEI value used
gsw_bool_t	bCTAG_PCP_DEI_Exclude	Exclude CTAG PCP & DEI value used for GSWIP-3.0 only
gsw_bool_t	bSTAG_PCP_DEI_Enable	STAG VLAN PCP/DEI value used
gsw_bool_t	bSTAG_PCP_DEI_Exclude	Exclude STAG PCP & DEI value used for GSWIP-3.0 only
gsw_bool_t	bPktLngEnable	Packet length used for classification
gsw_bool_t	bPktLng_Exclude	Exclude of Packet Length or range value used for GSWIP-3.0 only
gsw_bool_t	bMAC_DstEnable	Destination MAC address used
gsw_bool_t	bDstMAC_Exclude	Exclude Destination MAC Address used for GSWIP-3.0 only
gsw_bool_t	bMAC_SrcEnable	Source MAC address used
gsw_bool_t	bSrcMAC_Exclude	Exclude Source MAC Address used for GSWIP-3.0 only
gsw_bool_t	bAppDataMSB_Enable	MSB Application field used
gsw_bool_t	bAppMaskRangeMSB_Select	MSB Application mask/range selection. If set to LTQ_TRUE, the field 'nAppMaskRangeMSB' is used as a range parameter, otherwise it is used as a nibble mask field.
gsw_bool_t	bAppMSB_Exclude	
gsw_bool_t	bAppDataLSB_Enable	LSB Application used
gsw_bool_t	bAppMaskRangeLSB_Select	LSB Application mask/range selection. If set to LTQ_TRUE, the field 'nAppMaskRangeLSB' is used as a range parameter, otherwise it is used as a nibble mask field.
gsw_bool_t	bAppLSB_Exclude	LSB Application Data Exclude - for GSWIP-3.0 only

## Module Reference

Data Type	Name	Description
<b>GSW_PCE_IP_t</b>	eDstIP_Select	Destination IP Selection (Outer for GSWIP-3.0).
<b>GSW_IP_t</b>	nDstIP	Destination IP (Outer for GSWIP-3.0)
gsw_bool_t	bDstIP_Exclude	Exclude Destination IP Value - used for GSWIP-3.0 only
<b>GSW_PCE_IP_t</b>	eInnerDstIP_Select	Inner Destination IP Selection - for GSWIP-3.0 only.
<b>GSW_IP_t</b>	nInnerDstIP	Inner Destination IP - for GSWIP-3.0 only.
gsw_bool_t	bInnerDstIP_Exclude	Exclude Inner Destination IP Value - used for GSWIP-3.0 only
<b>GSW_PCE_IP_t</b>	eSrcIP_Select	Source IP Selection (Outer for GSWIP-3.0).
<b>GSW_IP_t</b>	nSrcIP	Source IP (Outer for GSWIP-3.0)
gsw_bool_t	bSrcIP_Exclude	Exclude Source IP Value - used for GSWIP-3.0 only
<b>GSW_PCE_IP_t</b>	eInnerSrcIP_Select	Inner Source IP Selection - for GSWIP-3.0 only.
<b>GSW_IP_t</b>	nInnerSrcIP	Inner Source IP - for GSWIP-3.0 only
gsw_bool_t	bInnerSrcIP_Exclude	Exclude Inner Source IP Value - used for GSWIP-3.0 only
gsw_bool_t	bEtherTypeEnable	Ethertype used.
gsw_bool_t	bEtherType_Exclude	Exclude for Ether Type Value - used for GSWIP-3.0 only.
gsw_bool_t	bProtocolEnable	IP protocol used
gsw_bool_t	bProtocol_Exclude	Exclude for IP Protocol Value - used for GSWIP-3.0 only.
gsw_bool_t	bInnerProtocolEnable	Inner IP protocol used - for GSWIP-3.0 only.
gsw_bool_t	bInnerProtocol_Exclude	Exclude for Inner IP Protocol Value - used for GSWIP-3.0 only.
gsw_bool_t	bSessionIdEnable	PPPoE used.
gsw_bool_t	bSessionId_Exclude	Exclude for PPPoE Session Value - used for GSWIP-3.0 only.
gsw_bool_t	bPPP_ProtocolEnable	PPP Protocol used - used for GSWIP-3.0 only
gsw_bool_t	bPPP_Protocol_Exclude	Exclude for PPP Protocol Value - used for GSWIP-3.0 only.
gsw_bool_t	bVid	VLAN ID (CVID) used. <i>Note: CVID is inner VLAN as defined in GSWIP-3.1</i>
gsw_bool_t	bVidRange_Select	VID mask/range selection. If set to 1, the field 'nVidRange' is used as a range parameter, otherwise it is used as a mask field. <i>Note: This must be range in GSWIP-3.1</i>
gsw_bool_t	bVid_Exclude	Exclude for VLAN Id (CVLAN) - used for GSWIP-3.0 only.

Module Reference

Data Type	Name	Description
gsw_bool_t	bVid_Original	If this field is TRUE, use original VLAN ID as key even it's modified in any stage before flow table process. Used for GSWIP-3.1 only.
gsw_bool_t	bSLAN_Vid	STAG VLAN ID used. <i>Note: SLAN is outer VLAN as defined GSWIP-3.1</i>
gsw_bool_t	bSLANVid_Exclude	Exclude for SVLAN Id (SVLAN) - used for GSWIP-3.0 only.
gsw_bool_t	bSVidRange_Select	VID mask/range selection. If set to 1, the field 'nVidRange' is used as a range parameter, otherwise it is used as a mask field. <i>Note: This must be range in GSWIP-3.1</i>
gsw_bool_t	bOuterVid_Original	If this field is TRUE, use original VLAN ID as key even it's modified in any stage before flow table process. Used for GSWIP-3.1 only.
gsw_bool_t	bPayload1_SrcEnable	Payload-1 used - for GSWIP-3.0 PAE only
gsw_bool_t	bPayload1MaskRange_Select	Payload1 mask/range selection. If set to LTQ_TRUE, the field 'nPayload1' is used as a range parameter, otherwise it is used as a bit mask field.
gsw_bool_t	bPayload1_Exclude	Exclude Payload-1 used for GSWIP-3.0 PAE only
gsw_bool_t	bPayload2_SrcEnable	Payload-2 used - for GSWIP-3.0 PAE only
gsw_bool_t	bPayload2MaskRange_Select	Payload2 mask/range selection. If set to LTQ_TRUE, the field 'nPayload2' is used as a range parameter, otherwise it is used as a bit mask field.
gsw_bool_t	bPayload2_Exclude	Exclude Payload-2 used for GSWIP-3.0 PAE only
gsw_bool_t	bParserFlagLSB_Enable	Parser Flag LSW (Bit position 15 to 0) is used - for GSWIP 3.0 only
gsw_bool_t	bParserFlagLSB_Exclude	Exclude for Parser Flag LSW specified in nParserFlagLSB
gsw_bool_t	bParserFlagMSB_Enable	Parser Flag MSW (Bit 31 to 16) is used - for GSWIP 3.0 only
gsw_bool_t	bParserFlagMSB_Exclude	Exclude for Parser Flag MSW specified in nParserFlagMSB
gsw_bool_t	bParserFlag1LSB_Enable	Parser Flag LSW (Bit position 47 to 32) is used - for GSWIP 3.1 only
gsw_bool_t	bParserFlag1LSB_Exclude	Exclude for Parser Flag LSW specified in nParserFlagLSB
gsw_bool_t	bParserFlag1MSB_Enable	Parser Flag MSW (Bit 63 to 48) is used - for GSWIP 3.1 only

Data Type	Name	Description
gsw_bool_t	bParserFlag1MSB_Exclude	Exclude for Parser Flag MSW specified in nParserFlagMSB
gsw_bool_t	bInsertionFlag_Enable	nInsertionFlag is used. For GSWIP-3.1 only
gsw_bool_t	bFlexibleField4Enable	
gsw_bool_t	bFlexibleField4_ExcludeEnable	Flexible Field 4 exclude mode 1 enable and 0 disable
gsw_bool_t	bFlexibleField4_RangeEnable	Flexible Field 4 parser mask or range selection - 0 mask/1 range
gsw_bool_t	bFlexibleField3Enable	
gsw_bool_t	bFlexibleField3_ExcludeEnable	Flexible Field 3 exclude mode 1 enable and 0 disable
gsw_bool_t	bFlexibleField3_RangeEnable	Flexible Field 3 parser mask or range selection - 0 mask/1 range
gsw_bool_t	bFlexibleField2Enable	
gsw_bool_t	bFlexibleField2_ExcludeEnable	Flexible Field 2 exclude mode 1 enable and 0 disable
gsw_bool_t	bFlexibleField2_RangeEnable	Flexible Field 2 parser mask or range selection - 0 mask/1 range
gsw_bool_t	bFlexibleField1Enable	
gsw_bool_t	bFlexibleField1_ExcludeEnable	Flexible Field 1 exclude mode 1 enable and 0 disable
gsw_bool_t	bFlexibleField1_RangeEnable	Flexible Field 1 parser mask or range selection - 0 mask/1 range

### 1.10.1.46 GSW\_PCE\_rule\_t

#### Description

Parameter to add/read a rule to/from the packet classification engine. Used by GSW\_PCE\_RULE\_WRITE and GSW\_PCE\_RULE\_READ.

#### Prototype

```
struct
{
    u8 logicalportid;
    u16 subifidgroup;
    GSW_PCE_RuleRegion_t region;
    GSW_PCE_pattern_t pattern;
    action;
} GSW_PCE_rule_t;
```

## Parameters

Data Type	Name	Description
u8	logicalportid	Logical Port Id. The valid range is hardware dependent.
u16	subifidgroup	Sub interface ID group, The valid range is hardware/protocol dependent.
<a href="#">GSW_PCE_RuleRegion_t</a>	region	PCE TABLE Region
<a href="#">GSW_PCE_pattern_t</a>	pattern	PCE Rule Pattern Part.
	action	PCE Rule Action Part.

### 1.10.1.47 GSW\_PCE\_ruleDelete\_t

#### Description

Parameter to delete a rule from the packet classification engine. Used by GSW\_PCE\_RULE\_DELETE.

#### Prototype

```
struct
{
    u8 logicalportid;
    u16 subifidgroup;
    GSW_PCE_RuleRegion_t region;
    u16 nIndex;
} GSW_PCE_ruleDelete_t;
```

## Parameters

Data Type	Name	Description
u8	logicalportid	Logical Port Id. The valid range is hardware dependent.
u16	subifidgroup	Sub interface ID group, The valid range is hardware/protocol dependent.
<a href="#">GSW_PCE_RuleRegion_t</a>	region	PCE TABLE Region
u16	nIndex	Rule Index in the PCE Table.

### 1.10.1.48 GSW\_PMAC\_BM\_Cfg\_t

#### Description

Configure the Backpressure mapping for egress Queues Congestion or ingress (receiving) ports to DMA channel. Used by GSW\_PMAC\_BM\_CFG\_SET and GSW\_PMAC\_BM\_CFG\_GET.

#### Prototype

```
struct
{
    u8 nPmacId;
    u8 nTxDMAChanId;
```

```

        u32 txQMask;
        u32 rxPortMask;
    } GSW_PMAC_BM_Cfg_t;

```

## Parameters

Data Type	Name	Description
u8	nPmacId	PMAC Interface ID
u8	nTxDmaChanId	Tx DMA Channel Identifier which receives sideband backpressure signal (0..15)
u32	txQMask	Transmit Queues Selection Mask which will generate backpressure - (Configurable upto 32 Egress Queues)
u32	rxPortMask	Receive (Ingress) ports selection congestion Mask which will generate backpressure - (Configurable upto 16 ports)

### 1.10.1.49 GSW\_PMAC\_Cnt\_t

#### Description

PMAC Counters available for specified DMA Channel. Used by GSW\_PMAC\_COUNT\_GET.

#### Prototype

```

struct
{
    u8 nPmacId;
    gsw_bool_t b64BitMode;
    u8 nTxDmaChanId;
    u32 nDiscPktsCount;
    u32 nDiscBytesCount;
    u32 nChkSumErrPktsCount;
    u32 nChkSumErrBytesCount;
    u32 nIngressPktsCount;
    u32 nIngressBytesCount;
    u32 nEgressPktsCount;
    u32 nEgressBytesCount;
    u32 nIngressHdrPktsCount;
    u32 nIngressHdrBytesCount;
    u32 nEgressHdrPktsCount;
    u32 nEgressHdrBytesCount;
    u32 nEgressHdrDiscPktsCount;
    u32 nEgressHdrDiscBytesCount;
} GSW_PMAC_Cnt_t;

```



## Parameters

Data Type	Name	Description
u8	nPmacId	PMAC Interface ID Applicable only for GSWIP 3.1
gsw_bool_t	b64BitMode	
u8	nTxDmaChanId	Transmit DMA Channel Identifier (0..15) for GSWIP3.0 (0..16) for GSWIP3.1 Source PortId for Egress Counters (0..15) for GSWIP3.1 - Index
u32	nDiscPktsCount	Ingress Total discarded packets counter (32-bits)
u32	nDiscBytesCount	Ingress Total discarded bytes counter (32-bits)
u32	nChkSumErrPktsCount	Egress Total TCP/UDP/IP checksum error-ed packets counter (32-bits)
u32	nChkSumErrBytesCount	Egress Total TCP/UDP/IP checksum error-ed bytes counter (32-bits)
u32	nIngressPktsCount	Total Ingress Packet Count in Applicable only for GSWIP 3.1 (32-bits)
u32	nIngressBytesCount	Total Ingress Bytes Count in Applicable only for GSWIP 3.1 (32-bits)
u32	nEgressPktsCount	Total Egress Packet Count in Applicable only for GSWIP 3.1 (32-bits)
u32	nEgressBytesCount	Total Egress Bytes Count in Applicable only for GSWIP 3.1 (32-bits)
u32	nIngressHdrPktsCount	Ingress header Packet Count Applicable only for GSWIP 3.2 (32-bits)
u32	nIngressHdrBytesCount	Ingress header Byte Count Applicable only for GSWIP 3.2 (32-bits)
u32	nEgressHdrPktsCount	Egress header Packet Count Applicable only for GSWIP 3.2 (32-bits)
u32	nEgressHdrBytesCount	Egress header Byte Count Applicable only for GSWIP 3.2 (32-bits)
u32	nEgressHdrDiscPktsCount	Egress header Discard Packet Count Applicable only for GSWIP 3.2 (32-bits)
u32	nEgressHdrDiscBytesCount	Egress header Discard Byte Count Applicable only for GSWIP 3.2 (32-bits)

### 1.10.1.50 GSW\_PMAC\_Eg\_Cfg\_t

#### Description

Configure the PMAC Egress Configuration. (Upto 1024 entries) This Egress PMAC table is addressed through combination of following fields (Bit0 - Bit 9). nDestPortId (Bits 0-3) + Combination of [bMpe1Flag (Bit 4) + bMpe2Flag (Bit 5) + bEncFlag (Bit 6) + bDecFlag (Bit 7) ] or TrafficClass Value (Bits 4-7) + nFlowIdMSB (Bits 8-9). The bits 4-7 of index option is either based upon TC (default) or combination of Processing flags is decided

through bProcFlagsEgPMACena. It is expected to pass the correct value in bProcFlagsSelect same as global bProcFlagsEgPMACena; Used by GSW\_PMAC\_EG\_CFG\_SET and GSW\_PMAC\_EG\_CFG\_GET.

## Prototype

```
struct
{
    u8 nPmacId;
    u8 nDestPortId;
    u8 nTrafficClass;
    gsw_bool_t bMpe1Flag;
    gsw_bool_t bMpe2Flag;
    gsw_bool_t bDecFlag;
    gsw_bool_t bEncFlag;
    u8 nFlowIDMSb;
    gsw_bool_t bProcFlagsSelect;
    u8 nRxDmaChanId;
    gsw_bool_t bRemL2Hdr;
    u8 numBytesRem;
    gsw_bool_t bFcsEna;
    gsw_bool_t bPmacEna;
    gsw_bool_t bRedirEnable;
    gsw_bool_t bBslSegmentDisable;
    u8 nBslTrafficClass;
    gsw_bool_t bResDW1Enable;
    u8 nResDW1;
    gsw_bool_t bRes1DW0Enable;
    u8 nRes1DW0;
    gsw_bool_t bRes2DW0Enable;
    u8 nRes2DW0;
    gsw_bool_t bTCEnable;
} GSW_PMAC_Eg_Cfg_t;
```

## Parameters

Data Type	Name	Description
u8	nPmacId	PMAC Interface ID
u8	nDestPortId	Destination Port Identifier (0..15) - Part of Table Index (Bits 0-3)
u8	nTrafficClass	Traffic Class value [Lower 4 -bits (LSB-0, 1, 2, 3)]. - Part of Table Index Bits 4-7. This value is considered, only when bProcFlagsSelect is not set
gsw_bool_t	bMpe1Flag	MPE-1 Flag value - Part of Table Index Bit 4. Valid only when bProcFlagsSelect is set.
gsw_bool_t	bMpe2Flag	MPE-2 Flag value - Part of Table Index Bit 5. Valid only when bProcFlagsSelect is set.
gsw_bool_t	bDecFlag	Cryptography Decryption Action Flag value - Part of Table Index Bit 6. Valid only, when bProcFlagsSelect is set.

Module Reference

Data Type	Name	Description
gsw_bool_t	bEncFlag	Cryptography Encryption Action Flag value - Part of Table Index Bit 7. Valid only, when bProcFlagsSelect is set.
u8	nFlowIDMSb	Flow-ID MSB (2-bits) value - valid range (0..2). - Part of Table Index Bits 8-9.
gsw_bool_t	bProcFlagsSelect	Selector for Processing Flags (MPE1, MPE2, DEC & ENC bits). If enabled, then the combination of flags bDecFlag, bEncFlag, bMpe1Flag and bMpe2Flag are considered as index instead of nTrafficClass. For using these combination flags, turn ON this boolean selector. TC or combination processing flag is decided at global level through bProcFlagsEgPMACEna. It is expected that user always passes correct value based upon bProcFlagsEgMPACEna. If mismatch found with global PMAC mode, SWAPI will return error code. \remarks In GSWIP-3.1, this is ignored and driver will determine automatically by reading register.
u8	nRxDmaChanId	Receive DMA Channel Identifier (0..15)
gsw_bool_t	bRemL2Hdr	To remove L2 header & additional bytes (True) or Not (False)
u8	numBytesRem	No. of bytes to be removed after Layer-2 Header, valid when bRemL2Hdr is set
gsw_bool_t	bFcsEna	Packet egressing will have FCS (True) or Not (False)
gsw_bool_t	bPmacEna	Packet egressing will have PMAC (True) or Not (False)
gsw_bool_t	bRedirEnable	Enable redirection flag. GSWIP-3.1 only. Overwritten by bRes1DW0Enable and nRes1DW0.
gsw_bool_t	bBslSegmentDisable	Allow (False) or not allow (True) segmentation during buffer selection. GSWIP-3.1 only. Overwritten by bResDW1Enable and nResDW1.
u8	nBslTrafficClass	Traffic class used for buffer selection. GSWIP-3.1 only. Overwritten by bResDW1Enable and nResDW1.
gsw_bool_t	bResDW1Enable	If false, nResDW1 is ignored.
u8	nResDW1	4-bits Reserved Field in DMA Descriptor - DW1 (bit 7 to 4) - for any future/custom usage. (Valid range : 0-15)
gsw_bool_t	bRes1DW0Enable	If false, nRes1DW0 is ignored.

Data Type	Name	Description
u8	nRes1DW0	3-bits Reserved Field in DMA Descriptor - DW0 (bit 31 to 29) - for any future/custom usage. (Valid range : 0-7)
gsw_bool_t	bRes2DW0Enable	If false, nRes2DW0 is ignored.
u8	nRes2DW0	2-bits Reserved Field in DMA Descriptor - DW0 (bit 14 to 13) - for any future/custom usage. (Valid range : 0-2)
gsw_bool_t	bTCEnable	Selector for TrafficClass bits. If enabled, then the flags bDecFlag, bEncFlag, bMpe1Flag and bMpe2Flag are not used instead nTrafficClass parameter is used. For using these flags turn off this boolean

### 1.10.1.51 GSW\_PMAC\_Glbl\_Cfg\_t

#### Description

Configure the global settings of PMAC for GSWIP-3.x. This includes settings such as Jumbo frame, Checksum handling, Padding and Engress PMAC Selector Config. Used by GSW\_PMAC\_GLBL\_CFG\_SET and GSW\_PMAC\_GLBL\_CFG\_GET.

#### Prototype

```
struct
{
    u8 nPmacId;
    gsw_bool_t bAPadEna;
    gsw_bool_t bPadEna;
    gsw_bool_t bVPadEna;
    gsw_bool_t bSVPadEna;
    gsw_bool_t bRxFCSDis;
    gsw_bool_t bTxFCSDis;
    gsw_bool_t bIPTransChkRegDis;
    gsw_bool_t bIPTransChkVerDis;
    gsw_bool_t bJumboEna;
    u16 nMaxJumboLen;
    u16 nJumboThreshLen;
    gsw_bool_t bLongFrmChkDis;
    GSW_PMAC_Short_Frame_Chk_t eShortFrmChkType;
    gsw_bool_t bProcFlagsEgCfgEna;
    GSW_PMAC_Proc_Flags_Eg_Cfg_t eProcFlagsEgCfg;
    u32 nBslThreshold[3];
} GSW_PMAC_Glbl_Cfg_t;
```

## Parameters

Data Type	Name	Description
u8	nPmacId	PMAC Interface Id
gsw_bool_t	bAPadEna	Automatic Padding Settings - Disabled (Default), to enable set it true.
gsw_bool_t	bPadEna	Global Padding Settings - Disabled (Default), to enable set it true.
gsw_bool_t	bVPadEna	VLAN Padding Setting - Disabled (Default), to enable set it true - applicable when bPadEna is set.
gsw_bool_t	bSVPadEna	Stacked VLAN Padding Setting - Disabled (Default), to enable set it true - applicable when bPadEna is set.
gsw_bool_t	bRxFCSDis	Packet carry FCS after PMAC process - Disabled (Default), to enable set it true.
gsw_bool_t	bTxFCSDis	Transmit FCS Regeneration Setting - Disabled (Default), to enable set it true.
gsw_bool_t	bIPTransChkRegDis	IP and Transport (TCP/UDP) Headers Checksum Generation Control - Enabled (Default), to disable set it true.
gsw_bool_t	bIPTransChkVerDis	IP and Transport (TCP/UDP) Headers Checksum Verification Control - Enabled (Default), to disable set it true.
gsw_bool_t	bJumboEna	To enable receipt of Jumbo frames - Disabled (Default - 1518 bytes normal frames without VLAN tags), to enable Jumbo set it true.
u16	nMaxJumboLen	Maximum length of Jumbo frames in terms of bytes (Bits 13:0). The maximum handled in Switch is 9990 bytes.
u16	nJumboThreshLen	Threshold length for Jumbo frames qualification in terms of bytes (Bits 13:0).
gsw_bool_t	bLongFrmChkDis	Long frame length check - Enabled (Default), to disable set it true.
<a href="#">GSW_PMAC_Short_Frame_Chk_t</a>	eShortFrmChkType	Short frame length check Type - default (Enabled for 64 bytes without considering VLAN).
gsw_bool_t	bProcFlagsEgCfgEna	GSWIP3.0 specific - Egress PMAC Config Table Selector - TrafficClass or Processing Flags (MPE1, MPE22, DEC, ENC based). The default setting is Traffic Class based selector for Egress PMAC.

Data Type	Name	Description
<a href="#">GSW_PMAC_Proc_Flags_Eg_Cfg_t</a>	eProcFlagsEgCfg	GSWIP3.1 specific - Egress PMAC Config Table Selector If this field is not <a href="#">GSW_PMAC_PROC_FLAGS_NONE</a> , it will override bProcFlagsEgCfgEna.
u32	nBslThreshold[3]	GSWIP3.1 specific - frame size threshold for buffer selection. Value in this array should be in ascending order.

### 1.10.1.52 GSW\_PMAC\_Ig\_Cfg\_t

#### Description

Configure the PMAC Ingress Configuration on a given Tx DMA channel to PMAC. (Upto 16 entries). This Ingress PMAC table is addressed through Transmit DMA Channel Identifier. Used by GSW\_PMAC\_IG\_CFG\_SET and GSW\_PMAC\_IG\_CFG\_GET.

#### Prototype

```
struct
{
    u8 nPmacId;
    u8 nTxDmaChanId;
    gsw_bool_t bErrPktsDisc;
    gsw_bool_t bPmapDefault;
    gsw_bool_t bPmapEna;
    gsw_bool_t bClassDefault;
    gsw_bool_t bClassEna;
    GSW_PMAC_Ig_Cfg_Src_t eSubId;
    gsw_bool_t bSpIdDefault;
    gsw_bool_t bPmacPresent;
    u8 defPmacHdr[8];
} GSW_PMAC_Ig_Cfg_t;
```

#### Parameters

Data Type	Name	Description
u8	nPmacId	PMAC Interface Id
u8	nTxDmaChanId	Tx DMA Channel Identifier (0..16) - Index of Ingress PMAC Config Table
gsw_bool_t	bErrPktsDisc	Error set packets to be discarded (True) or not (False)
gsw_bool_t	bPmapDefault	Port Map info from default PMAC header (True) or incoming PMAC header (False)
gsw_bool_t	bPmapEna	Port Map Enable info from default PMAC header (True) or incoming PMAC header (False)
gsw_bool_t	bClassDefault	Class Info from default PMAC header (True) or incoming PMAC header (False)

Data Type	Name	Description
gsw_bool_t	bClassEna	Class Enable info from default PMAC header (True) or incoming PMAC header (False)
<a href="#">GSW_PMAC_Ig_Cfg_Src_t</a>	eSubId	Sub_Interface Id Info from ingress PMAC header (GSW_PMAC_IG_CFG_SRC_PMAC), default PMAC header (GSW_PMAC_IG_CFG_SRC_DEF_PMAC), or source sub-If in packet descriptor (GSW_PMAC_IG_CFG_SRC_DMA_DESC)
gsw_bool_t	bSpldDefault	Source Port Id from default PMAC header (True) or incoming PMAC header (False)
gsw_bool_t	bPmacPresent	Packet PMAC header is present (True) or not (False)
u8	defPmacHdr[8]	Default PMAC header - 8 Bytes Configuration - Ingress PMAC Header Format

### 1.10.1.53 GSW\_PMAPPER\_t

#### Description

P-mapper Configuration Used by [GSW\\_CTP\\_portConfig\\_t](#), [GSW\\_BRIDGE\\_portConfig\\_t](#). In case of LAG, it is user's responsibility to provide the mapped entries in given P-mapper table. In other modes the entries are auto mapped from input packet.

#### Prototype

```
struct
{
    u16 nPmapperId;
    u8 nDestSubIfIdGroup[73];
} GSW_PMAPPER_t;
```

## Parameters

Data Type	Name	Description
u16	nPmapperId	Index of P-mapper <0-31>.
u8	nDestSubIfIdGroup[73]	Sub interface ID group. <i>Note: Entry 0 is for non-IP and non-VLAN tagged packets. Entries 1-8 are PCP mapping entries for VLAN tagged packets with <b>GSW_PMAPPER_MAPPING_PCP</b> selected. Entries 9-72 are DSCP or LAG mapping entries for IP packets without VLAN tag or VLAN tagged packets with <b>GSW_PMAPPER_MAPPING_DSCP</b> or <b>GSW_PMAPPER_MAPPING_LAG</b> selected. When LAG is selected this 8bit field is decoded as Destination sub-interface ID group field bits 3:0, Destination logical port ID field bits 7:4</i>

### 1.10.1.54 GSW\_portCfg\_t

#### Description

Port Configuration Parameters. Used by GSW\_PORT\_CFG\_GET and GSW\_PORT\_CFG\_SET.

#### Prototype

```
struct
{
    GSW_portType_t ePortType;
    u16 nPortId;
    GSW_portEnable_t eEnable;
    gsw_bool_t bUnicastUnknownDrop;
    gsw_bool_t bMulticastUnknownDrop;
    gsw_bool_t bReservedPacketDrop;
    gsw_bool_t bBroadcastDrop;
    gsw_bool_t bAging;
    gsw_bool_t bLearning;
    gsw_bool_t bLearningMAC_PortLock;
    u16 nLearningLimit;
    gsw_bool_t bMAC_SpoofingDetection;
    GSW_portFlow_t eFlowCtrl;
    GSW_portMonitor_t ePortMonitor;
    gsw_bool_t bIfCounters;
    int nIfCountStartIdx;
    GSW_If_RMON_Mode_t eIfRMONmode;
} GSW_portCfg_t;
```



## Parameters

Data Type	Name	Description
<a href="#">GSW_portType_t</a>	ePortType	Port Type. This gives information which type of port is configured. nPortId should be based on this field.
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.
<a href="#">GSW_portEnable_t</a>	eEnable	Enable Port (ingress only, egress only, both directions, or disabled). This parameter is used for Spanning Tree Protocol and 802.1X applications.
gsw_bool_t	bUnicastUnknownDrop	Drop unknown unicast packets. Do not send out unknown unicast packets on this port, if the boolean parameter is enabled. By default packets of this type are forwarded to this port.
gsw_bool_t	bMulticastUnknownDrop	Drop unknown multicast packets. Do not send out unknown multicast packets on this port, if boolean parameter is enabled. By default packets of this type are forwarded to this port. Some platforms also drop broadcast packets.
gsw_bool_t	bReservedPacketDrop	Drop reserved packet types (destination address from '01 80 C2 00 00 00' to '01 80 C2 00 00 2F') received on this port.
gsw_bool_t	bBroadcastDrop	Drop Broadcast packets received on this port. By default packets of this type are forwarded to this port.
gsw_bool_t	bAging	Enables MAC address table aging. The MAC table entries learned on this port are removed after the aging time has expired. The aging time is a global parameter, common to all ports.
gsw_bool_t	bLearning	MAC address table learning on the port specified by 'nPortId'. By default this parameter is always enabled.
gsw_bool_t	bLearningMAC_PortLock	Automatic MAC address table learning locking on the port specified by 'nPortId'. This parameter is only taken into account when 'bLearning' is enabled.
u16	nLearningLimit	Automatic MAC address table learning limitation on this port. The learning functionality is disabled when the limit value is zero. The value 0xFFFF to allow unlimited learned address. This parameter is only taken into account when 'bLearning' is enabled.

Data Type	Name	Description
gsw_bool_t	bMAC_SpoofingDetection	MAC spoofing detection. Identifies ingress packets that carry a MAC source address which was previously learned on a different ingress port (learned by MAC bridging table). This also applies to static added entries. Those violated packets could be accepted or discarded, depending on the global switch configuration 'bMAC_SpoofingAction'. This parameter is only taken into account when 'bLearning' is enabled.
<a href="#">GSW_portFlow_t</a>	eFlowCtrl	Port Flow Control Status. Enables the flow control function.
<a href="#">GSW_portMonitor_t</a>	ePortMonitor	Port monitor feature. Allows forwarding of egress and/or ingress packets to the monitor port. If enabled, the monitor port gets a copy of the selected packet type.
gsw_bool_t	blfCounters	Assign Interface RMON Counters for this Port - GSWIP-3.0
int	nIfCountStartIdx	Interface RMON Counters Start Index - GSWIP-3.0. Value of (-1) denotes unassigned Interface Counters. Valid range : 0-255 available to be shared amongst ports in desired way
<a href="#">GSW_If_RMON_Mode_t</a>	elfRMONmode	Interface RMON Counters Mode - GSWIP-3.0

### 1.10.1.55 GSW\_portLinkCfg\_t

#### Description

Ethernet port link, speed status and flow control status. Used by GSW\_PORT\_LINK\_CFG\_GET and GSW\_PORT\_LINK\_CFG\_SET.

#### Prototype

```
struct
{
    ul6 nPortId;
    gsw_bool_t bDuplexForce;
    GSW_portDuplex_t eDuplex;
    gsw_bool_t bSpeedForce;
    GSW_portSpeed_t eSpeed;
    gsw_bool_t bLinkForce;
    GSW_portLink_t eLink;
    GSW_MII_Mode_t eMII_Mode;
    GSW_MII_Type_t eMII_Type;
    GSW_clkMode_t eClkMode;
    gsw_bool_t bLPI;
} GSW_portLinkCfg_t;
```

## Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.
gsw_bool_t	bDuplexForce	Force Port Duplex Mode. <ul style="list-style-type: none"> <li>0: Negotiate Duplex Mode. Auto-negotiation mode. Negotiated duplex mode given in 'eDuplex' during GSW_PORT_LINK_CFG_GET calls.</li> <li>1: Force Duplex Mode. Force duplex mode in 'eDuplex'.</li> </ul>
<a href="#">GSW_portDuplex_t</a>	eDuplex	Port Duplex Status.
gsw_bool_t	bSpeedForce	Force Link Speed. <ul style="list-style-type: none"> <li>0: Negotiate Link Speed. Negotiated speed given in 'eSpeed' during GSW_PORT_LINK_CFG_GET calls.</li> <li>1: Force Link Speed. Forced speed mode in 'eSpeed'.</li> </ul>
<a href="#">GSW_portSpeed_t</a>	eSpeed	Ethernet port link up/down and speed status.
gsw_bool_t	bLinkForce	Force Link. <ul style="list-style-type: none"> <li>0: Auto-negotiate Link. Current link status is given in 'eLink' during GSW_PORT_LINK_CFG_GET calls.</li> <li>1: Force Duplex Mode. Force duplex mode in 'eLink'.</li> </ul>
<a href="#">GSW_portLink_t</a>	eLink	Link Status. Read out the current link status. Note that the link could be forced by setting 'bLinkForce'.
<a href="#">GSW_MII_Mode_t</a>	eMII_Mode	Selected interface mode (MII/RMII/RGMII/GMII/XGMII).
<a href="#">GSW_MII_Type_t</a>	eMII_Type	Select MAC or PHY mode (PHY = Reverse xMII).
<a href="#">GSW_clkMode_t</a>	eClkMode	Interface Clock mode (used for RMII mode).
gsw_bool_t	bLPI	'Low Power Idle' Support for 'Energy Efficient Ethernet'. Only enable this feature in case the attached PHY also supports it.

### 1.10.1.56 GSW\_QoS\_ClassPCP\_Cfg\_t

#### Description

Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW\_QOS\_CLASS\_PCP\_SET and GSW\_QOS\_CLASS\_PCP\_GET.

### Prototype

```
struct
{
    u8 nPCP[16];
} GSW_QoS_ClassPCP_Cfg_t;
```

### Parameters

Data Type	Name	Description
u8	nPCP[16]	Configures the traffic class to PCP (3-bit) mapping. The queue index starts counting from zero.

## 1.10.1.57 GSW\_QoS\_colorMarkingEntry\_t

### Description

Color Marking Table. There are standards to define the marking table. User should use GSW\_QOS\_COLOR\_MARKING\_TABLE\_SET to initialize the table before color marking happens. GSW\_QOS\_COLOR\_MARKING\_TABLE\_GET is used to get the marking table, mainly for debug purpose.

### Prototype

```
struct
{
    GSW_ColorMarkingMode_t eMode;
    u8 nPriority[64];
    u8 nColor[64];
} GSW_QoS_colorMarkingEntry_t;
```

### Parameters

Data Type	Name	Description
<a href="#">GSW_ColorMarkingMode_t</a>	eMode	Mode of color marking.

Data Type	Name	Description
u8	nPriority[64]	If eMode is GSW_REMARKING_DSCP_AF, index stands for 6-bit DSCP value. If eMode is one of GSW_REMARKING_PCP_8P0D, GSW_REMARKING_PCP_7P1D, GSW_REMARKING_PCP_6P2D and GSW_REMARKING_PCP_5P3D, index 0-7 is 3-bit PCP value with DEI is 0, and index 8-15 is 3-bit PCP value with DEI is 1. Ignored in other modes.
u8	nColor[64]	If eMode is GSW_REMARKING_DSCP_AF, index stands for 6-bit DSCP value. If eMode is one of GSW_REMARKING_PCP_8P0D, GSW_REMARKING_PCP_7P1D, GSW_REMARKING_PCP_6P2D and GSW_REMARKING_PCP_5P3D, index 0-7 is 3-bit PCP value with DEI is 0, and index 8-15 is 3-bit PCP value with DEI is 1. Ignored in other modes. Value refers to <a href="#">GSW_QoS_DropPrecedence_t</a> .

### 1.10.1.58 GSW\_QoS\_colorRemarkingEntry\_t

#### Description

Color Remarking Table. There are standards to define the remarking table. User should use GSW\_QOS\_COLOR\_REMARKING\_TABLE\_SET to initialize the table before color remarking happens. GSW\_QOS\_COLOR\_REMARKING\_TABLE\_GET is used to get the remarking table, mainly for debug purpose.

#### Prototype

```
struct
{
    GSW_ColorRemarkingMode_t eMode;
    u8 nVal[16];
} GSW_QoS_colorRemarkingEntry_t;
```

## Parameters

Data Type	Name	Description
<a href="#">GSW_ColorRemarkingMode_t</a>	eMode	Mode of color remarking.
u8	nVal[16]	Index stands for color and priority. Index 0-7 is green color with priority (traffic class) 0-7. Index 8-15 is yellow color with priority (traffic class) 0-7. Value is DSCP if eMode is GSW_REMARKING_DSCP_AF. Value bit 0 is DEI and bit 1-3 is PCP if eMode is one of GSW_REMARKING_PCP_8P0D, GSW_REMARKING_PCP_7P1D, GSW_REMARKING_PCP_6P2D and GSW_REMARKING_PCP_5P3D. Value is ignored for other mode.

### 1.10.1.59 GSW\_QoS\_DSCP\_ClassCfg\_t

#### Description

DSCP mapping table. Used by GSW\_QOS\_DSCP\_CLASS\_SET and GSW\_QOS\_DSCP\_CLASS\_GET.

#### Prototype

```
struct
{
    u8 nTrafficClass[64];
} GSW_QoS_DSCP_ClassCfg_t;
```

#### Parameters

Data Type	Name	Description
u8	nTrafficClass[64]	Traffic class associated with a particular DSCP value. DSCP is the index to an array of resulting traffic class values. The index starts counting from zero.

### 1.10.1.60 GSW\_QoS\_DSCP\_DropPrecedenceCfg\_t

#### Description

DSCP to Drop Precedence assignment table configuration. Used by GSW\_QOS\_DSCP\_DROP\_PRECEDENCE\_CFG\_SET and GSW\_QOS\_DSCP\_DROP\_PRECEDENCE\_CFG\_GET.

#### Prototype

```
struct
{
    u8 nDSCP_DropPrecedence[64];
```

```
} GSW_QoS_DSCP_DropPrecedenceCfg_t;
```

## Parameters

Data Type	Name	Description
u8	nDSCP_DropPrecedence[64]	DSCP to drop precedence assignment. Every array entry represents the drop precedence for one of the 64 existing DSCP values. DSCP is the index to an array of resulting drop precedence values. The index starts counting from zero. Value refers to <a href="#">GSW_QoS_DropPrecedence_t</a> .

### 1.10.1.61 GSW\_QoS\_FlowCtrlCfg\_t

#### Description

Configures the global buffer flow control threshold for conforming and non-conforming packets. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW\_CAP\_GET. Used by GSW\_QOS\_FLOWCTRL\_CFG\_SET and GSW\_QOS\_FLOWCTRL\_CFG\_GET.

#### Prototype

```
struct
{
    u16 nFlowCtrlNonConform_Min;
    u16 nFlowCtrlNonConform_Max;
    u16 nFlowCtrlConform_Min;
    u16 nFlowCtrlConform_Max;
} GSW_QoS_FlowCtrlCfg_t;
```

## Parameters

Data Type	Name	Description
u16	nFlowCtrlNonConform_Min	Global Buffer Non Conforming Flow Control Threshold Minimum [number of segments].
u16	nFlowCtrlNonConform_Max	Global Buffer Non Conforming Flow Control Threshold Maximum [number of segments].
u16	nFlowCtrlConform_Min	Global Buffer Conforming Flow Control Threshold Minimum [number of segments].
u16	nFlowCtrlConform_Max	Global Buffer Conforming Flow Control Threshold Maximum [number of segments].

### 1.10.1.62 GSW\_QoS\_FlowCtrlPortCfg\_t

#### Description

Configures the ingress port flow control threshold for used packet segments. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW\_CAP\_GET. Used by GSW\_QOS\_FLOWCTRL\_PORT\_CFG\_SET and GSW\_QOS\_FLOWCTRL\_PORT\_CFG\_GET.

## Prototype

```
struct
{
    u16 nPortId;
    u16 nFlowCtrl_Min;
    u16 nFlowCtrl_Max;
} GSW_QoS_FlowCtrlPortCfg_t;
```

## Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent.
u16	nFlowCtrl_Min	Ingress Port occupied Buffer Flow Control Threshold Minimum [number of segments].
u16	nFlowCtrl_Max	Ingress Port occupied Buffer Flow Control Threshold Maximum [number of segments].

### 1.10.1.63 GSW\_QoS\_meterCfg\_t

## Description

Configures the parameters of a rate meter instance. Used by GSW\_QOS\_METER\_ALLOC, GSW\_QOS\_METER\_FREE, GSW\_QOS\_METER\_CFG\_SET and GSW\_QOS\_METER\_CFG\_GET.

## Prototype

```
struct
{
    gsw_bool_t bEnable;
    u16 nMeterId;
    char cMeterName[32];
    GSW_QoS_Meter_Type eMtrType;
    u32 nCbs;
    u32 nCbs_ls;
    u32 nEbs;
    u32 nEbs_ls;
    u32 nRate;
    u32 nPiRate;
    u8 nColourBlindMode;
    u8 bPktMode;
    gsw_bool_t bLocalOverhd;
    u16 nLocaloverhd;
} GSW_QoS_meterCfg_t;
```



## Parameters

Data Type	Name	Description
gsw_bool_t	bEnable	Enable/Disable the meter shaper.
u16	nMeterId	Meter index (zero-based counting). <i>Note: For GSW_QOS_METER_FREE, this is the only input and other fields are ignored. For GSW_QOS_METER_ALLOC, this is output when allocation is successful. For GSW_QOS_METER_CFG_SET and GSW_QOS_METER_CFG_GET, this is input to indicate meter to configure/get-configuration.</i>
char	cMeterName[32]	Meter Name string for easy reference (Id to Name Mapping) - TBD
<a href="#">GSW_QoS_Meter_Type</a>	eMtrType	Meter Algorithm Type
u32	nCbs	Committed Burst Size (CBS [Bytes]).
u32	nCbs_Is	Committed Burst Size Exponent (CBS [Bytes]).
u32	nEbs	Excess Burst Size (EBS [Bytes]).
u32	nEbs_Is	Excess Burst Size Exponent (EBS [Bytes]).
u32	nRate	Committed Information Rate (CIR) <i>Note: CIR in [kbit/s] if <a href="#">GSW_QoS_meterCfg_t</a> is FALSE, or in [packet/s] if <a href="#">GSW_QoS_meterCfg_t</a> is TRUE.</i>
u32	nPiRate	Peak Information Rate (PIR) - applicable for trTCM only <i>Note: PIR in [kbit/s] if <a href="#">GSW_QoS_meterCfg_t</a> is FALSE, or in [packet/s] if <a href="#">GSW_QoS_meterCfg_t</a> is TRUE.</i>
u8	nColourBlindMode	Peak Burst Size (PBS [Bytes]) - applicable for trTCM only Meter colour mode
u8	bPktMode	Enable/Disable Packet Mode. 0- Byte, 1 - Pkt
gsw_bool_t	bLocalOverhd	Enable/Disable local overhead for metering rate calculation.
u16	nLocaloverhd	Local overhead for metering rate calculation when <a href="#">GSW_QoS_meterCfg_t</a> is TRUE.

### 1.10.1.64 GSW\_QoS\_PCP\_ClassCfg\_t

#### Description

Traffic class associated with a particular 802.1P (PCP) priority mapping value. This table is global for the entire switch device. Priority map entry structure. Used by GSW\_QOS\_PCP\_CLASS\_SET and GSW\_QOS\_PCP\_CLASS\_GET.

### Prototype

```
struct
{
    u8 nTrafficClass[16];
} GSW_QoS_PCP_ClassCfg_t;
```

### Parameters

Data Type	Name	Description
u8	nTrafficClass[16]	Configures the PCP to traffic class mapping. The queue index starts counting from zero.

## 1.10.1.65 GSW\_QoS\_portCfg\_t

### Description

Describes which priority information of ingress packets is used (taken into account) to identify the packet priority and the related egress priority queue. For DSCP, the priority to queue assignment is done using GSW\_QoS\_DSCP\_CLASS\_SET. For VLAN, the priority to queue assignment is done using GSW\_QoS\_PCP\_CLASS\_SET. Used by GSW\_QoS\_PORT\_CFG\_SET and GSW\_QoS\_PORT\_CFG\_GET.

### Prototype

```
struct
{
    u16 nPortId;
    GSW_QoS_ClassSelect_t eClassMode;
    u8 nTrafficClass;
} GSW_QoS_portCfg_t;
```

### Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.
<a href="#">GSW_QoS_ClassSelect_t</a>	eClassMode	Select the packet header field on which to base the traffic class assignment.
u8	nTrafficClass	Default port priority in case no other priority (such as VLAN-based PCP or IP-based DSCP) is used.

## 1.10.1.66 GSW\_QoS\_portRemarkingCfg\_t

### Description

Port Remarking Configuration. Ingress and Egress remarking options for dedicated packet fields DSCP, CTAG VLAN PCP, STAG VLAN PCP and STAG VLAN DEI. Remarking is done either on the used traffic class or the

drop precedence. Packet field specific remarking only applies on a packet if enabled on ingress and egress port. Used by GSW\_QOS\_PORT\_REMARKING\_CFG\_SET and GSW\_QOS\_PORT\_REMARKING\_CFG\_GET.

### Prototype

```
struct
{
    u16 nPortId;
    GSW_Qos_ingressRemarking_t eDSCP_IngressRemarkingEnable;
    gsw_bool_t bDSCP_EgressRemarkingEnable;
    gsw_bool_t bPCP_IngressRemarkingEnable;
    gsw_bool_t bPCP_EgressRemarkingEnable;
    gsw_bool_t bSTAG_PCP_IngressRemarkingEnable;
    gsw_bool_t bSTAG_DEI_IngressRemarkingEnable;
    gsw_bool_t bSTAG_PCP_DEI_EgressRemarkingEnable;
} GSW_QoS_portRemarkingCfg_t;
```

### Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available.
<a href="#">GSW_Qos_ingressRemarking_t</a>	eDSCP_IngressRemarkingEnable	Ingress DSCP Remarking. Specifies on ingress side how a packet should be remarked. This DSCP remarking only works in case remarking is enabled on the egress port. This configuration requires that remarking is also enabled on the egress port. DSCP remarking enable on either ingress or egress port side does not perform any remark operation.
gsw_bool_t	bDSCP_EgressRemarkingEnable	Egress DSCP Remarking. Applies remarking on egress packets in a fashion as specified on the ingress port. This ingress port remarking is configured by the parameter 'eDSCP_IngressRemarking'. This configuration requires that remarking is also enabled on the ingress port. DSCP remarking enable on either ingress or egress port side does not perform any remark operation.
gsw_bool_t	bPCP_IngressRemarkingEnable	Ingress PCP Remarking. Applies remarking to all port ingress packets. This configuration requires that remarking is also enabled on the egress port. PCP remarking enable on either ingress or egress port side does not perform any remark operation.

Data Type	Name	Description
gsw_bool_t	bPCP_EgressRemarkingEnable	Egress PCP Remarking. Applies remarking for all port egress packets. This configuration requires that remarking is also enabled on the ingress port. PCP remarking enable on either ingress or egress port side does not perform any remark operation.
gsw_bool_t	bSTAG_PCP_IngressRemarkingEnable	Ingress STAG VLAN PCP Remarking
gsw_bool_t	bSTAG_DEI_IngressRemarkingEnable	Ingress STAG VLAN DEI Remarking
gsw_bool_t	bSTAG_PCP_DEI_EgressRemarkingEnable	Egress STAG VLAN PCP & DEI Remarking

### 1.10.1.67 GSW\_QoS\_QueueBufferReserveCfg\_t

#### Description

Reserved egress queue buffer segments. Used by GSW\_QOS\_QUEUE\_BUFFER\_RESERVE\_CFG\_SET and GSW\_QOS\_QUEUE\_BUFFER\_RESERVE\_CFG\_GET.

#### Prototype

```
struct
{
    u16 nQueueId;
    u16 nBufferReserved;
} GSW_QoS_QueueBufferReserveCfg_t;
```

#### Parameters

Data Type	Name	Description
u16	nQueueId	QoS queue index (zero-based counting). This is an input parameter for GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_GET.
u16	nBufferReserved	Reserved Buffer Segment Threshold [number of segments]. This is an output parameter for GSW_QOS_QUEUE_BUFFER_RESERVE_CFG_GET.

### 1.10.1.68 GSW\_QoS\_queuePort\_t

#### Description

Sets the Queue ID for one traffic class of one port. Used by GSW\_QOS\_QUEUE\_PORT\_SET and GSW\_QOS\_QUEUE\_PORT\_GET.

#### Prototype

```
struct
```

```
{
    u16 nPortId;
    gsw_bool_t bExtractionEnable;
    GSW_QoS_qMapMode_t eQMapMode;
    u8 nTrafficClassId;
    u8 nQueueId;
    gsw_bool_t bRedirectionBypass;
    u8 nRedirectPortId;
    gsw_bool_t bEnableIngressPceBypass;
    gsw_bool_t bReservedPortMode;
} GSW_QoS_queuePort_t;
```

### Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available. This is an input parameter for GSW_QOS_QUEUE_PORT_GET.
gsw_bool_t	bExtractionEnable	Forward CPU (extraction) before external QoS queueing (DownMEP). GSWIP-3.1 only.
<a href="#">GSW_QoS_qMapMode_t</a>	eQMapMode	When <a href="#">GSW_QoS_queuePort_t</a> is FALSE, this field defines Queue Mapping Mode. GSWIP-3.1 only.
u8	nTrafficClassId	Traffic Class index (zero-based counting). This is an input parameter for GSW_QOS_QUEUE_PORT_GET.
u8	nQueueId	QoS queue index (zero-based counting). This is an output parameter for GSW_QOS_QUEUE_PORT_GET.
gsw_bool_t	bRedirectionBypass	Queue Redirection bypass Option. If enabled, all packets destined to 'nQueueId' are redirected from the 'nPortId' to 'nRedirectPortId'. This is used for 2nd stage of FULL QoS Path, where the packet has completed QoS process at CBM/CQEM and been injected into GSWIP again.
u8	nRedirectPortId	Redirected traffic forward port. All egress packets to 'nPortId' are redirected to "nRedirectPortId". If there is no redirection required, it should be same as "nPortId". GSWIP-3.0/3.1 only.

Data Type	Name	Description
gsw_bool_t	bEnableIngressPceBypass	To enable Ingress PCE Bypass. Applicable for GSWIP 3.2 and above. For GSW_QoS_QueuePortGet, set TRUE as input to check whether Ingress PCE Bypass is enabled, and this field is updated as output. For GSW_QoS_QueuePortSet, set FALSE to configure normal path first, then set TRUE to configure Ingress PCE Bypass path (only if application requires).
gsw_bool_t	bReservedPortMode	Internal purpose only - user not allowed to use it. Applicable for GSWIP 3.2 and above.

### 1.10.1.69 GSW\_QoS\_schedulerCfg\_t

#### Description

Configures the egress queues attached to a single port, and that are scheduled to transmit the queued Ethernet packets. Used by GSW\_QOS\_SCHEDULER\_CFG\_SET and GSW\_QOS\_SCHEDULER\_CFG\_GET.

#### Prototype

```
struct
{
    u8 nQueueId;
    u8 eType;
    u16 nWeight;
} GSW_QoS_schedulerCfg_t;
```

#### Parameters

Data Type	Name	Description
u8	nQueueId	QoS queue index (zero-based counting).
u8	eType	Scheduler Type (Strict Priority/Weighted Fair Queuing). Refers to <a href="#">GSW_QoS_Scheduler_t</a> for detail values.
u16	nWeight	Weight in Token. Parameter used for WFQ configuration. Sets the weight in token in relation to all remaining queues on this egress port having WFQ configuration. This parameter is only used when 'eType=GSW_QOS_SCHEDULER_WFQ'.

### 1.10.1.70 GSW\_QoS\_ShaperCfg\_t

#### Description

Configures a rate shaper instance with the rate and the burst size. Used by GSW\_QOS\_SHAPER\_CFG\_SET and GSW\_QOS\_SHAPER\_CFG\_GET.

### Prototype

```
struct
{
    u8 nRateShaperId;
    gsw_bool_t bEnable;
    gsw_bool_t bAVB;
    u32 nCbs_ls;
    u32 nCbs;
    u32 nRate;
} GSW_QoS_ShaperCfg_t;
```

### Parameters

Data Type	Name	Description
u8	nRateShaperId	Rate shaper index (zero-based counting).
gsw_bool_t	bEnable	Enable/Disable the rate shaper.
gsw_bool_t	bAVB	802.1Qav credit based shaper mode. This specific shaper algorithm mode is used by the audio/video bridging (AVB) network (according to 802.1Qav). By default, a token based shaper algorithm is used.
u32	nCbs_ls	Committed Burst Size Exponent (CBS [Bytes]).
u32	nCbs	Committed Burst Size (CBS [bytes])
u32	nRate	Rate [kbit/s]

#### 1.10.1.71 GSW\_QoS\_ShaperQueue\_t

### Description

Assign one rate shaper instance to a QoS queue. Used by GSW\_QOS\_SHAPER\_QUEUE\_ASSIGN and GSW\_QOS\_SHAPER\_QUEUE\_DEASSIGN.

### Prototype

```
struct
{
    u8 nRateShaperId;
    u8 nQueueId;
} GSW_QoS_ShaperQueue_t;
```

### Parameters

Data Type	Name	Description
u8	nRateShaperId	Rate shaper index (zero-based counting).
u8	nQueueId	QoS queue index (zero-based counting).

### 1.10.1.72 GSW\_QoS\_ShaperQueueGet\_t

#### Description

Retrieve if a rate shaper instance is assigned to a QoS egress queue. Used by GSW\_QOS\_SHAPER\_QUEUE\_GET.

#### Prototype

```
struct
{
    u8 nQueueId;
    gsw_bool_t bAssigned;
    u8 nRateShaperId;
} GSW_QoS_ShaperQueueGet_t;
```

#### Parameters

Data Type	Name	Description
u8	nQueueId	QoS queue index (zero-based counting). This parameter is the input parameter for the GET function.
gsw_bool_t	bAssigned	Rate shaper instance assigned. If 1, a rate shaper instance is assigned to the queue. Otherwise no shaper instance is assigned.
u8	nRateShaperId	Rate shaper index (zero-based counting). Only a valid instance is returned in case 'bAssigned == 1'.

### 1.10.1.73 GSW\_QoS\_stormCfg\_t

#### Description

Assigns one meter instances for storm control. Used by GSW\_QOS\_STORM\_CFG\_SET and GSW\_QOS\_STORM\_CFG\_GET. Not applicable to GSWIP-3.1.

#### Prototype

```
struct
{
    u16 nMeterId;
    gsw_bool_t bBroadcast;
    gsw_bool_t bMulticast;
    gsw_bool_t bUnknownUnicast;
} GSW_QoS_stormCfg_t;
```

#### Parameters

Data Type	Name	Description
u16	nMeterId	Meter index 0 (zero-based counting).
gsw_bool_t	bBroadcast	Meter instances used for broadcast traffic.



Data Type	Name	Description
gsw_bool_t	bMulticast	Meter instances used for multicast traffic.
gsw_bool_t	bUnknownUnicast	Meter instances used for unknown unicast traffic.

#### 1.10.1.74 GSW\_QoS\_SVLAN\_PCP\_ClassCfg\_t

##### Description

Traffic class associated with a particular STAG VLAN 802.1P (PCP) priority and Drop Eligible Indicator (DEI) mapping value. This table is global for the entire switch device. Priority map entry structure. The table index value is calculated by 'index=PCP + 8\*DEI' Used by GSW\_QOS\_SVLAN\_PCP\_CLASS\_SET and GSW\_QOS\_SVLAN\_PCP\_CLASS\_GET.

##### Prototype

```
struct
{
    u8 nTrafficClass[16];
    u8 nTrafficColor[16];
    u8 nPCP_Remark_Enable[16];
    u8 nDEI_Remark_Enable[16];
} GSW_QoS_SVLAN_PCP_ClassCfg_t;
```

##### Parameters

Data Type	Name	Description
u8	nTrafficClass[16]	Configures the PCP and DEI to traffic class mapping. The queue index starts counting from zero.
u8	nTrafficColor[16]	Configures the PCP traffic color. Not applicable to GSWIP-3.1.
u8	nPCP_Remark_Enable[16]	PCP Remark disable control. Not applicable to GSWIP-3.1.
u8	nDEI_Remark_Enable[16]	DEI Remark disable control. Not applicable to GSWIP-3.1.

#### 1.10.1.75 GSW\_QoS\_WRED\_Cfg\_t

##### Description

Configures the global probability profile of the device. The min. and max. threshold values are given in number of packet buffer segments and required only in case of Manual Mode. The GSWIP-3.0/3.1 supports Auto mode and the threshold values are dynamically computed internally by GSWIP. The size of a segment can be retrieved using GSW\_CAP\_GET. Used by GSW\_QOS\_WRED\_CFG\_SET and GSW\_QOS\_WRED\_CFG\_GET.

##### Prototype

```
struct
{
    GSW_QoS_WRED_WATERMARK_t eCongestionWatermark;
```

```

GSW_QoS_WRED_Profile_t eProfile;
GSW_QoS_WRED_Mode_t eMode;
GSW_QoS_WRED_ThreshMode_t eThreshMode;
u16 nRed_Min;
u16 nRed_Max;
u16 nYellow_Min;
u16 nYellow_Max;
u16 nGreen_Min;
u16 nGreen_Max;
} GSW_QoS_WRED_Cfg_t;

```

### Parameters

Data Type	Name	Description
<a href="#">GSW_QoS_WRED_WATTERM ARK_t</a>	eCongestionWatermark	Egress Queue Congestion Notification Watermark only applicable for GSWIP 3.1
<a href="#">GSW_QoS_WRED_Profile_t</a>	eProfile	Drop Probability Profile.
<a href="#">GSW_QoS_WRED_Mode_t</a>	eMode	Automatic or Manual Mode of Thresholds Config
<a href="#">GSW_QoS_WRED_ThreshM ode_t</a>	eThreshMode	WRED Threshold Mode Config
u16	nRed_Min	WRED Red Threshold Min [number of segments] - Valid for Manual Mode only.
u16	nRed_Max	WRED Red Threshold Max [number of segments] - Valid for Manual Mode only
u16	nYellow_Min	WRED Yellow Threshold Min [number of segments] - Valid for Manual Mode only
u16	nYellow_Max	WRED Yellow Threshold Max [number of segments] - Valid for Manual Mode only
u16	nGreen_Min	WRED Green Threshold Min [number of segments] - Valid for Manual Mode only
u16	nGreen_Max	WRED Green Threshold Max [number of segments] - Valid for Manual Mode only

### 1.10.1.76 GSW\_QoS\_WRED\_PortCfg\_t

#### Description

Configures the WRED threshold parameter per port. The configured thresholds apply to fill level sum of all egress queues which are assigned to the egress port. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW\_CAP\_GET. Used by GSW\_QOS\_WRED\_PORT\_CFG\_SET and GSW\_QOS\_WRED\_PORT\_CFG\_GET.

#### Prototype

```

struct
{

```

```

    u16 nPortId;
    u16 nRed_Min;
    u16 nRed_Max;
    u16 nYellow_Min;
    u16 nYellow_Max;
    u16 nGreen_Min;
    u16 nGreen_Max;
} GSW_QoS_WRED_PortCfg_t;

```

### Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent.
u16	nRed_Min	WRED Red Threshold Min [number of segments].
u16	nRed_Max	WRED Red Threshold Max [number of segments].
u16	nYellow_Min	WRED Yellow Threshold Min [number of segments].
u16	nYellow_Max	WRED Yellow Threshold Max [number of segments].
u16	nGreen_Min	WRED Green Threshold Min [number of segments].
u16	nGreen_Max	WRED Green Threshold Max [number of segments].

### 1.10.1.77 GSW\_QoS\_WRED\_QueueCfg\_t

#### Description

Configures the WRED threshold level values. The min. and max. values are given in number of packet buffer segments. The size of a segment can be retrieved using GSW\_CAP\_GET. Used by GSW\_QOS\_WRED\_QUEUE\_CFG\_SET and GSW\_QOS\_WRED\_QUEUE\_CFG\_GET.

#### Prototype

```

struct
{
    u16 nQueueId;
    u16 nRed_Min;
    u16 nRed_Max;
    u16 nYellow_Min;
    u16 nYellow_Max;
    u16 nGreen_Min;
    u16 nGreen_Max;
    u16 nReserveThreshold;
} GSW_QoS_WRED_QueueCfg_t;

```

## Parameters

Data Type	Name	Description
u16	nQueueId	QoS queue index (zero-based counting).
u16	nRed_Min	WRED Red Threshold Min [number of segments].
u16	nRed_Max	WRED Red Threshold Max [number of segments].
u16	nYellow_Min	WRED Yellow Threshold Min [number of segments].
u16	nYellow_Max	WRED Yellow Threshold Max [number of segments].
u16	nGreen_Min	WRED Green Threshold Min [number of segments].
u16	nGreen_Max	WRED Green Threshold Max [number of segments].
u16	nReserveThreshold	Reserved Buffer Threshold

### 1.10.1.78 GSW\_register\_mod\_t

#### Description

Register access parameter to directly modify internal registers. Used by GSW\_REGISTER\_MOD.

#### Prototype

```
struct
{
    u16 nRegAddr;
    u16 nData;
    u16 nMask;
} GSW_register_mod_t;
```

## Parameters

Data Type	Name	Description
u16	nRegAddr	Register Address Offset for modification.
u16	nData	Value to write to 'nRegAddr'.
u16	nMask	Mask of bits to be modified. 1 to modify, 0 to ignore.

### 1.10.1.79 GSW\_register\_t

#### Description

Register access parameter to directly read or write switch internal registers. Used by GSW\_REGISTER\_SET and GSW\_REGISTER\_GET.

### Prototype

```
struct
{
    u16 nRegAddr;
    u16 nData;
} GSW_register_t;
```

### Parameters

Data Type	Name	Description
u16	nRegAddr	Register Address Offset for read or write access.
u16	nData	Value to write to or read from 'nRegAddr'.

### 1.10.1.80 GSW\_RMON\_clear\_t

#### Description

RMON Counters Data Structure for clearance of values. Used by GSW\_RMON\_CLEAR.

### Prototype

```
struct
{
    GSW_RMON_type_t eRmonType;
    u8 nRmonId;
} GSW_RMON_clear_t;
```

### Parameters

Data Type	Name	Description
<a href="#">GSW_RMON_type_t</a>	eRmonType	RMON Counters Type
u8	nRmonId	RMON Counters Identifier - Meter, Port, If, Route, etc.

### 1.10.1.81 GSW\_RMON\_flowGet\_t

#### Description

Hardware platform extended RMON Counters. GSWIP-3.1 only. This structure contains additional RMON counters. These counters can be used by the packet classification engine and can be freely assigned to dedicated packet rules and flows. Used by GSW\_RMON\_FLOW\_GET.

### Prototype

```
struct
{
    gsw_bool_t bIndex;
    u16 nIndex;
    u16 nPortId;
    u16 nFlowId;
```

```

    u32 nRxPkts;
    u32 nTxPkts;
    u32 nTxPceBypassPkts;
} GSW_RMON_flowGet_t;

```

## Parameters

Data Type	Name	Description
gsw_bool_t	bIndex	If TRUE, use <a href="#">GSW_RMON_flowGet_t</a> to access the Flow Counter, otherwise, use GSW_TFLOW_COUNT_MODE_GET to determine mode and use <a href="#">GSW_RMON_flowGet_t</a> and <a href="#">GSW_RMON_flowGet_t</a> to calculate index of the Flow Counter.
u16	nIndex	Absolute index of Flow Counter.
u16	nPortId	Port ID. This could be Logical Port, CTP or Bridge Port. It depends on the mode set by GSW_TFLOW_COUNT_MODE_SET.
u16	nFlowId	. The range depends on the mode set by GSW_TFLOW_COUNT_MODE_SET.
u32	nRxPkts	Rx Packet Counter
u32	nTxPkts	Tx Packet Counter (non-PCE-Bypass)
u32	nTxPceBypassPkts	Tx Packet Counter (PCE-Bypass)

### 1.10.1.82 GSW\_RMON\_Meter\_cnt\_t

#### Description

RMON Counters for Meter - Type (GSWIP-3.0 only). This structure contains the RMON counters of one Meter Instance. Used by GSW\_RMON\_METER\_GET.

#### Prototype

```

struct
{
    u8 nMeterId;
    u32 nGreenCount;
    u32 nYellowCount;
    u32 nRedCount;
    u32 nResCount;
} GSW_RMON_Meter_cnt_t;

```

## Parameters

Data Type	Name	Description
u8	nMeterId	Meter Instance number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected meter is not available. This parameter specifies for which Meter Id the RMON-1 counter is read. It has to be set by the application before calling GSW_RMON_METER_GET.
u32	nGreenCount	Metered Green colored packets or bytes (depending upon mode) count.
u32	nYellowCount	Metered Yellow colored packets or bytes (depending upon mode) count.
u32	nRedCount	Metered Red colored packets or bytes (depending upon mode) count.
u32	nResCount	Metered Reserved (Future Use) packets or bytes (depending upon mode) count.

### 1.10.1.83 GSW\_RMON\_mode\_t

#### Description

RMON Counters Mode for different Elements. This structure takes RMON Counter Element Name and mode config.

#### Prototype

```
struct
{
    GSW_RMON_type_t eRmonType;
    GSW_RMON_CountMode_t eCountMode;
} GSW_RMON_mode_t;
```

#### Parameters

Data Type	Name	Description
<a href="#">GSW_RMON_type_t</a>	eRmonType	RMON Counters Type
<a href="#">GSW_RMON_CountMode_t</a>	eCountMode	RMON Counters Mode

### 1.10.1.84 GSW\_RMON\_Port\_cnt\_t

#### Description

RMON Counters for individual Port. This structure contains the RMON counters of an Ethernet Switch Port. Used by GSW\_RMON\_PORT\_GET.

#### Prototype

```
struct
{
```

```
GSW_portType_t ePortType;
u16 nPortId;
u16 nSubIfIdGroup;
gsw_bool_t bPceBypass;
u32 nRxExtendedVlanDiscardPkts;
u32 nMtuExceedDiscardPkts;
u32 nTxUnderSizeGoodPkts;
u32 nTxOversizeGoodPkts;
u32 nRxGoodPkts;
u32 nRxUnicastPkts;
u32 nRxBroadcastPkts;
u32 nRxMulticastPkts;
u32 nRxFCSErrorPkts;
u32 nRxUnderSizeGoodPkts;
u32 nRxOversizeGoodPkts;
u32 nRxUnderSizeErrorPkts;
u32 nRxGoodPausePkts;
u32 nRxOversizeErrorPkts;
u32 nRxAlignErrorPkts;
u32 nRxFilteredPkts;
u32 nRx64BytePkts;
u32 nRx127BytePkts;
u32 nRx255BytePkts;
u32 nRx511BytePkts;
u32 nRx1023BytePkts;
u32 nRxMaxBytePkts;
u32 nTxGoodPkts;
u32 nTxUnicastPkts;
u32 nTxBroadcastPkts;
u32 nTxMulticastPkts;
u32 nTxSingleCollCount;
u32 nTxMultCollCount;
u32 nTxLateCollCount;
u32 nTxExcessCollCount;
u32 nTxCollCount;
u32 nTxPauseCount;
u32 nTx64BytePkts;
u32 nTx127BytePkts;
u32 nTx255BytePkts;
u32 nTx511BytePkts;
u32 nTx1023BytePkts;
u32 nTxMaxBytePkts;
u32 nTxDroppedPkts;
u32 nTxAcnDroppedPkts;
u32 nRxDroppedPkts;
u64 nRxGoodBytes;
u64 nRxBadBytes;
u64 nTxGoodBytes;
} GSW_RMON_Port_cnt_t;
```



## Parameters

Data Type	Name	Description
<a href="#">GSW_portType_t</a>	ePortType	Port Type. This gives information which type of port to get RMON. nPortId should be based on this field. This is new in GSWIP-3.1. For GSWIP-2.1/2.2/3.0, this field is always ZERO (GSW_LOGICAL_PORT).
u16	nPortId	Ethernet Port number (zero-based counting). The valid range is hardware dependent. An error code is delivered if the selected port is not available. This parameter specifies for which MAC port the RMON counter is read. It has to be set by the application before calling GSW_RMON_PORT_GET.
u16	nSubIfIdGroup	Sub interface ID group. The valid range is hardware/protocol dependent. <i>Note: This field is valid when <a href="#">GSW_RMON_Port_cnt_t</a> is <a href="#">GSW_portType_t</a>. Sub interface ID group is defined for each of <a href="#">GSW_LogicalPortMode_t</a>. For both <a href="#">GSW_LOGICAL_PORT_8BIT_WLAN</a> and <a href="#">GSW_LOGICAL_PORT_9BIT_WLAN</a>, this field is VAP. For <a href="#">GSW_LOGICAL_PORT_GPON</a>, this field is GEM index. For <a href="#">GSW_LOGICAL_PORT_EPON</a>, this field is stream index. For <a href="#">GSW_LOGICAL_PORT_GINT</a>, this field is LLID. For others, this field is 0.</i>
gsw_bool_t	bPceBypass	Separate set of CTP Tx counters when PCE is bypassed. GSWIP-3.1 only.
u32	nRxExtendedVlanDiscardPkts	Discarded at Extended VLAN Operation Packet Count. GSWIP-3.1 only.
u32	nMtuExceedDiscardPkts	Discarded MTU Exceeded Packet Count. GSWIP-3.1 only.
u32	nTxUnderSizeGoodPkts	Tx Undersize (<64) Packet Count. GSWIP-3.1 only.
u32	nTxOversizeGoodPkts	Tx Oversize (>1518) Packet Count. GSWIP-3.1 only.
u32	nRxGoodPkts	Receive Packet Count (only packets that are accepted and not discarded).
u32	nRxUnicastPkts	Receive Unicast Packet Count.
u32	nRxBroadcastPkts	Receive Broadcast Packet Count.
u32	nRxMulticastPkts	Receive Multicast Packet Count.

Module Reference

Data Type	Name	Description
u32	nRxFCSErrorPkts	Receive FCS Error Packet Count.
u32	nRxUnderSizeGoodPkts	Receive Undersize Good Packet Count.
u32	nRxOversizeGoodPkts	Receive Oversize Good Packet Count.
u32	nRxUnderSizeErrorPkts	Receive Undersize Error Packet Count.
u32	nRxGoodPausePkts	Receive Good Pause Packet Count.
u32	nRxOversizeErrorPkts	Receive Oversize Error Packet Count.
u32	nRxAlignErrorPkts	Receive Align Error Packet Count.
u32	nRxFilteredPkts	Filtered Packet Count.
u32	nRx64BytePkts	Receive Size 64 Bytes Packet Count.
u32	nRx127BytePkts	Receive Size 65-127 Bytes Packet Count.
u32	nRx255BytePkts	Receive Size 128-255 Bytes Packet Count.
u32	nRx511BytePkts	Receive Size 256-511 Bytes Packet Count.
u32	nRx1023BytePkts	Receive Size 512-1023 Bytes Packet Count.
u32	nRxMaxBytePkts	Receive Size 1024-1522 Bytes (or more, if configured) Packet Count.
u32	nTxGoodPkts	Overall Transmit Good Packets Count.
u32	nTxUnicastPkts	Transmit Unicast Packet Count.
u32	nTxBroadcastPkts	Transmit Broadcast Packet Count.
u32	nTxMulticastPkts	Transmit Multicast Packet Count.
u32	nTxSingleCollCount	Transmit Single Collision Count.
u32	nTxMultCollCount	Transmit Multiple Collision Count.
u32	nTxLateCollCount	Transmit Late Collision Count.
u32	nTxExcessCollCount	Transmit Excessive Collision Count.
u32	nTxCollCount	Transmit Collision Count.
u32	nTxPauseCount	Transmit Pause Packet Count.
u32	nTx64BytePkts	Transmit Size 64 Bytes Packet Count.
u32	nTx127BytePkts	Transmit Size 65-127 Bytes Packet Count.
u32	nTx255BytePkts	Transmit Size 128-255 Bytes Packet Count.
u32	nTx511BytePkts	Transmit Size 256-511 Bytes Packet Count.
u32	nTx1023BytePkts	Transmit Size 512-1023 Bytes Packet Count.
u32	nTxMaxBytePkts	Transmit Size 1024-1522 Bytes (or more, if configured) Packet Count.
u32	nTxDroppedPkts	Transmit Drop Packet Count.
u32	nTxAcmdDroppedPkts	Transmit Dropped Packet Count, based on Congestion Management.
u32	nRxDroppedPkts	Receive Dropped Packet Count.
u64	nRxGoodBytes	Receive Good Byte Count (64 bit).
u64	nRxBadBytes	Receive Bad Byte Count (64 bit).
u64	nTxGoodBytes	Transmit Good Byte Count (64 bit).

### 1.10.1.85 GSW\_STP\_BPDU\_Rule\_t

#### Description

Spanning tree packet detection and forwarding. Used by GSW\_STP\_BPDU\_RULE\_SET and GSW\_STP\_BPDU\_RULE\_GET.

#### Prototype

```
struct
{
    GSW_portForward_t eForwardPort;
    u8 nForwardPortId;
} GSW_STP_BPDU_Rule_t;
```

#### Parameters

Data Type	Name	Description
<a href="#">GSW_portForward_t</a>	eForwardPort	Filter spanning tree packets and forward them, discard them or disable the filter.
u8	nForwardPortId	Target (bridge) port for forwarded packets; only used if selected by 'eForwardPort'. Forwarding is done if 'eForwardPort = GSW_PORT_FORWARD_PORT'.

### 1.10.1.86 GSW\_STP\_portCfg\_t

#### Description

Configures the Spanning Tree Protocol state of an Ethernet port. Used by GSW\_STP\_PORT\_CFG\_SET and GSW\_STP\_PORT\_CFG\_GET.

#### Prototype

```
struct
{
    u16 nPortId;
    u16 nFId;
    GSW_STP_PortState_t ePortState;
} GSW_STP_portCfg_t;
```

## Parameters

Data Type	Name	Description
u16	nPortId	Ethernet Port number (zero-based counting) in GSWIP-2.1/2.2/3.0. From GSWIP-3.1, this field is Bridge Port ID. The valid range is hardware dependent. An error code is delivered if the selected port is not available.
u16	nFid	Filtering Identifier (FID) (not supported by all switches). The FID allows to keep multiple STP states per physical Ethernet port. Multiple CTAG VLAN groups could be assigned to one FID and therefore share the same STP port state. Switch API ignores the FID value in case the switch device does not support it or switch CTAG VLAN awareness is disabled.
<a href="#">GSW_STP_PortState_t</a>	ePortState	Spanning Tree Protocol state of the port.

### 1.10.1.87 GSW\_TflowCmodeConf\_t

#### Description

Hardware platform TFLOW counter mode. Supported modes include, Global (default), Logical, CTP, Bridge port mode. The number of counters that can be assigned varies based these mode type. Used by GSW\_TFLOW\_COUNT\_MODE\_SET and GSW\_TFLOW\_COUNT\_MODE\_GET.

#### Prototype

```
struct
{
    GSW_TflowCountConfType_t eCountType;
    GSW_TflowCmodeType_t eCountMode;
    u16 nPortMsb;
    GSW_TflowCtpValBits_t nCtpLsb;
    GSW_TflowBrpValBits_t nBrpLsb;
} GSW_TflowCmodeConf_t;
```

#### Parameters

Data Type	Name	Description
GSW_TflowCountConfType_t	eCountType	
GSW_TflowCmodeType_t	eCountMode	
u16	nPortMsb	
GSW_TflowCtpValBits_t	nCtpLsb	
GSW_TflowBrpValBits_t	nBrpLsb	

### 1.10.1.88 GSW\_trunkingCfg\_t

#### Description

Global Ethernet trunking configuration. Used by GSW\_TRUNKING\_CFG\_GET and GSW\_TRUNKING\_CFG\_SET.

#### Prototype

```
struct
{
    gsw_bool_t bIP_Src;
    gsw_bool_t bIP_Dst;
    gsw_bool_t bMAC_Src;
    gsw_bool_t bMAC_Dst;
    gsw_bool_t bSrc_Port;
    gsw_bool_t bDst_Port;
} GSW_trunkingCfg_t;
```

#### Parameters

Data Type	Name	Description
gsw_bool_t	bIP_Src	IP source address is used by the hash algorithm to calculate the egress trunking port index.
gsw_bool_t	bIP_Dst	IP destination address is used by the hash algorithm to calculate the egress trunking port index.
gsw_bool_t	bMAC_Src	MAC source address is used by the hash algorithm to calculate the egress trunking port index.
gsw_bool_t	bMAC_Dst	MAC destination address is used by the hash algorithm to calculate the egress trunking port index.
gsw_bool_t	bSrc_Port	TCP/UDP Source Port is used by the hash algorithm to calculate the egress trunking port index.
gsw_bool_t	bDst_Port	TCP/UDP Destination Port is used by the hash algorithm to calculate the egress trunking port index.

### 1.10.1.89 mdio\_relay\_data

#### Description

Struct defined for MDIO Relay Data Structure

#### Prototype

```
struct
{
```

```
uint16_t data;
uint8_t phy;
uint8_t mmd;
uint16_t reg;
} mdio_relay_data;
```

### Parameters

Data Type	Name	Description
uint16_t	data	data to be read or written
uint8_t	phy	PHY index (0~7) for internal PHY PHY address (0~31) for external PHY access via MDIO bus
uint8_t	mmd	MMD device (0~31)
uint16_t	reg	Register Index 0~31 if mmd is 0 (CL22) 0~65535 otherwise (CL45)

### 1.10.1.90 mdio\_relay\_mod\_data

#### Description

Struct defined for MDIO Relay Mode Data Structure

#### Prototype

```
struct
{
    uint16_t data;
    uint8_t phy;
    uint8_t mmd;
    uint16_t reg;
    uint16_t mask;
} mdio_relay_mod_data;
```

### Parameters

Data Type	Name	Description
uint16_t	data	data to be written with mask
uint8_t	phy	PHY index (0~7) for internal PHY PHY address (0~31) for external PHY access via MDIO bus.
uint8_t	mmd	MMD device (0~31)
uint16_t	reg	Register Index 0~31 if mmd is 0 (CL22) 0~65535 otherwise (CL45)
uint16_t	mask	mask of bit fields to be updated 1 to write the bit 0 to ignore

### 1.10.2 Union Reference

This chapter contains the Union reference.

Table 39 Union Overview

Name	Description
<a href="#">GSW_IP_t</a>	This is a union to describe the IPv4 and IPv6 Address in numeric representation. Used by multiple Structures and APIs. The member selection would be based upon GSW_IP_Select_t.

### 1.10.2.1 GSW\_IP\_t

#### Description

This is a union to describe the IPv4 and IPv6 Address in numeric representation. Used by multiple Structures and APIs. The member selection would be based upon GSW\_IP\_Select\_t.

#### Prototype

```
union
{
    u32 nIPv4;
    u16 nIPv6[8];
} GSW_IP_t;
```

#### Parameters

Data Type	Name	Description
u32	nIPv4	Describe the IPv4 address. Only used if the IPv4 address should be read or configured. Cannot be used together with the IPv6 address fields.
u16	nIPv6[8]	Describe the IPv6 address. Only used if the IPv6 address should be read or configured. Cannot be used together with the IPv4 address fields.

### 1.10.3 Type Definition Reference

This chapter contains the Type Definition reference.

Table 40 Type Definition Overview

Name	Description
<a href="#">gsw_bool_t</a>	This is the boolean datatype.
<a href="#">i16</a>	This is the signed 16-bit datatype.
<a href="#">i32</a>	This is the signed 32-bit datatype.
<a href="#">i64</a>	This is the signed 64-bit datatype.
<a href="#">i8</a>	This is the signed 8-bit datatype.
<a href="#">s32</a>	This is the signed 8-bit datatype.
<a href="#">s8</a>	This is the signed 8-bit datatype.
<a href="#">u16</a>	This is the unsigned 16-bit datatype.
<a href="#">u32</a>	This is the unsigned 32-bit datatype.

Table 40 Type Definition Overview (cont'd)

Name	Description
<b>u64</b>	This is the unsigned 64-bit datatype.
<b>u8</b>	This is the unsigned 8-bit datatype.

### 1.10.3.1 u64

#### Prototype

```
typedef uint64_t u64;
```

#### Parameters

Data Type	Name	Description
uint64_t	u64	This is the unsigned 64-bit datatype.

### 1.10.3.2 u32

#### Prototype

```
typedef uint32_t u32;
```

#### Parameters

Data Type	Name	Description
uint32_t	u32	This is the unsigned 32-bit datatype.

### 1.10.3.3 u16

#### Prototype

```
typedef uint16_t u16;
```

#### Parameters

Data Type	Name	Description
uint16_t	u16	This is the unsigned 16-bit datatype.

### 1.10.3.4 u8

#### Prototype

```
typedef uint8_t u8;
```

#### Parameters

Data Type	Name	Description
uint8_t	u8	This is the unsigned 8-bit datatype.



### 1.10.3.5 i64

#### Prototype

```
typedef int64_t i64;
```

#### Parameters

Data Type	Name	Description
int64_t	i64	This is the signed 64-bit datatype.

### 1.10.3.6 i32

#### Prototype

```
typedef int32_t i32;
```

#### Parameters

Data Type	Name	Description
int32_t	i32	This is the signed 32-bit datatype.

### 1.10.3.7 i16

#### Prototype

```
typedef int16_t i16;
```

#### Parameters

Data Type	Name	Description
int16_t	i16	This is the signed 16-bit datatype.

### 1.10.3.8 i8

#### Prototype

```
typedef int8_t i8;
```

#### Parameters

Data Type	Name	Description
int8_t	i8	This is the signed 8-bit datatype.

### 1.10.3.9 s32

#### Prototype

```
typedef int32_t s32;
```

## Parameters

Data Type	Name	Description
int32_t	s32	This is the signed 8-bit datatype.

### 1.10.3.10 s8

#### Prototype

```
typedef int8_t s8;
```

## Parameters

Data Type	Name	Description
int8_t	s8	This is the signed 8-bit datatype.

### 1.10.3.11 gsw\_bool\_t

#### Prototype

```
typedef uint8_t gsw_bool_t;
```

## Parameters

Data Type	Name	Description
uint8_t	gsw_bool_t	This is the boolean datatype.

## 1.10.4 Constant Reference

This chapter contains the Constant reference.

**Table 41** Constant Overview

Name	Value	Description
MII_ADDR_C45	(1<<30)	Flag to enable 21-bit IEEE 802.3ae Clause 45 addressing mode.
LINK_MODE_MASK	(1ULL << (LINK_MODE_##base_name##_BIT))	Wrapper for link mode used by Supported Link Mode and Advertised Link Mode macros.
GPY2XX_SUPPORTED_10baseT_Half	LINK_MODE_MASK(10baseT_Half)	Macros used by supported in gpy2xx_link.
GPY2XX_SUPPORTED_10baseT_Full	LINK_MODE_MASK(10baseT_Full)	10M full-duplex twisted-pair
GPY2XX_SUPPORTED_100baseT_Half	LINK_MODE_MASK(100baseT_Half)	100M half-duplex twisted-pair

**Table 41 Constant Overview** (cont'd)

Name	Value	Description
GPY2XX_SUPPORTED_100baseT_Full	LINK_MODE_MASK(100baseT_Full)	100M full-duplex twisted-pair
GPY2XX_SUPPORTED_1000baseT_Half	LINK_MODE_MASK(1000baseT_Half)	1G half-duplex twisted-pair
GPY2XX_SUPPORTED_1000baseT_Full	LINK_MODE_MASK(1000baseT_Full)	1G full-duplex twisted-pair
GPY2XX_SUPPORTED_Autoneg	LINK_MODE_MASK(Autoneg)	Auto-negotiation.
GPY2XX_SUPPORTED_TP	LINK_MODE_MASK(TP)	Twisted-pair.
GPY2XX_SUPPORTED_MII	LINK_MODE_MASK(MII)	Media-independent interface.
GPY2XX_SUPPORTED_Pause	LINK_MODE_MASK(Pause)	Pause supported.
GPY2XX_SUPPORTED_Asym_Pause	LINK_MODE_MASK(Asym_Pause)	Asymmetric-pause supported
GPY2XX_SUPPORTED_2500baseT_Full	LINK_MODE_MASK(2500baseT_Full)	2.5G full-duplex twisted-pair
GPY2XX_SUPPORTED_5000baseT_Full	LINK_MODE_MASK(5000baseT_Full)	5G full-duplex twisted-pair
GPY2XX_SUPPORTED_2500baseT_FR	LINK_MODE_MASK(2500baseT_FR)	2.5G Base-T fast retrain
GPY2XX_SUPPORTED_5000baseT_FR	LINK_MODE_MASK(5000baseT_FR)	5G Base-T fast retrain
GPY2XX_ADVERTISED_10baseT_Half	LINK_MODE_MASK(10baseT_Half)	Macros used by advertising in gpy2xx_link.
GPY2XX_ADVERTISED_10baseT_Full	LINK_MODE_MASK(10baseT_Full)	10M full-duplex twisted-pair
GPY2XX_ADVERTISED_100baseT_Half	LINK_MODE_MASK(100baseT_Half)	100M half-duplex twisted-pair
GPY2XX_ADVERTISED_100baseT_Full	LINK_MODE_MASK(100baseT_Full)	100M full-duplex twisted-pair

**Table 41**    **Constant Overview** (cont'd)

Name	Value	Description
GPY2XX_ADVERTISED_1000baseT_Half	LINK_MODE_MASK(1000baseT_Half)	1G half-duplex twisted-pair
GPY2XX_ADVERTISED_1000baseT_Full	LINK_MODE_MASK(1000baseT_Full)	1G full-duplex twisted-pair
GPY2XX_ADVERTISED_Autoneg	LINK_MODE_MASK(Autoneg)	Auto-negotiation.
GPY2XX_ADVERTISED_TP	LINK_MODE_MASK(TP)	Twisted-pair.
GPY2XX_ADVERTISED_MII	LINK_MODE_MASK(MII)	Media-independent interface.
GPY2XX_ADVERTISED_Pause	LINK_MODE_MASK(Pause)	Pause supported.
GPY2XX_ADVERTISED_Asym_Pause	LINK_MODE_MASK(Asym_Pause)	Asymmetric-pause supported
GPY2XX_ADVERTISED_2500baseT_Full	LINK_MODE_MASK(2500baseT_Full)	2.5G full-duplex twisted-pair
GPY2XX_ADVERTISED_5000baseT_Full	LINK_MODE_MASK(5000baseT_Full)	5G full-duplex twisted-pair
GPY2XX_ADVERTISED_2500baseT_FR	LINK_MODE_MASK(2500baseT_FR)	2.5G Base-T fast retrain
GPY2XX_ADVERTISED_5000baseT_FR	LINK_MODE_MASK(5000baseT_FR)	5G Base-T fast retrain
SPEED_10	10	Macros used by speed in gpy2xx_link.
SPEED_100	100	100 Mbps
SPEED_1000	1000	1 Gbps
SPEED_2500	2500	2.5 Gbps
SPEED_5000	5000	5 Gbps
SPEED_10000	10000	10 Gbps
DUPLEX_HALF	0x00	Macros used by duplex in gpy2xx_link.
DUPLEX_FULL	0x01	Full duplex.
GPIOF_DIR_OUT	(0 << 0)	Macros used by flags in gpy2xx_gpio.
GPIOF_DIR_IN	(1 << 0)	Input pin.
GPIOF_OUTPUT_LOW	(0 << 1)	Output pin low.
GPIOF_OUTPUT_HIGH	(1 << 1)	Output pin high.
GPIOF_INPUT_LOW	(0 << 2)	Input pin low.
GPIOF_INPUT_HIGH	(1 << 2)	Input pin high.

**Table 41**    **Constant Overview** (cont'd)

Name	Value	Description
GPIOF_OPEN_DRAIN	(1 << 3)	GPIO pin is open-drain.
GPIOF_PULL_UP	(2 << 8)	GPIO pin is pull up.
GPIOF_PULL_DOWN	(3 << 8)	GPIO pin pull down.
GPIOF_FUNC	((x)&0x03)<<10)	GPIO pin select alternative function "x" (0~3)
GPIOF_PAD_STR	((x)&0x03)<<12)	GPIO pin pad strength "x": 0 - 2 mA, 1 - 4 mA, 2 - 8 mA, 3 - 12 mA.
GPIOF_SLOW_SLEW	(0 << 15)	GPIO pin slow slew.
GPIOF_FAST_SLEW	(1 << 15)	GPIO pin fast slew.
GPIO_PIN10_GPC0_FUN	10	GPC-0 mux selected on GPIO pin 10 on GPY2XX.
GPIO_PIN07_GPC0_FUN	07	GPC-0 mux selected on GPIO pin 10 on GPY2XX.
WAKE_PHY	(1 << 0)	Macros used by wolopts in gpy2xx_wol_cfg.
WAKE_UCAST	(1 << 1)	Wake up when received Unicast frame.
WAKE_MCAST	(1 << 2)	Wake up when received Multicast fram.
WAKE_BCAST	(1 << 3)	Wake up when received Broadcast frame.
WAKE_ARP	(1 << 4)	Wake up when received ARP frame.
WAKE_MAGIC	(1 << 5)	Wake up when received Magic frame.
WAKE_MAGICSECURE	(1 << 6)	Secured wake upOnly meaningful if WAKE_MAGIC is use.
FW_FWR_DEF_TIMEOUT	5000	Default timeout (in milliseconds) for field firmware upgrade APIs.
SLICE_NUM	8	Max number of slices.
LED_ID_0	0	(Macros used by id in gpy2xx_led_cfg)
LED_ID_1	1	LED ID 1.
LED_ID_2	2	LED ID 2.
LED_ID_3	3	LED ID 3 (Not applicable to gpy24X)
ADS_DOWNSHIFT_THR_MIN	0	Macros used by downshift_thr in gpy2xx_ads_ctrl.
ADS_DOWNSHIFT_THR_MAX	15	ADS_DOWNSHIFT THRESHOLD MAX.
ADS_NRG_RST_CNT_MIN	0	Macros used by nrg_rst_cnt in gpy2xx_ads_ctrl.
ADS_NRG_RST_CNT_MAX	255	ADS_DOWNSHIFT COUNTER MAX.
GSW_PORTMAP_FLAG_SET	(1 << (sizeof(((varType *)0)->nPortId) * 8 - 1))	Sets the portmap flag of a PortID variable. Some Switch API commands allow to use a port index as portmap variable. This requires that the MSB bit is set to indicate that this variable contains a portmap, instead of a port index. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port.

Table 41 Constant Overview (cont'd)

Name	Value	Description
GSW_PORTMAP_FLAG_GET	(1 << (sizeof(((varType *)0)->nPortId) * 8 - 1))	Checks the portmap flag of a PortID variable. Some Switch API commands allow to use a port index as portmap variable. This requires that the MSB bit is set to indicate that this variable contains a portmap, instead of a port index. In portmap mode, every value bit represents an Ethernet port. LSB represents Port 0 with incrementing counting. The (MSB - 1) bit represent the last port.
GSW_MAC_ADDR_LEN	6	MAC Address Field Size. Number of bytes used to store MAC address information.

### 1.10.5 Enumerator Reference

This chapter contains the Enumerator reference.

Table 42 Enumerator Overview

Name	Description
<a href="#">@6</a>	This enumeration type defines two boolean states: False and True.
<a href="#">ads_adv_status</a>	Macros used by no_nrg_rst in gpy2xx_ads_ctrl.
<a href="#">ads_force_rst_status</a>	Macros used by force_rst in gpy2xx_ads_ctrl.
<a href="#">ads_nbt_ds_status</a>	Macros used by downshift_en in gpy2xx_ads_ctrl.
<a href="#">gpy2xx_abist_test</a>	Macros used by test in gpy2xx_abist_param.
<a href="#">gpy2xx_cdiag_state</a>	Macros used by state in gpy2xx_cdiag_sum.
<a href="#">gpy2xx_data_rate</a>	Macros used by data_rate in gpy2xx_synce.
<a href="#">gpy2xx_errcnt_event</a>	Error events to be counted.
<a href="#">gpy2xx_extin_im2_mask</a>	Macros used by ext_imask or ext_istat in gpy2xx_phy_extin.
<a href="#">gpy2xx_extin_phy_event</a>	Macros used by std_imask or std_istat in gpy2xx_phy_extin.
<a href="#">gpy2xx_fwboot_mode</a>	Macros used by fw_memory in gpy2xx_id.
<a href="#">gpy2xx_gpc_sel</a>	Macros used by gpc_sel in gpy2xx_synce and gpy2xx_pps_ctrl.
<a href="#">gpy2xx_led_bsrc</a>	Macros used by slow_blink_src or fast_blink_src or const_on in gpy2xx_led_cfg.
<a href="#">gpy2xx_led_colormode</a>	Macros used by color_mode in gpy2xx_led_cfg.
<a href="#">gpy2xx_led_pulse</a>	Macros used by pulse in gpy2xx_led_cfg.
<a href="#">gpy2xx_sgmii_aneg_mode</a>	Macros used by aneg_mode in gpy2xx_sgmii.
<a href="#">gpy2xx_sgmii_linkcfg_dir</a>	Macros used by linkcfg_dir in gpy2xx_sgmii.
<a href="#">gpy2xx_sgmii_operation</a>	SGMII operation mode.
<a href="#">gpy2xx_synce_clk</a>	Macros used by synce_refclk in gpy2xx_synce.
<a href="#">gpy2xx_synce_master_mode</a>	Macros used by master_sel in gpy2xx_synce.
<a href="#">gpy2xx_test_loop</a>	Test loop modes.
<a href="#">gpy2xx_test_mode</a>	Test modes.
<a href="#">GSW_8021X_portState_t</a>	Describes the 802.1x port state. Used by GSW_8021X_portCfg_t.
<a href="#">GSW_ageTimer_t</a>	Aging Timer Value. Used by GSW_cfg_t.

Table 42 Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_BridgeConfigMask_t</a>	Bridge configuration mask. Used by GSW_BRIDGE_config_t.
<a href="#">GSW_BridgeForwardMode_t</a>	Bridge forwarding type of packet. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_BridgePortConfigMask_t</a>	Bridge Port configuration mask. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_BridgePortEgressMeter_t</a>	Meters for various egress traffic type. Used by GSW_BRIDGE_portConfig_t.
<a href="#">GSW_clkMode_t</a>	Ethernet port clock source configuration. Used by GSW_portLinkCfg_t.
<a href="#">GSW_ColorMarkingMode_t</a>	Color Marking Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_ColorRemarkingMode_t</a>	Color Remarking Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_CPU_ParserHeaderCfg_t</a>	Parser Flags and Offsets Header settings on CPU Port for GSWIP-3.0. Used by GSW_CPU_PortCfg_t.
<a href="#">GSW_CPU_SpecialTagEthType_t</a>	Special tag Ethertype mode.
<a href="#">GSW_CtpPortConfigMask_t</a>	CTP Port configuration mask. Used by GSW_CTP_portConfig_t.
<a href="#">GSW_direction_t</a>	Specifies the direction for ingress and egress. Used by GSW_QoS_meterPort_t and GSW_QoS_meterPortGet_t.
<a href="#">GSW_FCS_TxOps_t</a>	FCS and Pad Insertion operations for GSWIP 3.1 Used by GSW_CPU_PortCfgSet/Get.
<a href="#">GSW_If_RMON_Mode_t</a>	Interface RMON Counter Mode - (FID, SUBID or FLOWID) Config - GSWIP-3.0 only. Used by GSW_portCfg_t.
<a href="#">GSW_IGMP_MemberMode_t</a>	Defines the multicast group member mode. Used by GSW_multicastTable_t and GSW_multicastTableRead_t.
<a href="#">GSW_IP_Select_t</a>	Selection to use IPv4 or IPv6. Used along with GSW_IP_t to denote which union member to be accessed.
<a href="#">GSW_LogicalPortMode_t</a>	Logical port mode. Used by GSW_CTP_portAssignment_t.
<a href="#">GSW_MacClearType_t</a>	MAC Table Clear Type Used by GSW_MAC_tableClearCond_t.
<a href="#">GSW_MacFilterType_t</a>	MAC Address Filter Type. Used by GSW_MACFILTER_default_t.
<a href="#">GSW_MII_Mode_t</a>	Ethernet port interface mode. A port might support only a subset of the possible settings. Used by GSW_portLinkCfg_t.
<a href="#">GSW_MII_Type_t</a>	Ethernet port configuration for PHY or MAC mode. Used by GSW_portLinkCfg_t.
<a href="#">GSW_multicastReportSuppression_t</a>	Configure the IGMP report suppression mode. Used by GSW_multicastSnoopCfg_t.
<a href="#">GSW_multicastSnoopMode_t</a>	Configure the IGMP snooping mode. Used by GSW_multicastSnoopCfg_t.
<a href="#">GSW_PCE_ActionColorFrame_t</a>	Color Frame Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCriticalFrame_t</a>	Critical Frame Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCrossState_t</a>	Cross State Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionCrossVLAN_t</a>	Cross VLAN Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionIGMP_Snoop_t</a>	IGMP Snooping Control. Used by GSW_PCE_action_t.

Table 42 Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_PCE_ActionIrq_t</a>	Interrupt Control Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionLearning_t</a>	MAC Address Learning control. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionMeter_t</a>	Flow Meter Assignment control. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionPortmap_t</a>	Forwarding Group Action Selector. This flow table action and the 'bFlowID_Action' action can be used exclusively. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionTimestamp_t</a>	Timestamp Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionTrafficClass_t</a>	Traffic Class Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ActionVLAN_t</a>	VLAN Group Action Selector. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_IP_t</a>	Rule selection for IPv4/IPv6. Used by GSW_PCE_pattern_t.
<a href="#">GSW_PCE_PortFilterAction_t</a>	Port Filter Action-1/2/3/4/5/6 Selector - used for GSWIP-3.0 only. This can be used only along with PortMember config. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_ProcessingPathAction_t</a>	MPE Processing Path Assignment Selector - used for GSWIP-3.0 only. Used by GSW_PCE_action_t.
<a href="#">GSW_PCE_RuleRegion_t</a>	Traffic Flow Table Management. Used by GSW_PCE_rule_t.
<a href="#">GSW_PCE_SUBIFID_TYPE_t</a>	Select Mode of Sub-Interface ID Field. Used by GSW_PCE_pattern_t.
<a href="#">GSW_PMAC_Ig_Cfg_Src_t</a>	PMAC Ingress Configuration Source Source of the corresponding field.
<a href="#">GSW_PMAC_Proc_Flags_Eg_Cfg_t</a>	Egress PMAC Config Table Selector.
<a href="#">GSW_PMAC_Short_Frame_Chk_t</a>	Short Length Received Frame Check Type for PMAC. Used by PMAC structure GSW_PMAC_Glbl_Cfg_t.
<a href="#">GSW_PmapperMappingMode_t</a>	P-mapper Mapping Mode Used by GSW_CTP_portConfig_t.
<a href="#">GSW_portDuplex_t</a>	Ethernet port duplex status. Used by GSW_portLinkCfg_t.
<a href="#">GSW_portEnable_t</a>	Port Enable Type Selection. Used by GSW_portCfg_t.
<a href="#">GSW_portFlow_t</a>	Ethernet flow control status. Used by GSW_portCfg_t.
<a href="#">GSW_portForward_t</a>	Packet forwarding. Used by GSW_STP_BPDU_Rule_t and GSW_multicastSnoopCfg_t and GSW_8021X_EAPOL_Rule_t.
<a href="#">GSW_portLink_t</a>	Force the MAC and PHY link modus. Used by GSW_portLinkCfg_t.
<a href="#">GSW_portMonitor_t</a>	Port Mirror Options. Used by GSW_portCfg_t.
<a href="#">GSW_portSpeed_t</a>	Ethernet port speed mode. For certain generations of GSWIP, a port might support only a subset of the possible settings. Used by GSW_portLinkCfg_t.
<a href="#">GSW_portType_t</a>	Port Type - GSWIP-3.1 only. Used by GSW_portCfg_t.
<a href="#">GSW_QoS_ClassSelect_t</a>	Selection of the traffic class field. Used by GSW_QoS_portCfg_t. The port default traffic class is assigned in case non of the configured protocol code points given by the packet.
<a href="#">GSW_QoS_DropPrecedence_t</a>	DSCP Drop Precedence to color code assignment. Used by GSW_QoS_DSCP_DropPrecedenceCfg_t.



Table 42 Enumerator Overview (cont'd)

Name	Description
<a href="#">GSW_QoS_ingressRemarking_t</a>	Ingress DSCP remarking attribute. This attribute defines on the ingress port packets how these will be remarked on the egress port. A packet is only remarked in case its ingress and its egress port have remarking enabled. Used by GSW_QoS_portRemarkingCfg_t.
<a href="#">GSW_QoS_Meter_Type</a>	Meter Type - srTCM or trTCM. Defines the Metering algorithm Type. Used by GSW_QoS_meterCfg_t.
<a href="#">GSW_QoS_qMapMode_t</a>	Describes the QoS Queue Mapping Mode. GSWIP-3.1 only. Used by GSW_QoS_queuePort_t.
<a href="#">GSW_QoS_Scheduler_t</a>	Select the type of the Egress Queue Scheduler. Used by GSW_QoS_schedulerCfg_t.
<a href="#">GSW_QoS_WRED_Mode_t</a>	WRED Cfg Type - Automatic (Adaptive) or Manual. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_Profile_t</a>	Drop Probability Profile. Defines the drop probability profile. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_ThreshMode_t</a>	WRED Thresholds Mode Type. - GSWIP-3.0/3.1 only Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_QoS_WRED_WATERMARK_t</a>	Egress Queue Congestion Notification Watermark. Used by GSW_QoS_WRED_Cfg_t.
<a href="#">GSW_return_t</a>	Enumeration for function status return. The upper four bits are reserved for error classification.
<a href="#">GSW_RMON_CountMode_t</a>	RMON Counters Mode Enumeration. This enumeration defines Counters mode - Packets based or Bytes based counting. Metering and Routing Sessions RMON counting support either Byte based or packets based only.
<a href="#">GSW_RMON_type_t</a>	RMON Counters Type enumeration. Used by GSW_RMON_clear_t and GSW_RMON_mode_t.
<a href="#">GSW_RmonMeterColor_t</a>	Used for getting metering RMON counters. Used by GSW_RMON_METER_GET.
<a href="#">GSW_STP_PortState_t</a>	Spanning Tree Protocol port states. Used by GSW_STP_portCfg_t.
<a href="#">GSW_VlanFilterTciMask_t</a>	VLAN Filter TCI Mask. Used by GSW_VLANFILTER_config_t.
<a href="#">link_mode_bit_indices</a>	Link mode bit indices.

### 1.10.5.1 link\_mode\_bit\_indices

#### Description

Link mode bit indices.

#### Prototype

```
enum
{
    LINK_MODE_10baseT_Half_BIT = 0,
    LINK_MODE_10baseT_Full_BIT = 1,
    LINK_MODE_100baseT_Half_BIT = 2,
```

```

LINK_MODE_100baseT_Full_BIT = = 3,
LINK_MODE_1000baseT_Half_BIT = = 4,
LINK_MODE_1000baseT_Full_BIT = = 5,
LINK_MODE_Autoneg_BIT = = 6,
LINK_MODE_TP_BIT = = 7,
LINK_MODE_AUI_BIT = = 8,
LINK_MODE_MII_BIT = = 9,
LINK_MODE_FIBRE_BIT = = 10,
LINK_MODE_BNC_BIT = = 11,
LINK_MODE_10000baseT_Full_BIT = = 12,
LINK_MODE_Pause_BIT = = 13,
LINK_MODE_Asym_Pause_BIT = = 14,
LINK_MODE_2500baseX_Full_BIT = = 15,
LINK_MODE_Backplane_BIT = = 16,
LINK_MODE_1000baseKX_Full_BIT = = 17,
LINK_MODE_10000baseKX4_Full_BIT = = 18,
LINK_MODE_10000baseKR_Full_BIT = = 19,
LINK_MODE_10000baseR_FEC_BIT = = 20,
LINK_MODE_20000baseMLD2_Full_BIT = = 21,
LINK_MODE_20000baseKR2_Full_BIT = = 22,
LINK_MODE_40000baseKR4_Full_BIT = = 23,
LINK_MODE_40000baseCR4_Full_BIT = = 24,
LINK_MODE_40000baseSR4_Full_BIT = = 25,
LINK_MODE_40000baseLR4_Full_BIT = = 26,
LINK_MODE_56000baseKR4_Full_BIT = = 27,
LINK_MODE_56000baseCR4_Full_BIT = = 28,
LINK_MODE_56000baseSR4_Full_BIT = = 29,
LINK_MODE_56000baseLR4_Full_BIT = = 30,
LINK_MODE_25000baseCR_Full_BIT = = 31,
LINK_MODE_25000baseKR_Full_BIT = = 32,
LINK_MODE_25000baseSR_Full_BIT = = 33,
LINK_MODE_50000baseCR2_Full_BIT = = 34,
LINK_MODE_50000baseKR2_Full_BIT = = 35,
LINK_MODE_100000baseKR4_Full_BIT = = 36,
LINK_MODE_100000baseSR4_Full_BIT = = 37,
LINK_MODE_100000baseCR4_Full_BIT = = 38,
LINK_MODE_100000baseLR4_ER4_Full_BIT = = 39,
LINK_MODE_50000baseSR2_Full_BIT = = 40,
LINK_MODE_1000baseX_Full_BIT = = 41,
LINK_MODE_10000baseCR_Full_BIT = = 42,
LINK_MODE_10000baseSR_Full_BIT = = 43,
LINK_MODE_10000baseLR_Full_BIT = = 44,
LINK_MODE_10000baseLRM_Full_BIT = = 45,
LINK_MODE_10000baseER_Full_BIT = = 46,
LINK_MODE_2500baseT_Full_BIT = = 47,
LINK_MODE_5000baseT_Full_BIT = = 48,
LINK_MODE_2500baseT_FR_BIT = = 49,
LINK_MODE_5000baseT_FR_BIT = = 50,
LINK_MODE_LAST = = LINK_MODE_5000baseT_FR_BIT
} link_mode_bit_indices;

```

## Parameters

Name	Value	Description
LINK_MODE_10baseT_Half_BIT	= 0	10M half-duplex twisted-pair
LINK_MODE_10baseT_Full_BIT	= 1	10M full-duplex twisted-pair
LINK_MODE_100baseT_Half_BIT	= 2	100M half-duplex twisted-pair
LINK_MODE_100baseT_Full_BIT	= 3	100M full-duplex twisted-pair
LINK_MODE_1000baseT_Half_BIT	= 4	1G half-duplex twisted-pair
LINK_MODE_1000baseT_Full_BIT	= 5	1G full-duplex twisted-pair
LINK_MODE_Autoneg_BIT	= 6	Auto-negotiation.
LINK_MODE_TP_BIT	= 7	Twisted-pair.
LINK_MODE_AUI_BIT	= 8	Attachment unit interface.
LINK_MODE_MII_BIT	= 9	Media-independent interface.
LINK_MODE_FIBRE_BIT	= 10	Fiber.
LINK_MODE_BNC_BIT	= 11	BNC (Bayonet Neill-Concelman) Connector.
LINK_MODE_10000baseT_Full_BIT	= 12	10G full-duplex twisted-pair
LINK_MODE_Pause_BIT	= 13	Pause supported.
LINK_MODE_Asym_Pause_BIT	= 14	Asymmetric-pause supported.
LINK_MODE_2500baseX_Full_BIT	= 15	2.5G full-duplex
LINK_MODE_Backplane_BIT	= 16	Backplane.
LINK_MODE_1000baseKX_Full_BIT	= 17	1G full-duplex backplane C48 coding
LINK_MODE_10000baseKX4_Full_BIT	= 18	10G full-duplex 4-lane backplane C48 coding
LINK_MODE_10000baseKR_Full_BIT	= 19	10G full-duplex 1-lane backplane C49 coding
LINK_MODE_10000baseR_FEC_BIT	= 20	10G full-duplex C49 coding
LINK_MODE_20000baseMLD2_Full_BIT	= 21	20G full-duplex
LINK_MODE_20000baseKR2_Full_BIT	= 22	20G full-duplex 2-lane backplane C49 coding
LINK_MODE_40000baseKR4_Full_BIT	= 23	40G full-duplex 4-lane backplane C49 coding
LINK_MODE_40000baseCR4_Full_BIT	= 24	40G full-duplex fiber
LINK_MODE_40000baseSR4_Full_BIT	= 25	40G full-duplex fiber
LINK_MODE_40000baseLR4_Full_BIT	= 26	40G full-duplex fiber
LINK_MODE_56000baseKR4_Full_BIT	= 27	56G full-duplex 4-lane backplane C49 coding
LINK_MODE_56000baseCR4_Full_BIT	= 28	56G full-duplex fiber
LINK_MODE_56000baseSR4_Full_BIT	= 29	56G full-duplex fiber
LINK_MODE_56000baseLR4_Full_BIT	= 30	56G full-duplex fiber
LINK_MODE_25000baseCR_Full_BIT	= 31	25G full-duplex fiber
LINK_MODE_25000baseKR_Full_BIT	= 32	25G full-duplex backplane C49 coding
LINK_MODE_25000baseSR_Full_BIT	= 33	25G full-duplex fiber
LINK_MODE_50000baseCR2_Full_BIT	= 34	50G full-duplex fiber
LINK_MODE_50000baseKR2_Full_BIT	= 35	50G full-duplex 2-lane backplane C49 coding
LINK_MODE_100000baseKR4_Full_BIT	= 36	100G full-duplex 4-lane backplane C49 coding

Name	Value	Description
LINK_MODE_100000baseSR4_Full_BIT	= 37	100G full-duplex fiber
LINK_MODE_100000baseCR4_Full_BIT	= 38	100G full-duplex fiber
LINK_MODE_100000baseLR4_ER4_Full_BIT	= 39	100G full-duplex fiber
LINK_MODE_50000baseSR2_Full_BIT	= 40	50G full-duplex fiber
LINK_MODE_1000baseX_Full_BIT	= 41	1G full-duplex
LINK_MODE_10000baseCR_Full_BIT	= 42	10G full-duplex fiber
LINK_MODE_10000baseSR_Full_BIT	= 43	10G full-duplex fiber
LINK_MODE_10000baseLR_Full_BIT	= 44	10G full-duplex fiber
LINK_MODE_10000baseLRM_Full_BIT	= 45	10G full-duplex fiber
LINK_MODE_10000baseER_Full_BIT	= 46	10G full-duplex fiber
LINK_MODE_2500baseT_Full_BIT	= 47	2.5G full-duplex twisted-pair
LINK_MODE_5000baseT_Full_BIT	= 48	5G full-duplex twisted-pair
LINK_MODE_2500baseT_FR_BIT	= 49	2.5G Base-T fast retrain
LINK_MODE_5000baseT_FR_BIT	= 50	5G Base-T fast retrain
LINK_MODE_LAST	= LINK_MODE_5000baseT_FR_BIT	Last Mode.

### 1.10.5.2 gpy2xx\_extin\_phy\_event

#### Description

Macros used by std\_imask or std\_istat in [gpy2xx\\_phy\\_extin](#).

External interrupt event

#### Prototype

```
enum
{
    EXTIN_PHY_LSTC = (1 << 0),
    EXTIN_PHY_LSPC = (1 << 1),
    EXTIN_PHY_DXMC = (1 << 2),
    EXTIN_PHY_MDIXC = (1 << 3),
    EXTIN_PHY_MDIPC = (1 << 4),
    EXTIN_PHY_ADSC = (1 << 5),
    EXTIN_PHY_TEMP = (1 << 6),
    EXTIN_PHY_ULP = (1 << 7),
    EXTIN_PHY_LOR = (1 << 8),
    EXTIN_PHY_ANC = (1 << 10),
    EXTIN_PHY_ANE = (1 << 11),
    EXTIN_PHY_NPTX = (1 << 12),
    EXTIN_PHY_NPRX = (1 << 13),
    EXTIN_PHY_MSRE = (1 << 14),
    EXTIN_PHY_WOL = (1 << 15)
} gpy2xx_extin_phy_event;
```

## Parameters

Name	Value	Description
EXTIN_PHY_LSTC	= (1 << 0)	Link state change.
EXTIN_PHY_LSPC	= (1 << 1)	Link speed change.
EXTIN_PHY_DXMC	= (1 << 2)	Duplex mode change.
EXTIN_PHY_MDIXC	= (1 << 3)	MDI/MDIX crossover change.
EXTIN_PHY_MDIPC	= (1 << 4)	MDI polarity change.
EXTIN_PHY_ADSC	= (1 << 5)	Link's auto-downspeed change.
EXTIN_PHY_TEMP	= (1 << 6)	Link's auto-downspeed change.
EXTIN_PHY_ULP	= (1 << 7)	Link's auto-downspeed change.
EXTIN_PHY_LOR	= (1 << 8)	SyncE loss of reference clock.
EXTIN_PHY_ANC	= (1 << 10)	Auto-negotiation complete.
EXTIN_PHY_ANE	= (1 << 11)	Auto-negotiation error.
EXTIN_PHY_NPTX	= (1 << 12)	Next page transmitted.
EXTIN_PHY_NPRX	= (1 << 13)	Next page received.
EXTIN_PHY_MSRE	= (1 << 14)	Master/slave resolution error.
EXTIN_PHY_WOL	= (1 << 15)	Wake-on-LAN event.

### 1.10.5.3 gpy2xx\_extin\_im2\_mask

#### Description

Macros used by ext\_imask or ext\_istat in [gpy2xx\\_phy\\_extin](#).

IM2 interrupt enable

#### Prototype

```
enum
{
    EXTIN_IM2_IE_LPI = = (1 << 1),
    EXTIN_IM2_IE_TS_FIFO = = (1 << 3),
    EXTIN_IM2_IE_MACSEC = = (1 << 4)
} gpy2xx_extin_im2_mask;
```

#### Parameters

Name	Value	Description
EXTIN_IM2_IE_LPI	= (1 << 1)	Enable interrupt on LPI event hit.
EXTIN_IM2_IE_TS_FIFO	= (1 << 3)	Enable interrupt on any of Rx/Tx Timestamp FIFO is non-zero.
EXTIN_IM2_IE_MACSEC	= (1 << 4)	Enable interrupt on MACsec event hit.

### 1.10.5.4 gpy2xx\_test\_mode

#### Description

Test modes.

#### Prototype

```
enum
{
    TEST_NOP = 0,
    TEST_MODE1 = 1,
    TEST_WAV = TEST_MODE1,
    TEST_MODE2 = 2,
    TEST_JITM = TEST_MODE2,
    TEST_MODE3 = 3,
    TEST_JITS = TEST_MODE3,
    TEST_MODE4 = 4,
    TEST_DIST = TEST_MODE4,
    TEST_AFE = 5,
    TEST_CDIAG = 6,
    TEST_ABIST = 7
} gpy2xx_test_mode;
```

#### Parameters

Name	Value	Description
TEST_NOP	= 0	Normal operation without test.
TEST_MODE1	= 1	Test mode 1 (transmit waveform test) Refer to IEEE 802.3-2015 Table 40-7.
TEST_WAV	= TEST_MODE1	Test mode 1 (transmit waveform test) Refer to IEEE 802.3-2015 Table 40-7.
TEST_MODE2	= 2	Test mode 2 (transmit jitter test in MASTER mode) Refer to IEEE 802.3-2015 Table 40-7.
TEST_JITM	= TEST_MODE2	Test mode 2 (transmit jitter test in MASTER mode) Refer to IEEE 802.3-2015 Table 40-7.
TEST_MODE3	= 3	Test mode 3 (transmit jitter test in SLAVE mode) Refer to IEEE 802.3-2015 Table 40-7.
TEST_JITS	= TEST_MODE3	Test mode 3 (transmit jitter test in SLAVE mode) Refer to IEEE 802.3-2015 Table 40-7.
TEST_MODE4	= 4	Test mode 4 (transmitter distortion test) Refer to IEEE 802.3-2015 Table 40-7.
TEST_DIST	= TEST_MODE4	Test mode 4 (transmitter distortion test) Refer to IEEE 802.3-2015 Table 40-7.
TEST_AFE	= 5	AFE Test.
TEST_CDIAG	= 6	Cable diagnostics.
TEST_ABIST	= 7	Analog built-in self-test.

### 1.10.5.5 gpy2xx\_cdiag\_state

#### Description

Macros used by state in [gpy2xx\\_cdiag\\_sum](#).

Pair state in cable diagnostics

#### Prototype

```
enum
{
    CDIAG_REFLECTION = = 1,
    CDIAG_OPEN = = 2,
    CDIAG_SHORT = = 4,
    CDIAG_MATCHED = = 8
} gpy2xx_cdiag_state;
```

#### Parameters

Name	Value	Description
CDIAG_REFLECTION	= 1	Indicates non-trivial echo due to mismatch at the reported distance (essentially the level is not ignorable, but not as strong as expected from a full reflection)
CDIAG_OPEN	= 2	Indicates a clear level of echo due to an open termination.
CDIAG_SHORT	= 4	Indicates a clear level of echo due to a short termination.
CDIAG_MATCHED	= 8	Indicates no detectable echo impulse (essentially the cable is properly matched)

### 1.10.5.6 gpy2xx\_abist\_test

#### Description

Macros used by test in [gpy2xx\\_abist\\_param](#).

Flags in analog built-in self-test (ABIST)

#### Prototype

```
enum
{
    ABIST_ANALOG_IPV_0 = = 0,
    ABIST_DC_10BT_MAX_PVE = = (1 << 4) | 0,
    ABIST_DC_10BT_0 = = (1 << 4) | 1,
    ABIST_DC_10BT_MAX_NVE = = (1 << 4) | 2,
    ABIST_DC_100BT_MAX_PVE = = (1 << 4) | 3,
    ABIST_DC_100BT_0 = = (1 << 4) | 4,
    ABIST_DC_100BT_MAX_NVE = = (1 << 4) | 5,
    ABIST_DC_1000BT_MAX_PVE = = (1 << 4) | 6,
    ABIST_DC_1000BT_0 = = (1 << 4) | 7,
    ABIST_DC_1000BT_MAX_NVE = = (1 << 4) | 8,
```

```

ABIST_DC_2500BT_MAX_PVE = = (1 << 4) | 9,
ABIST_DC_2500BT_0 = = (1 << 4) | 10,
ABIST_DC_2500BT_MAX_NVE = = (1 << 4) | 11
} gpy2xx_abist_test;

```

## Parameters

Name	Value	Description
ABIST_ANALOG_IPV_0	= 0	Analog test for IP version < 1.5.
ABIST_DC_10BT_MAX_PVE	= (1 << 4)   0	Analog test for IP version >= 1.5. DC test for 10BT mode LD, max +ve differential level
ABIST_DC_10BT_0	= (1 << 4)   1	DC test for 10BT mode LD, 0 differential level.
ABIST_DC_10BT_MAX_NVE	= (1 << 4)   2	DC test for 10BT mode LD, max -ve differential level.
ABIST_DC_100BT_MAX_PVE	= (1 << 4)   3	DC test for 100BT mode LD, max +ve differential level.
ABIST_DC_100BT_0	= (1 << 4)   4	DC test for 100BT mode LD, 0 differential level.
ABIST_DC_100BT_MAX_NVE	= (1 << 4)   5	DC test for 100BT mode LD, max -ve differential level.
ABIST_DC_1000BT_MAX_PVE	= (1 << 4)   6	DC test for 1000BT mode LD, max +ve differential level.
ABIST_DC_1000BT_0	= (1 << 4)   7	DC test for 1000BT mode LD, 0 differential level.
ABIST_DC_1000BT_MAX_NVE	= (1 << 4)   8	DC test for 10000BT mode LD, max -ve differential level.
ABIST_DC_2500BT_MAX_PVE	= (1 << 4)   9	DC test for 2500BT mode LD, max +ve differential level.
ABIST_DC_2500BT_0	= (1 << 4)   10	DC test for 2500BT mode LD, 0 differential level.
ABIST_DC_2500BT_MAX_NVE	= (1 << 4)   11	DC test for 2500BT mode LD, max -ve differential level.

### 1.10.5.7 gpy2xx\_test\_loop

#### Description

Test loop modes.

#### Prototype

```

enum
{
    TLOOP_OFF = = 0,
    TLOOP_NET1 = = 1,
    TLOOP_FET1 = = 2,
    TLOOP_ECHO = = 3,
    TLOOP_RJT1 = = 4,

```



```
TLOOP_FELTS = = 5,
TLOOP_NETLI = = 8
} gpy2xx_test_loop;
```

## Parameters

Name	Value	Description
TLOOP_OFF	= 0	Disable test loop.
TLOOP_NETI	= 1	GMII (Near End) Test Loop: This test loop allows raw (G)MII transmit data to be looped back to the (G)MII receive port. The setting will only take effect after a link down/up event takes place.
TLOOP_FETI	= 2	Far End Test Loop: This PCS far end test loop allows for the receive data at the output of the receive PCS to be fed back into the transmit path, that is, the input of the transmit PCS. The received data is also available at the xMII interface output, however all xMII transmit data is ignored in this test mode. The setting will only take effect after a link down/up event takes place.
TLOOP_ECHO	= 3	DEC (Digital Echo Canceler) Test Loop: This test loop allows the transmit signal to be looped back via the Digital Echo Canceler (DEC). This loopback is similar to the functionality of the MDI test loop (TLOOP_RJTI), except that it does not require special termination circuitry at the MDI connector. The user of this test loop has the option to terminate each twisted pair with a 100 Ohm resistor. This test loop is only applicable for 1000Base-T/2.5GBase-T. The setting will only take effect after a link down/up event takes place.
TLOOP_RJTI	= 4	MDI (RJ45 Near End) Test Loop: This test loop allows for loopback of the signal at the MDI connector, for example RJ45 or SMB. Referring to the four available twisted pairs in a CAT5 or equivalent cable type, pair A is connected to pair B, and pair C to pair D. This shorting of near-end twisted pairs must be enabled using specialized termination circuitry. No additional resistors are required. The setting will only take effect after a link down/up event takes place.

Name	Value	Description
TLOOP_FELTS	= 5	Far End Test Loop: This is the same as TLOOP_FETI, except that TLOOP_FETI is dependent on the availability of TX_CLK and RX_CLK from the MII interface, but the IP takes care of generating the necessary clocks for the loopback to work in this mode. The setting will only take effect after a link down/up event takes place.
TLOOP_NETLI	= 8	GMII (Near End) Test Loop: This test loop allows raw (G)MII transmit data to be looped back to the (G)MII receive port. The difference compared to TLOOP_NETI is that this setting takes effect immediately. The Ethernet port indicates a link down and enters test mode, in addition to closing the (G)MII-to-PCS buffer loop.

### 1.10.5.8 gpy2xx\_errcnt\_event

#### Description

Error events to be counted.

#### Prototype

```
enum
{
    ERRCNT_RXERR = 0,
    ERRCNT_RXACT = 1,
    ERRCNT_ESDERR = 2,
    ERRCNT_SSDERR = 3,
    ERRCNT_TXERR = 4,
    ERRCNT_TXACT = 5,
    ERRCNT_COL = 6,
    ERRCNT_NLD = 8,
    ERRCNT_ADS = 9,
    ERRCNT_CRC = 10,
    ERRCNT_TTL = 11
} gpy2xx_errcnt_event;
```

#### Parameters

Name	Value	Description
ERRCNT_RXERR	= 0	Receive errors are counted.
ERRCNT_RXACT	= 1	Receive frames are counted.
ERRCNT_ESDERR	= 2	ESD errors are counted.
ERRCNT_SSDERR	= 3	SSD errors are counted.
ERRCNT_TXERR	= 4	Transmit errors are counted.

Name	Value	Description
ERRCNT_TXACT	= 5	Transmit frames are counted.
ERRCNT_COL	= 6	Collision events are counted.
ERRCNT_NLD	= 8	Link down events are counted.
ERRCNT_ADS	= 9	Auto-downspeed events are counted.
ERRCNT_CRC	= 10	CRC error events are counted.
ERRCNT_TTL	= 11	Time to link events are counted.

### 1.10.5.9 gpy2xx\_synce\_clk

#### Description

Macros used by `synce_refclk` in [gpy2xx\\_synce](#).

SyncE reference clock input frequency

#### Prototype

```
enum
{
    SYNCE_CLK_PSTN = 0,
    SYNCE_CLK_EEC1 = 1,
    SYNCE_CLK_EEC2 = 2,
    SYNCE_CLK_RES = 3
} gpy2xx_synce_clk;
```

#### Parameters

Name	Value	Description
SYNCE_CLK_PSTN	= 0	SyncE clock frequency is PSTN class: 8KHz.
SYNCE_CLK_EEC1	= 1	SyncE clock frequency is EEC-1 class: 2.048MHz.
SYNCE_CLK_EEC2	= 2	SyncE clock frequency is EEC-2 class: 1.544MHz.
SYNCE_CLK_RES	= 3	Reserved.

### 1.10.5.10 gpy2xx\_synce\_master\_mode

#### Description

Macros used by `master_sel` in [gpy2xx\\_synce](#).

Select synce master, slave mode

#### Prototype

```
enum
{
    SYNCE_SLAVE = 0,
    SYNCE_MASTER = 1
} gpy2xx_synce_master_mode;
```

## Parameters

Name	Value	Description
SYNCE_SLAVE	= 0	SLAVE mode.
SYNCE_MASTER	= 1	Master mode.

### 1.10.5.11 gpy2xx\_data\_rate

#### Description

Macros used by data\_rate in [gpy2xx\\_synce](#).

Data rate

#### Prototype

```
enum
{
    SYNCE_1G = = 0,
    SYNCE_2G5 = = 1
} gpy2xx_data_rate;
```

## Parameters

Name	Value	Description
SYNCE_1G	= 0	SYNCE_1G.
SYNCE_2G5	= 1	SYNCE_2G5.

### 1.10.5.12 gpy2xx\_gpc\_sel

#### Description

Macros used by gpc\_sel in [gpy2xx\\_synce](#) and gpy2xx\_pps\_ctrl.

Time Stamp Capture Input Signal Selection. This is to specify the input signal selected for time stamp capture.

#### Prototype

```
enum
{
    OUTTIMER = = 0,
    SYNCE_GPC0 = = 1,
    SYNCE_GPC1 = = 2,
    SYNCE_GPC2 = = 3
} gpy2xx_gpc_sel;
```

## Parameters

Name	Value	Description
OUTTIMER	= 0	OUT_TIMER signal from PM is selected as input signal to trigger time stamp capture.
SYNCE_GPC0	= 1	GPC0 is selected as input signal to trigger time stamp capture.
SYNCE_GPC1	= 2	GPC1 is selected as input signal to trigger time stamp capture.
SYNCE_GPC2	= 3	GPC2 is selected as input signal to trigger time stamp capture.

### 1.10.5.13 gpy2xx\_sgmii\_linkcfg\_dir

#### Description

Macros used by linkcfg\_dir in gpy2xx\_sgmii.

SGMII link configuration direction

#### Prototype

```
enum
{
    SGMII_LINKCFG_TPI = 0,
    SGMII_LINKCFG_SGMII = 1
} gpy2xx_sgmii_linkcfg_dir;
```

## Parameters

Name	Value	Description
SGMII_LINKCFG_TPI	= 0	SGMII configuration is taken from twisted pair link status.
SGMII_LINKCFG_SGMII	= 1	SGMII configuration is taken from SGMII registers.

### 1.10.5.14 gpy2xx\_sgmii\_aneg\_mode

#### Description

Macros used by aneg\_mode in gpy2xx\_sgmii.

SGMII auto-negotiation mode

#### Prototype

```
enum
{
    SGMII_ANEG_1000BX = 1,
    SGMII_ANEG_CISCO_PHY = 2,
    SGMII_ANEG_CISCO_MAC = 3
} gpy2xx_sgmii_aneg_mode;
```

## Parameters

Name	Value	Description
SGMII_ANEG_1000BX	= 1	1000-Bx ANEG mode
SGMII_ANEG_CISCO_PHY	= 2	SGMII ANEG mode with GPY2xx acting as a PHY.
SGMII_ANEG_CISCO_MAC	= 3	SGMII ANEG mode with GPY2xx acting as a MAC.

### 1.10.5.15 gpy2xx\_sgmii\_operation

#### Description

SGMII operation mode.

#### Prototype

```
enum
{
    SGMII_OP_NORMAL = 0,
    SGMII_OP_DOWN = 1,
    SGMII_OP_LOOPBACK = 2,
    SGMII_OP_RESET = 3
} gpy2xx_sgmii_operation;
```

## Parameters

Name	Value	Description
SGMII_OP_NORMAL	= 0	Normal operation.
SGMII_OP_DOWN	= 1	Power down.
SGMII_OP_LOOPBACK	= 2	Loopback data coming in from analog interface back to itself.
SGMII_OP_RESET	= 3	Reset SGMII block.

### 1.10.5.16 gpy2xx\_led\_colormode

#### Description

Macros used by color\_mode in [gpy2xx\\_led\\_cfg](#).

LED color mode

#### Prototype

```
enum
{
    LED_SINGLE = 0,
    LED_DUAL = 1
} gpy2xx_led_colormode;
```

## Parameters

Name	Value	Description
LED_SINGLE	= 0	Ground mode.
LED_DUAL	= 1	Power mode.

### 1.10.5.17 gpy2xx\_led\_bsrc

#### Description

Macros used by slow\_blink\_src or fast\_blink\_src or const\_on in [gpy2xx\\_led\\_cfg](#).

Stats trigger PHY blinking

#### Prototype

```
enum
{
    LED_BSRC_NONE = 0,
    LED_BSRC_LINK10 = 1,
    LED_BSRC_LINK100 = 2,
    LED_BSRC_LINK1000 = 4,
    LED_BSRC_LINK2500 = 8
} gpy2xx_led_bsrc;
```

## Parameters

Name	Value	Description
LED_BSRC_NONE	= 0	No blinking.
LED_BSRC_LINK10	= 1	Blink on 10 Mbps link.
LED_BSRC_LINK100	= 2	Blink on 100 Mbps link.
LED_BSRC_LINK1000	= 4	Blink on 1000 Mbps link.
LED_BSRC_LINK2500	= 8	Blink on 2500 Mbps link.

### 1.10.5.18 gpy2xx\_led\_pulse

#### Description

Macros used by pulse in [gpy2xx\\_led\\_cfg](#).

LED pulse flags

#### Prototype

```
enum
{
    LED_PULSE_NONE = 0,
    LED_PULSE_TX = 1,
    LED_PULSE_RX = 2,
    LED_PULSE_COL = 4,
    LED_PULSE_NO_CON = 8
} gpy2xx_led_pulse;
```

## Parameters

Name	Value	Description
LED_PULSE_NONE	= 0	No pulsing.
LED_PULSE_TX	= 1	Generate pulse on LED when TX activity is detected.
LED_PULSE_RX	= 2	Generate pulse on LED when RX activity is detected.
LED_PULSE_COL	= 4	Generate pulse on LED when Collision is detected.
LED_PULSE_NO_CON	= 8	Constant ON behavior is switched off.

### 1.10.5.19 ads\_adv\_status

#### Description

Macros used by no\_nrg\_rst in [gpy2xx\\_ads\\_ctrl](#).

Auto-downspeed configuration

#### Prototype

```
enum
{
    ADS_NO_ENERGY_ADV_DIS = = 0,
    ADS_NO_ENERGY_ADV_EN = = 1
} ads_adv_status;
```

## Parameters

Name	Value	Description
ADS_NO_ENERGY_ADV_DIS	= 0	Disable advertising all speeds when no energy is detected.
ADS_NO_ENERGY_ADV_EN	= 1	Enable advertising all speeds when no energy is detected.

### 1.10.5.20 ads\_nbt\_ds\_status

#### Description

Macros used by downshift\_en in [gpy2xx\\_ads\\_ctrl](#).

Auto-downspeed status

#### Prototype

```
enum
{
    ADS_NBT_DOWNSHIFT_DIS = = 0,
    ADS_NBT_DOWNSHIFT_EN = = 1
} ads_nbt_ds_status;
```



## Parameters

Name	Value	Description
ADS_NBT_DOWNSHIFT_DIS	= 0	downshift, disable
ADS_NBT_DOWNSHIFT_EN	= 1	downshift, enable

### 1.10.5.21 ads\_force\_rst\_status

#### Description

Macros used by force\_rst in [gpy2xx\\_ads\\_ctrl](#).

Force reset setting

#### Prototype

```
enum
{
    ADS_FORCE_RST_DIS == 0,
    ADS_FORCE_RST_EN == 1
} ads_force_rst_status;
```

## Parameters

Name	Value	Description
ADS_FORCE_RST_DIS	= 0	Wait for timeout before reset capability advertisement.
ADS_FORCE_RST_EN	= 1	Reset capability advertisement.

### 1.10.5.22 gpy2xx\_fwboot\_mode

#### Description

Macros used by fw\_memory in [gpy2xx\\_id](#).

Indicates the memory target used for firmware execution

#### Prototype

```
enum
{
    FW_EXECUTED_FROM_ROM == 0,
    FW_EXECUTED_FROM_OTP == 1,
    FW_EXECUTED_FROM_FLASH == 2,
    FW_EXECUTED_FROM_SRAM == 3
} gpy2xx_fwboot_mode;
```

## Parameters

Name	Value	Description
FW_EXECUTED_FROM_ROM	= 0	Firmware is executed from ROM.
FW_EXECUTED_FROM_OTP	= 1	Firmware is executed from OTP (OneTimeProgram memory)
FW_EXECUTED_FROM_FLASH	= 2	Firmware is executed from FLASH.
FW_EXECUTED_FROM_SRAM	= 3	Firmware is executed from SRAM (GPY24x only)

### 1.10.5.23 GSW\_STP\_PortState\_t

#### Description

Spanning Tree Protocol port states. Used by [GSW\\_STP\\_portCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_STP_PORT_STATE_FORWARD = 0,
    GSW_STP_PORT_STATE_DISABLE = 1,
    GSW_STP_PORT_STATE_LEARNING = 2,
    GSW_STP_PORT_STATE_BLOCKING = 3
} GSW_STP_PortState_t;
```

#### Parameters

Name	Value	Description
GSW_STP_PORT_STATE_FORWARD	= 0	Forwarding state. The port is allowed to transmit and receive all packets. Address Learning is allowed.
GSW_STP_PORT_STATE_DISABLE	= 1	Disabled/Discarding state. The port entity will not transmit and receive any packets. Learning is disabled in this state.
GSW_STP_PORT_STATE_LEARNING	= 2	Learning state. The port entity will only transmit and receive Spanning Tree Protocol packets (BPDU). All other packets are discarded. MAC table address learning is enabled for all good frames.
GSW_STP_PORT_STATE_BLOCKING	= 3	Blocking/Listening. Only the Spanning Tree Protocol packets will be received and transmitted. All other packets are discarded by the port entity. MAC table address learning is disabled in this state.

### 1.10.5.24 GSW\_8021X\_portState\_t

#### Description

Describes the 802.1x port state. Used by GSW\_8021X\_portCfg\_t.

#### Prototype

```
enum
{
    GSW_8021X_PORT_STATE_AUTHORIZED = 0,
    GSW_8021X_PORT_STATE_UNAUTHORIZED = 1,
    GSW_8021X_PORT_STATE_RX_AUTHORIZED = 2,
    GSW_8021X_PORT_STATE_TX_AUTHORIZED = 3
} GSW_8021X_portState_t;
```

#### Parameters

Name	Value	Description
GSW_8021X_PORT_STATE_AUTHORIZED	= 0	Receive and transmit direction are authorized. The port is allowed to transmit and receive all packets and the address learning process is also allowed.
GSW_8021X_PORT_STATE_UNAUTHORIZED	= 1	Receive and transmit direction are unauthorized. All the packets except EAPOL are not allowed to transmit and receive. The address learning process is disabled.
GSW_8021X_PORT_STATE_RX_AUTHORIZED	= 2	Receive direction is authorized, transmit direction is unauthorized. The port is allowed to receive all packets. Packet transmission to this port is not allowed. The address learning process is also allowed.
GSW_8021X_PORT_STATE_TX_AUTHORIZED	= 3	Transmit direction is authorized, receive direction is unauthorized. The port is allowed to transmit all packets. Packet reception on this port is not allowed. The address learning process is disabled.

### 1.10.5.25 GSW\_ColorRemarkingMode\_t

#### Description

Color Remarking Mode Used by [GSW\\_CTP\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_REMARKING_NONE = 0,
    GSW_REMARKING_DEI = 2,
    GSW_REMARKING_PCP_8P0D = 3,
    GSW_REMARKING_PCP_7P1D = 4,
```

```

        GSW_REMARKING_PCP_6P2D == 5,
        GSW_REMARKING_PCP_5P3D == 6,
        GSW_REMARKING_DSCP_AF == 7
    } GSW_ColorRemarkingMode_t;

```

## Parameters

Name	Value	Description
GSW_REMARKING_NONE	= 0	values from last process stage
GSW_REMARKING_DEI	= 2	DEI mark mode
GSW_REMARKING_PCP_8P0D	= 3	PCP 8P0D mark mode
GSW_REMARKING_PCP_7P1D	= 4	PCP 7P1D mark mode
GSW_REMARKING_PCP_6P2D	= 5	PCP 6P2D mark mode
GSW_REMARKING_PCP_5P3D	= 6	PCP 5P3D mark mode
GSW_REMARKING_DSCP_AF	= 7	DSCP AF class

### 1.10.5.26 GSW\_BridgePortEgressMeter\_t

#### Description

Meters for various egress traffic type. Used by [GSW\\_BRIDGE\\_portConfig\\_t](#).

#### Prototype

```

enum
{
    GSW_BRIDGE_PORT_EGRESS_METER_BROADCAST == 0,
    GSW_BRIDGE_PORT_EGRESS_METER_MULTICAST == 1,
    GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_MC_IP == 2,
    GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_MC_NON_IP == 3,
    GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_UC == 4,
    GSW_BRIDGE_PORT_EGRESS_METER_OTHERS == 5,
    GSW_BRIDGE_PORT_EGRESS_METER_MAX == 6
} GSW_BridgePortEgressMeter_t;

```

## Parameters

Name	Value	Description
GSW_BRIDGE_PORT_EGRESS_METER_BROADCAST	= 0	Index of broadcast traffic meter
GSW_BRIDGE_PORT_EGRESS_METER_MULTICAST	= 1	Index of known multicast traffic meter
GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_MC_IP	= 2	Index of unknown multicast IP traffic meter
GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_MC_NON_IP	= 3	Index of unknown multicast non-IP traffic meter
GSW_BRIDGE_PORT_EGRESS_METER_UNKNOWN_UC	= 4	Index of unknown unicast traffic meter

Name	Value	Description
GSW_BRIDGE_PORT_EGRESS_METER_OTHERS	= 5	Index of traffic meter for other types
GSW_BRIDGE_PORT_EGRESS_METER_MAX	= 6	Number of index

### 1.10.5.27 GSW\_PmapperMappingMode\_t

#### Description

P-mapper Mapping Mode Used by [GSW\\_CTP\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_PMAPPER_MAPPING_PCP == 0,
    GSW_PMAPPER_MAPPING_LAG == 1,
    GSW_PMAPPER_MAPPING_DSCP == 2
} GSW_PmapperMappingMode_t;
```

#### Parameters

Name	Value	Description
GSW_PMAPPER_MAPPING_PCP	= 0	Use PCP for VLAN tagged packets to derive sub interface ID group. \remarks P-mapper table entry 1-8.
GSW_PMAPPER_MAPPING_LAG	= 1	Use LAG Index for Pmapper access (regardless of IP and VLAN packet)to derive sub interface ID group. <i>Note: P-mapper table entry 9-72.</i>
GSW_PMAPPER_MAPPING_DSCP	= 2	Use DSCP for VLAN tagged IP packets to derive sub interface ID group. \remarks P-mapper table entry 9-72.

### 1.10.5.28 GSW\_BridgePortConfigMask\_t

#### Description

Bridge Port configuration mask. Used by [GSW\\_BRIDGE\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_BRIDGE_PORT_CONFIG_MASK_BRIDGE_ID == 0x00000001,
    GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_VLAN == 0x00000002,
    GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN == 0x00000004,
    GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_MARKING == 0x00000008,
    GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_REMARKING == 0x00000010,
    GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_METER == 0x00000020,
```

```

GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_SUB_METER = = 0x00000040,
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_CTP_MAPPING = = 0x00000080,
GSW_BRIDGE_PORT_CONFIG_MASK_BRIDGE_PORT_MAP = = 0x00000100,
GSW_BRIDGE_PORT_CONFIG_MASK_MC_DEST_IP_LOOKUP = = 0x00000200,
GSW_BRIDGE_PORT_CONFIG_MASK_MC_SRC_IP_LOOKUP = = 0x00000400,
GSW_BRIDGE_PORT_CONFIG_MASK_MC_DEST_MAC_LOOKUP = = 0x00000800,
GSW_BRIDGE_PORT_CONFIG_MASK_MC_SRC_MAC_LEARNING = = 0x00001000,
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_SPOOFING = = 0x00002000,
GSW_BRIDGE_PORT_CONFIG_MASK_PORT_LOCK = = 0x00004000,
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_LEARNING_LIMIT = = 0x00008000,
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_LEARNED_COUNT = = 0x00010000,
GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_VLAN_FILTER = = 0x00020000,
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN_FILTER1 = = 0x00040000,
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN_FILTER2 = = 0x00080000,
GSW_BRIDGE_PORT_CONFIG_MASK_VLAN_BASED_MAC_LEARNING = = 0x00100000,
GSW_BRIDGE_PORT_CONFIG_MASK_VLAN_BASED_MULTICAST_LOOKUP = = 0x00200000,
GSW_BRIDGE_PORT_CONFIG_MASK_LOOP_VIOLATION_COUNTER = = 0x00400000,
GSW_BRIDGE_PORT_CONFIG_MASK_ALL = = 0x7FFFFFFF,
GSW_BRIDGE_PORT_CONFIG_MASK_FORCE = = 0x80000000
} GSW_BridgePortConfigMask_t;

```

## Parameters

Name	Value	Description
GSW_BRIDGE_PORT_CONFIG_MASK_BRIDGE_ID	= 0x00000001	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_VLAN	= 0x00000002	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN	= 0x00000004	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_MARKING	= 0x00000008	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_REMARKING	= 0x00000010	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_METER	= 0x00000020	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_SUB_METER	= 0x00000040	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_CTP_MAPPING	= 0x00000080	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_BRIDGE_PORT_MAP	= 0x00000100	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>

## Module Reference

Name	Value	Description
GSW_BRIDGE_PORT_CONFIG_MASK_MC_DEST_IP_LOOKUP	= 0x00000200	Mask for GSW_BRIDGE_portConfig_t::bMcDestIpLookupEnable.
GSW_BRIDGE_PORT_CONFIG_MASK_MC_SRC_IP_LOOKUP	= 0x00000400	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_MC_DEST_MAC_LOOKUP	= 0x00000800	Mask for GSW_BRIDGE_portConfig_t::bDestMacLookupEnable.
GSW_BRIDGE_PORT_CONFIG_MASK_MC_SRC_MAC_LEARNING	= 0x00001000	Mask for GSW_BRIDGE_portConfig_t::bSrcMacLearningEnable.
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_SPOOFING	= 0x00002000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_PORT_LOCK	= 0x00004000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_LEARNING_LIMIT	= 0x00008000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_MAC_LEARNED_COUNT	= 0x00010000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_INGRESS_VLAN_FILTER	= 0x00020000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN_FILTER1	= 0x00040000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_EGRESS_VLAN_FILTER2	= 0x00080000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> and <a href="#">GSW_BRIDGE_portConfig_t</a> .
GSW_BRIDGE_PORT_CONFIG_MASK_VLAN_BASED_MAC_LEARNING	= 0x00100000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_VLAN_BASED_MULTICAST_LOOKUP	= 0x00200000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a> , <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_LOOP_VIOLATION_COUNTER	= 0x00400000	Mask for <a href="#">GSW_BRIDGE_portConfig_t</a>
GSW_BRIDGE_PORT_CONFIG_MASK_ALL	= 0x7FFFFFFF	Enable all
GSW_BRIDGE_PORT_CONFIG_MASK_FORCE	= 0x80000000	Bypass any check for debug purpose

### 1.10.5.29 GSW\_ColorMarkingMode\_t

#### Description

Color Marking Mode Used by [GSW\\_CTP\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_MARKING_ALL_GREEN = 0,
    GSW_MARKING_INTERNAL_MARKING = 1,
    GSW_MARKING_DEI = 2,
    GSW_MARKING_PCP_8P0D = 3,
    GSW_MARKING_PCP_7P1D = 4,
    GSW_MARKING_PCP_6P2D = 5,
    GSW_MARKING_PCP_5P3D = 6,
    GSW_MARKING_DSCP_AF = 7
} GSW_ColorMarkingMode_t;
```

#### Parameters

Name	Value	Description
GSW_MARKING_ALL_GREEN	= 0	mark packets (except critical) to green
GSW_MARKING_INTERNAL_MARKING	= 1	do not change color and priority
GSW_MARKING_DEI	= 2	DEI mark mode
GSW_MARKING_PCP_8P0D	= 3	PCP 8P0D mark mode
GSW_MARKING_PCP_7P1D	= 4	PCP 7P1D mark mode
GSW_MARKING_PCP_6P2D	= 5	PCP 6P2D mark mode
GSW_MARKING_PCP_5P3D	= 6	PCP 5P3D mark mode
GSW_MARKING_DSCP_AF	= 7	DSCP AF class

### 1.10.5.30 GSW\_BridgeConfigMask\_t

#### Description

Bridge configuration mask. Used by [GSW\\_BRIDGE\\_config\\_t](#).

#### Prototype

```
enum
{
    GSW_BRIDGE_CONFIG_MASK_MAC_LEARNING_LIMIT = 0x00000001,
    GSW_BRIDGE_CONFIG_MASK_MAC_LEARNED_COUNT = 0x00000002,
    GSW_BRIDGE_CONFIG_MASK_MAC_DISCARD_COUNT = 0x00000004,
    GSW_BRIDGE_CONFIG_MASK_SUB_METER = 0x00000008,
    GSW_BRIDGE_CONFIG_MASK_FORWARDING_MODE = 0x00000010,
    GSW_BRIDGE_CONFIG_MASK_ALL = 0x7FFFFFFF,
    GSW_BRIDGE_CONFIG_MASK_FORCE = 0x80000000
} GSW_BridgeConfigMask_t;
```



## Parameters

Name	Value	Description
GSW_BRIDGE_CONFIG_MASK_MAC_LEARNING_LIMIT	= 0x00000001	Mask for <a href="#">GSW_BRIDGE_config_t</a> and <a href="#">GSW_BRIDGE_config_t</a> .
GSW_BRIDGE_CONFIG_MASK_MAC_LEARNED_COUNT	= 0x00000002	Mask for <a href="#">GSW_BRIDGE_config_t</a>
GSW_BRIDGE_CONFIG_MASK_MAC_DISCARD_COUNT	= 0x00000004	Mask for <a href="#">GSW_BRIDGE_config_t</a>
GSW_BRIDGE_CONFIG_MASK_SUB_METER	= 0x00000008	Mask for <a href="#">GSW_BRIDGE_config_t</a> and <a href="#">GSW_BRIDGE_config_t</a>
GSW_BRIDGE_CONFIG_MASK_FORWARDING_MODE	= 0x00000010	Mask for <a href="#">GSW_BRIDGE_config_t</a> , <a href="#">GSW_BRIDGE_config_t</a> , <a href="#">GSW_BRIDGE_config_t</a> , and <a href="#">GSW_BRIDGE_config_t</a> .
GSW_BRIDGE_CONFIG_MASK_ALL	= 0x7FFFFFFF	Enable all
GSW_BRIDGE_CONFIG_MASK_FORCE	= 0x80000000	Bypass any check for debug purpose

### 1.10.5.31 GSW\_BridgeForwardMode\_t

#### Description

Bridge forwarding type of packet. Used by [GSW\\_BRIDGE\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_BRIDGE_FORWARD_FLOOD = 0,
    GSW_BRIDGE_FORWARD_DISCARD = 1,
    GSW_BRIDGE_FORWARD_CPU = 2
} GSW_BridgeForwardMode_t;
```

## Parameters

Name	Value	Description
GSW_BRIDGE_FORWARD_FLOOD	= 0	Packet is flooded to port members of ingress bridge port
GSW_BRIDGE_FORWARD_DISCARD	= 1	Packet is discarded
GSW_BRIDGE_FORWARD_CPU	= 2	Packet is forwarded to logical port 0 CTP port 0 bridge port 0

### 1.10.5.32 GSW\_VlanFilterTciMask\_t

#### Description

VLAN Filter TCI Mask. Used by [GSW\\_VLANFILTER\\_config\\_t](#).

### Prototype

```
enum
{
    GSW_VLAN_FILTER_TCI_MASK_VID == 0,
    GSW_VLAN_FILTER_TCI_MASK_PCP == 1,
    GSW_VLAN_FILTER_TCI_MASK_TCI == 2
} GSW_VlanFilterTciMask_t;
```

### Parameters

Name	Value	Description
GSW_VLAN_FILTER_TCI_MASK_VID	= 0	
GSW_VLAN_FILTER_TCI_MASK_PCP	= 1	
GSW_VLAN_FILTER_TCI_MASK_TCI	= 2	

## 1.10.5.33 GSW\_portSpeed\_t

### Description

Ethernet port speed mode. For certain generations of GSWIP, a port might support only a subset of the possible settings. Used by [GSW\\_portLinkCfg\\_t](#).

### Prototype

```
enum
{
    GSW_PORT_SPEED_10,
    GSW_PORT_SPEED_100,
    GSW_PORT_SPEED_200,
    GSW_PORT_SPEED_1000,
    GSW_PORT_SPEED_2500,
    GSW_PORT_SPEED_5000,
    GSW_PORT_SPEED_10000,
    GSW_PORT_SPEED_AUTO
} GSW_portSpeed_t;
```

### Parameters

Name	Value	Description
GSW_PORT_SPEED_10		10 Mbit/s
GSW_PORT_SPEED_100		100 Mbit/s
GSW_PORT_SPEED_200		200 Mbit/s
GSW_PORT_SPEED_1000		1000 Mbit/s
GSW_PORT_SPEED_2500		2.5 Gbit/s
GSW_PORT_SPEED_5000		5 Gbit/s
GSW_PORT_SPEED_10000		10 Gbit/s
GSW_PORT_SPEED_AUTO		Auto speed for XGMAC

### 1.10.5.34 GSW\_portDuplex\_t

#### Description

Ethernet port duplex status. Used by [GSW\\_portLinkCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_DUPLEX_FULL = 0,
    GSW_DUPLEX_HALF = 1,
    GSW_DUPLEX_AUTO = 2
} GSW_portDuplex_t;
```

#### Parameters

Name	Value	Description
GSW_DUPLEX_FULL	= 0	Port operates in full-duplex mode
GSW_DUPLEX_HALF	= 1	Port operates in half-duplex mode
GSW_DUPLEX_AUTO	= 2	Port operates in Auto mode

### 1.10.5.35 GSW\_portLink\_t

#### Description

Force the MAC and PHY link modus. Used by [GSW\\_portLinkCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_PORT_LINK_UP = 0,
    GSW_PORT_LINK_DOWN = 1,
    GSW_PORT_LINK_AUTO = 2
} GSW_portLink_t;
```

#### Parameters

Name	Value	Description
GSW_PORT_LINK_UP	= 0	Link up. Any connected LED still behaves based on the real PHY status.
GSW_PORT_LINK_DOWN	= 1	Link down.
GSW_PORT_LINK_AUTO	= 2	Link Auto.

### 1.10.5.36 GSW\_MII\_Mode\_t

#### Description

Ethernet port interface mode. A port might support only a subset of the possible settings. Used by [GSW\\_portLinkCfg\\_t](#).

### Prototype

```
enum
{
    GSW_PORT_HW_MII = 0,
    GSW_PORT_HW_RMII = 1,
    GSW_PORT_HW_GMII = 2,
    GSW_PORT_HW_RGMII = 3,
    GSW_PORT_HW_XGMII = 4
} GSW_MII_Mode_t;
```

### Parameters

Name	Value	Description
GSW_PORT_HW_MII	= 0	Normal PHY interface (twisted pair), use the internal MII Interface.
GSW_PORT_HW_RMII	= 1	Reduced MII interface in normal mode.
GSW_PORT_HW_GMII	= 2	GMII or MII, depending upon the speed.
GSW_PORT_HW_RGMII	= 3	RGMII mode.
GSW_PORT_HW_XGMII	= 4	XGMII mode.

## 1.10.5.37 GSW\_MII\_Type\_t

### Description

Ethernet port configuration for PHY or MAC mode. Used by [GSW\\_portLinkCfg\\_t](#).

### Prototype

```
enum
{
    GSW_PORT_MAC = 0,
    GSW_PORT_PHY = 1
} GSW_MII_Type_t;
```

### Parameters

Name	Value	Description
GSW_PORT_MAC	= 0	MAC Mode. The Ethernet port is configured to work in MAC mode.
GSW_PORT_PHY	= 1	PHY Mode. The Ethernet port is configured to work in PHY mode.

## 1.10.5.38 GSW\_clkMode\_t

### Description

Ethernet port clock source configuration. Used by [GSW\\_portLinkCfg\\_t](#).

### Prototype

```
enum
{
    GSW_PORT_CLK_NA = 0,
    GSW_PORT_CLK_MASTER = 1,
    GSW_PORT_CLK_SLAVE = 2
} GSW_clkMode_t;
```

### Parameters

Name	Value	Description
GSW_PORT_CLK_NA	= 0	Clock Mode not applicable.
GSW_PORT_CLK_MASTER	= 1	Clock Master Mode. The port is configured to provide the clock as output signal.
GSW_PORT_CLK_SLAVE	= 2	Clock Slave Mode. The port is configured to use the input clock signal.

## 1.10.5.39 GSW\_portEnable\_t

### Description

Port Enable Type Selection. Used by [GSW\\_portCfg\\_t](#).

### Prototype

```
enum
{
    GSW_PORT_DISABLE = 0,
    GSW_PORT_ENABLE_RXTX = 1,
    GSW_PORT_ENABLE_RX = 2,
    GSW_PORT_ENABLE_TX = 3
} GSW_portEnable_t;
```

### Parameters

Name	Value	Description
GSW_PORT_DISABLE	= 0	The port is disabled in both directions.
GSW_PORT_ENABLE_RXTX	= 1	The port is enabled in both directions (ingress and egress).
GSW_PORT_ENABLE_RX	= 2	The port is enabled in the receive (ingress) direction only.
GSW_PORT_ENABLE_TX	= 3	The port is enabled in the transmit (egress) direction only.

## 1.10.5.40 GSW\_portFlow\_t

### Description

Ethernet flow control status. Used by [GSW\\_portCfg\\_t](#).

## Prototype

```
enum
{
    GSW_FLOW_AUTO = 0,
    GSW_FLOW_RX = 1,
    GSW_FLOW_TX = 2,
    GSW_FLOW_RXTX = 3,
    GSW_FLOW_OFF = 4
} GSW_portFlow_t;
```

## Parameters

Name	Value	Description
GSW_FLOW_AUTO	= 0	Automatic flow control mode selection through auto-negotiation.
GSW_FLOW_RX	= 1	Receive flow control only
GSW_FLOW_TX	= 2	Transmit flow control only
GSW_FLOW_RXTX	= 3	Receive and Transmit flow control
GSW_FLOW_OFF	= 4	No flow control

### 1.10.5.41 GSW\_portMonitor\_t

## Description

Port Mirror Options. Used by [GSW\\_portCfg\\_t](#).

## Prototype

```
enum
{
    GSW_PORT_MONITOR_NONE = 0,
    GSW_PORT_MONITOR_RX = 1,
    GSW_PORT_MONITOR_TX = 2,
    GSW_PORT_MONITOR_RXTX = 3,
    GSW_PORT_MONITOR_VLAN_UNKNOWN = 4,
    GSW_PORT_MONITOR_VLAN_MEMBERSHIP = 16,
    GSW_PORT_MONITOR_PORT_STATE = 32,
    GSW_PORT_MONITOR_LEARNING_LIMIT = 64,
    GSW_PORT_MONITOR_PORT_LOCK = 128
} GSW_portMonitor_t;
```

## Parameters

Name	Value	Description
GSW_PORT_MONITOR_NONE	= 0	Mirror Feature is disabled. Normal port usage.
GSW_PORT_MONITOR_RX	= 1	Port Ingress packets are mirrored to the monitor port.
GSW_PORT_MONITOR_TX	= 2	Port Egress packets are mirrored to the monitor port.

Name	Value	Description
GSW_PORT_MONITOR_RXTX	= 3	Port Ingress and Egress packets are mirrored to the monitor port.
GSW_PORT_MONITOR_VLAN_UNKNOWN	= 4	Packet mirroring of 'unknown VLAN violation' frames.
GSW_PORT_MONITOR_VLAN_MEMBERSHIP	= 16	Packet mirroring of 'VLAN ingress or egress membership violation' frames.
GSW_PORT_MONITOR_PORT_STATE	= 32	Packet mirroring of 'port state violation' frames.
GSW_PORT_MONITOR_LEARNING_LIMIT	= 64	Packet mirroring of 'MAC learning limit violation' frames.
GSW_PORT_MONITOR_PORT_LOCK	= 128	Packet mirroring of 'port lock violation' frames.

#### 1.10.5.42 GSW\_If\_RMON\_Mode\_t

##### Description

Interface RMON Counter Mode - (FID, SUBID or FLOWID) Config - GSWIP-3.0 only. Used by [GSW\\_portCfg\\_t](#).

##### Prototype

```
enum
{
    GSW_IF_RMON_FID = 0,
    GSW_IF_RMON_SUBID = 1,
    GSW_IF_RMON_FLOWID_LSB = 2,
    GSW_IF_RMON_FLOWID_MSB = 3
} GSW_If_RMON_Mode_t;
```

##### Parameters

Name	Value	Description
GSW_IF_RMON_FID	= 0	FID based Interface RMON counters Usage
GSW_IF_RMON_SUBID	= 1	Sub-Interface Id based Interface RMON counters Usage
GSW_IF_RMON_FLOWID_LSB	= 2	Flow Id (LSB bits 3 to 0) based Interface RMON counters Usage
GSW_IF_RMON_FLOWID_MSB	= 3	Flow Id (MSB bits 7 to 4) based Interface RMON counters Usage

#### 1.10.5.43 GSW\_return\_t

##### Description

Enumeration for function status return. The upper four bits are reserved for error classification.

##### Prototype

```
enum
{
    GSW_statusOk = 0,
```

```

GSW_statusErr = -1,
GSW_statusParam = -(GSW_ERROR_BASE + 2),
GSW_statusVLAN_Space = -(GSW_ERROR_BASE + 3),
GSW_statusVLAN_ID = -(GSW_ERROR_BASE + 4),
GSW_statusInvalIoctl = -(GSW_ERROR_BASE + 5),
GSW_statusNoSupport = -(GSW_ERROR_BASE + 6),
GSW_statusTimeout = -(GSW_ERROR_BASE + 7),
GSW_statusValueRange = -(GSW_ERROR_BASE + 8),
GSW_statusPortInvalid = -(GSW_ERROR_BASE + 9),
GSW_statusIRQ_Invalid = -(GSW_ERROR_BASE + 10),
GSW_statusMAC_TableFull = -(GSW_ERROR_BASE + 11),
GSW_statusLock_Failed = -(GSW_ERROR_BASE + 12),
GSW_statusEntryNotFound = -(GSW_ERROR_BASE + 13)
} GSW_return_t;

```

### Parameters

Name	Value	Description
GSW_statusOk	= 0	Correct or Expected Status
GSW_statusErr	= -1	Generic or unknown error occurred
GSW_statusParam	= -(GSW_ERROR_BASE + 2)	Invalid function parameter
GSW_statusVLAN_Space	= -(GSW_ERROR_BASE + 3)	No space left in VLAN table
GSW_statusVLAN_ID	= -(GSW_ERROR_BASE + 4)	Requested VLAN ID not found in table
GSW_statusInvalIoctl	= -(GSW_ERROR_BASE + 5)	Invalid ioctl
GSW_statusNoSupport	= -(GSW_ERROR_BASE + 6)	Operation not supported by hardware
GSW_statusTimeout	= -(GSW_ERROR_BASE + 7)	Timeout
GSW_statusValueRange	= -(GSW_ERROR_BASE + 8)	At least one value is out of range
GSW_statusPortInvalid	= -(GSW_ERROR_BASE + 9)	The PortId/QueueId/MeterId/etc. is not available in this hardware or the selected feature is not available on this port
GSW_statusIRQ_Invalid	= -(GSW_ERROR_BASE + 10)	The interrupt is not available in this hardware



Name	Value	Description
GSW_statusMAC_TableFull	= - (GSW_ERROR_BASE + 11)	The MAC table is full, an entry could not be added
GSW_statusLock_Failed	= - (GSW_ERROR_BASE + 12)	Locking failed - SWAPI is busy
GSW_statusEntryNotFound	= - (GSW_ERROR_BASE + 13)	Multicast Forwarding table entry not found

#### 1.10.5.44 GSW\_QoS\_Meter\_Type

##### Description

Meter Type - srTCM or trTCM. Defines the Metering algorithm Type. Used by [GSW\\_QoS\\_meterCfg\\_t](#).

##### Prototype

```
enum
{
    GSW_QoS_Meter_srTCM = 0,
    GSW_QoS_Meter_trTCM = 1
} GSW_QoS_Meter_Type;
```

##### Parameters

Name	Value	Description
GSW_QoS_Meter_srTCM	= 0	srTCM Meter Type
GSW_QoS_Meter_trTCM	= 1	trTCM Meter Type - GSWIP-3.0 only

#### 1.10.5.45 GSW\_direction\_t

##### Description

Specifies the direction for ingress and egress. Used by [GSW\\_QoS\\_meterPort\\_t](#) and [GSW\\_QoS\\_meterPortGet\\_t](#).

##### Prototype

```
enum
{
    GSW_DIRECTION_NONE = 0,
    GSW_DIRECTION_INGRESS = 1,
    GSW_DIRECTION_EGRESS = 2,
    GSW_DIRECTION_BOTH = 3
} GSW_direction_t;
```

## Parameters

Name	Value	Description
GSW_DIRECTION_NONE	= 0	No direction.
GSW_DIRECTION_INGRESS	= 1	Ingress direction.
GSW_DIRECTION_EGRESS	= 2	Egress direction.
GSW_DIRECTION_BOTH	= 3	Ingress and egress direction.

### 1.10.5.46 GSW\_QoS\_DropPrecedence\_t

#### Description

DSCP Drop Precedence to color code assignment. Used by [GSW\\_QoS\\_DSCP\\_DropPrecedenceCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_DROP_PRECEDENCE_CRITICAL = 0,
    GSW_DROP_PRECEDENCE_GREEN = 1,
    GSW_DROP_PRECEDENCE_YELLOW = 2,
    GSW_DROP_PRECEDENCE_RED = 3
} GSW_QoS_DropPrecedence_t;
```

## Parameters

Name	Value	Description
GSW_DROP_PRECEDENCE_CRITICAL	= 0	Critical Packet. Metering never changes the drop precedence of these packets.
GSW_DROP_PRECEDENCE_GREEN	= 1	Green Drop Precedence Packet. Packet is marked with a 'low' drop precedence.
GSW_DROP_PRECEDENCE_YELLOW	= 2	Yellow Drop Precedence Packet. Packet is marked with a 'middle' drop precedence.
GSW_DROP_PRECEDENCE_RED	= 3	Red Drop Precedence Packet. Packet is marked with a 'high' drop precedence.

### 1.10.5.47 GSW\_QoS\_ClassSelect\_t

#### Description

Selection of the traffic class field. Used by [GSW\\_QoS\\_portCfg\\_t](#). The port default traffic class is assigned in case non of the configured protocol code points given by the packet.

#### Prototype

```
enum
{
    GSW_QOS_CLASS_SELECT_NO = 0,
    GSW_QOS_CLASS_SELECT_DSCP = 1,
    GSW_QOS_CLASS_SELECT_PCP = 2,
}
```

```

GSW_QOS_CLASS_SELECT_DSCP_PCP = = 3,
GSW_QOS_CLASS_SELECT_PCP_DSCP = = 4,
GSW_QOS_CLASS_SELECT_SPCP = = 5,
GSW_QOS_CLASS_SELECT_SPCP_DSCP = = 6,
GSW_QOS_CLASS_SELECT_DSCP_SPCP = = 7,
GSW_QOS_CLASS_SELECT_SPCP_PCP = = 8,
GSW_QOS_CLASS_SELECT_SPCP_PCP_DSCP = = 9,
GSW_QOS_CLASS_SELECT_DSCP_SPCP_PCP = = 10
} GSW_QoS_ClassSelect_t;

```

## Parameters

Name	Value	Description
GSW_QOS_CLASS_SELECT_NO	= 0	No traffic class assignment based on DSCP or PCP
GSW_QOS_CLASS_SELECT_DSCP	= 1	Traffic class assignment based on DSCP. PCP information is ignored. The Port Class is used in case DSCP is not available in the packet.
GSW_QOS_CLASS_SELECT_PCP	= 2	Traffic class assignment based on PCP. DSCP information is ignored. The Port Class is used in case PCP is not available in the packet.
GSW_QOS_CLASS_SELECT_DSCP_PCP	= 3	Traffic class assignment based on DSCP. Make the assignment based on PCP in case the DSCP information is not available in the packet header. The Port Class is used in case both are not available in the packet.
GSW_QOS_CLASS_SELECT_PCP_DSCP	= 4	CTAG VLAN PCP, IP DSCP. Traffic class assignment based on CTAG VLAN PCP, alternative use DSCP based assignment.
GSW_QOS_CLASS_SELECT_SPCP	= 5	STAG VLAN PCP. Traffic class assignment based on STAG VLAN PCP.
GSW_QOS_CLASS_SELECT_SPCP_DSCP	= 6	STAG VLAN PCP, IP DSCP. Traffic class assignment based on STAG VLAN PCP, alternative use DSCP based assignment.
GSW_QOS_CLASS_SELECT_DSCP_SPCP	= 7	IP DSCP, STAG VLAN PCP. Traffic class assignment based on DSCP, alternative use STAG VLAN PCP based assignment.
GSW_QOS_CLASS_SELECT_SPCP_PCP	= 8	STAG VLAN PCP, CTAG VLAN PCP. Traffic class assignment based on STAG VLAN PCP, alternative use CTAG VLAN PCP based assignment.

Name	Value	Description
GSW_QOS_CLASS_SELECT_SPCP_PCP_DSCP	= 9	STAG VLAN PCP, CTAG VLAN PCP, IP DSCP. Traffic class assignment based on STAG VLAN PCP, alternative use CTAG VLAN PCP based assignment, alternative use DSCP based assignment.
GSW_QOS_CLASS_SELECT_DSCP_SPCP_PCP	= 10	IP DSCP, STAG VLAN PCP, CTAG VLAN PCP. Traffic class assignment based on DSCP, alternative use STAG VLAN PCP based assignment, alternative use CTAG VLAN PCP based assignment.

### 1.10.5.48 GSW\_Qos\_ingressRemarking\_t

#### Description

Ingress DSCP remarking attribute. This attribute defines on the ingress port packets how these will be remarked on the egress port. A packet is only remarked in case its ingress and its egress port have remarking enabled. Used by [GSW\\_QoS\\_portRemarkingCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_DSCP_REMARK_DISABLE = 0,
    GSW_DSCP_REMARK_TC6 = 1,
    GSW_DSCP_REMARK_TC3 = 2,
    GSW_DSCP_REMARK_DP3 = 3,
    GSW_DSCP_REMARK_DP3_TC3 = 4
} GSW_Qos_ingressRemarking_t;
```

#### Parameters

Name	Value	Description
GSW_DSCP_REMARK_DISABLE	= 0	No DSCP Remarking. No remarking is done on the egress port.
GSW_DSCP_REMARK_TC6	= 1	TC DSCP 6-Bit Remarking. The complete DSCP remarking is done based on the traffic class. The traffic class to DSCP value mapping is given in a device global table.
GSW_DSCP_REMARK_TC3	= 2	TC DSCP 3-Bit Remarking. The upper 3-Bits of the DSCP field are remarked based on the traffic class. The traffic class to DSCP value mapping is given in a device global table.

Name	Value	Description
GSW_DSCP_REMARK_DP3	= 3	Drop Precedence Marking. The Drop Precedence is remarked on the egress side.
GSW_DSCP_REMARK_DP3_TC3	= 4	TC Drop Precedence Marking. The Drop Precedence is remarked on the egress side and the upper 3-Bits of the DSCP field are remarked based on the traffic class. The traffic class to DSCP value mapping is given in a device global table.

#### 1.10.5.49 GSW\_QoS\_qMapMode\_t

##### Description

Describes the QoS Queue Mapping Mode. GSWIP-3.1 only. Used by [GSW\\_QoS\\_queuePort\\_t](#).

##### Prototype

```
enum
{
    GSW_QOS_QMAP_SINGLE_MODE = 0,
    GSW_QOS_QMAP_SUBIFID_MODE = 1
} GSW_QoS_qMapMode_t;
```

##### Parameters

Name	Value	Description
GSW_QOS_QMAP_SINGLE_MODE	= 0	This is default mode where the QID is fixed at GSW_QOS_QUEUE_PORT_SET.
GSW_QOS_QMAP_SUBIFID_MODE	= 1	This is new mode in GSWIP-3.1. The QID given in GSW_QOS_QUEUE_PORT_SET is base, and bit 0~3 of sub-interface ID is offset. The final QID is base + SubIfId[0:3].

#### 1.10.5.50 GSW\_QoS\_Scheduler\_t

##### Description

Select the type of the Egress Queue Scheduler. Used by [GSW\\_QoS\\_schedulerCfg\\_t](#).

##### Prototype

```
enum
{
    GSW_QOS_SCHEDULER_STRICT = 0,
    GSW_QOS_SCHEDULER_WFQ = 1
} GSW_QoS_Scheduler_t;
```

## Parameters

Name	Value	Description
GSW_QOS_SCHEDULER_STRICT	= 0	Strict Priority Scheduler.
GSW_QOS_SCHEDULER_WFQ	= 1	Weighted Fair Queuing Shceduler.

### 1.10.5.51 GSW\_QoS\_WRED\_WATERMARK\_t

#### Description

Egress Queue Congestion Notification Watermark. Used by [GSW\\_QoS\\_WRED\\_Cfg\\_t](#).

#### Prototype

```
enum
{
    GSW_QOS_WRED_WATERMARK_1_4 = 0,
    GSW_QOS_WRED_WATERMARK_1_8 = 1,
    GSW_QOS_WRED_WATERMARK_1_12 = 2,
    GSW_QOS_WRED_WATERMARK_1_16 = 3
} GSW_QoS_WRED_WATERMARK_t;
```

## Parameters

Name	Value	Description
GSW_QOS_WRED_WATERMARK_1_4	= 0	>= 1/4 of green max water mark assert <= 1/4 of green max water mark de assert
GSW_QOS_WRED_WATERMARK_1_8	= 1	>= 1/8 of green max water mark assert <= 1/8 of green max water mark de assert
GSW_QOS_WRED_WATERMARK_1_12	= 2	>= 1/12 of green max water mark assert <= 1/12 of green max water mark de assert
GSW_QOS_WRED_WATERMARK_1_16	= 3	>= 1/16 of green max water mark assert <= 1/16 of green max water mark de assert

### 1.10.5.52 GSW\_QoS\_WRED\_Profile\_t

#### Description

Drop Probability Profile. Defines the drop probability profile. Used by [GSW\\_QoS\\_WRED\\_Cfg\\_t](#).

#### Prototype

```
enum
{
    GSW_QOS_WRED_PROFILE_P0 = 0,
    GSW_QOS_WRED_PROFILE_P1 = 1,
    GSW_QOS_WRED_PROFILE_P2 = 2,
    GSW_QOS_WRED_PROFILE_P3 = 3
} GSW_QoS_WRED_Profile_t;
```

## Parameters

Name	Value	Description
GSW_QOS_WRED_PROFILE_P0	= 0	Pmin = 25%, Pmax = 75% (default)
GSW_QOS_WRED_PROFILE_P1	= 1	Pmin = 25%, Pmax = 50%
GSW_QOS_WRED_PROFILE_P2	= 2	Pmin = 50%, Pmax = 50%
GSW_QOS_WRED_PROFILE_P3	= 3	Pmin = 50%, Pmax = 75%

### 1.10.5.53 GSW\_QoS\_WRED\_Mode\_t

#### Description

WRED Cfg Type - Automatic (Adaptive) or Manual. Used by [GSW\\_QoS\\_WRED\\_Cfg\\_t](#).

#### Prototype

```
enum
{
    GSW_QOS_WRED_Adaptive = 0,
    GSW_QOS_WRED_Manual = 1
} GSW_QoS_WRED_Mode_t;
```

## Parameters

Name	Value	Description
GSW_QOS_WRED_Adaptive	= 0	Automatic - Adaptive Watermark Type - GSWIP-3.0/3.1 only
GSW_QOS_WRED_Manual	= 1	Manual Threshold Levels Type

### 1.10.5.54 GSW\_QoS\_WRED\_ThreshMode\_t

#### Description

WRED Thresholds Mode Type. - GSWIP-3.0/3.1 only Used by [GSW\\_QoS\\_WRED\\_Cfg\\_t](#).

#### Prototype

```
enum
{
    GSW_QOS_WRED_Local_Thresh = 0,
    GSW_QOS_WRED_Global_Thresh = 1,
    GSW_QOS_WRED_Port_Thresh = 2
} GSW_QoS_WRED_ThreshMode_t;
```

## Parameters

Name	Value	Description
GSW_QOS_WRED_Local_Thresh	= 0	Local Thresholds Mode

Name	Value	Description
GSW_QOS_WRED_Global_Thresh	= 1	Global Thresholds Mode
GSW_QOS_WRED_Port_Thresh	= 2	Port queue and Port WRED Thresholds

### 1.10.5.55 GSW\_MacClearType\_t

#### Description

MAC Table Clear Type Used by [GSW\\_MAC\\_tableClearCond\\_t](#).

#### Prototype

```
enum
{
    GSW_MAC_CLEAR_PHY_PORT = 0,
    GSW_MAC_CLEAR_DYNAMIC = 1
} GSW_MacClearType_t;
```

#### Parameters

Name	Value	Description
GSW_MAC_CLEAR_PHY_PORT	= 0	Clear dynamic entries based on Physical Port
GSW_MAC_CLEAR_DYNAMIC	= 1	Clear all dynamic entries

### 1.10.5.56 GSW\_MacFilterType\_t

#### Description

MAC Address Filter Type. Used by [GSW\\_MACFILTER\\_default\\_t](#).

#### Prototype

```
enum
{
    GSW_MACFILTERTYPE_SRC = 0,
    GSW_MACFILTERTYPE_DEST = 1
} GSW_MacFilterType_t;
```

#### Parameters

Name	Value	Description
GSW_MACFILTERTYPE_SRC	= 0	Source MAC Address Filter
GSW_MACFILTERTYPE_DEST	= 1	Destination MAC Address Filter

### 1.10.5.57 GSW\_ageTimer\_t

#### Description

Aging Timer Value. Used by [GSW\\_cfg\\_t](#).



## Prototype

```
enum
{
    GSW_AGETIMER_1_SEC = 1,
    GSW_AGETIMER_10_SEC = 2,
    GSW_AGETIMER_300_SEC = 3,
    GSW_AGETIMER_1_HOUR = 4,
    GSW_AGETIMER_1_DAY = 5,
    GSW_AGETIMER_CUSTOM = 6
} GSW_ageTimer_t;
```

## Parameters

Name	Value	Description
GSW_AGETIMER_1_SEC	= 1	1 second aging time
GSW_AGETIMER_10_SEC	= 2	10 seconds aging time
GSW_AGETIMER_300_SEC	= 3	300 seconds aging time
GSW_AGETIMER_1_HOUR	= 4	1 hour aging time
GSW_AGETIMER_1_DAY	= 5	24 hours aging time
GSW_AGETIMER_CUSTOM	= 6	Custom aging time in seconds

### 1.10.5.58 GSW\_multicastSnoopMode\_t

#### Description

Configure the IGMP snooping mode. Used by [GSW\\_multicastSnoopCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_MULTICAST_SNOOP_MODE_DISABLED = 0,
    GSW_MULTICAST_SNOOP_MODE_AUTOLEARNING = 1,
    GSW_MULTICAST_SNOOP_MODE_FORWARD = 2
} GSW_multicastSnoopMode_t;
```

## Parameters

Name	Value	Description
GSW_MULTICAST_SNOOP_MODE_DISABLED	= 0	IGMP management packet snooping and multicast level 3 table learning is disabled.

Name	Value	Description
GSW_MULTICAST_SNOOP_MODE_AUTOLEARNING	= 1	IGMP management packet snooping is enabled and used for the hardware auto-learning to fill the multicast level 3 table. This is not supported and reserved.
GSW_MULTICAST_SNOOP_MODE_FORWARD	= 2	IGMP management packet snooping is enabled and forwarded to the configured port. No autolearning of the multicast level 3 table. This table has to be maintained by the management software.

### 1.10.5.59 GSW\_portForward\_t

#### Description

Packet forwarding. Used by [GSW\\_STP\\_BPDU\\_Rule\\_t](#) and [GSW\\_multicastSnoopCfg\\_t](#) and [GSW\\_8021X\\_EAPOL\\_Rule\\_t](#).

#### Prototype

```
enum
{
    GSW_PORT_FORWARD_DEFAULT = 0,
    GSW_PORT_FORWARD_DISCARD = 1,
    GSW_PORT_FORWARD_CPU = 2,
    GSW_PORT_FORWARD_PORT = 3
} GSW_portForward_t;
```

#### Parameters

Name	Value	Description
GSW_PORT_FORWARD_DEFAULT	= 0	Default; portmap is determined by the forwarding classification.
GSW_PORT_FORWARD_DISCARD	= 1	Discard; discard packets.
GSW_PORT_FORWARD_CPU	= 2	Forward to the CPU port. This requires that the CPU port is previously set by calling <a href="#">GSW_CPU_PORT_CFG_SET</a> .
GSW_PORT_FORWARD_PORT	= 3	Forward to a port, selected by the parameter 'nForwardPortId'. Please note that this feature is not supported by all hardware platforms.

### 1.10.5.60 GSW\_multicastReportSuppression\_t

#### Description

Configure the IGMP report suppression mode. Used by [GSW\\_multicastSnoopCfg\\_t](#).

#### Prototype

```
enum
{
```

```

        GSW_MULTICAST_REPORT_JOIN == 0,
        GSW_MULTICAST_REPORT == 1,
        GSW_MULTICAST_TRANSPARENT == 2
    } GSW_multicastReportSuppression_t;

```

### Parameters

Name	Value	Description
GSW_MULTICAST_REPORT_JOIN	= 0	Report Suppression and Join Aggregation.
GSW_MULTICAST_REPORT	= 1	Report Suppression. No Join Aggregation.
GSW_MULTICAST_TRANSPARENT	= 2	Transparent Mode. No Report Suppression and no Join Aggregation.

### 1.10.5.61 GSW\_CPU\_SpecialTagEthType\_t

#### Description

Special tag Ethertype mode.

#### Prototype

```

enum
{
    GSW_CPU_ETHTYPE_PREDEFINED == 0,
    GSW_CPU_ETHTYPE_FLOWID == 1
} GSW_CPU_SpecialTagEthType_t;

```

### Parameters

Name	Value	Description
GSW_CPU_ETHTYPE_PREDEFINED	= 0	The EtherType field of the Special Tag of egress packets is always set to a predefined value. This same defined value applies for all switch ports.
GSW_CPU_ETHTYPE_FLOWID	= 1	The Ethertype field of the Special Tag of egress packets is set to the FlowID parameter, which is a results of the switch flow classification result. The switch flow table rule provides this FlowID as action parameter.

### 1.10.5.62 GSW\_CPU\_ParserHeaderCfg\_t

#### Description

Parser Flags and Offsets Header settings on CPU Port for GSWIP-3.0. Used by [GSW\\_CPU\\_PortCfg\\_t](#).

#### Prototype

```

enum
{
    GSW_CPU_PARSER_NIL == 0,

```

```

        GSW_CPU_PARSER_FLAGS = = 1,
        GSW_CPU_PARSER_OFFSETS_FLAGS = = 2,
        GSW_CPU_PARSER_RESERVED = = 3
    } GSW_CPU_ParserHeaderCfg_t;

```

## Parameters

Name	Value	Description
GSW_CPU_PARSER_NIL	= 0	No Parsing Flags or Offsets accompanying to CPU Port for this combination
GSW_CPU_PARSER_FLAGS	= 1	8-Bytes Parsing Flags (Bit 63:0) accompanying to CPU Port for this combination
GSW_CPU_PARSER_OFFSETS_FLAGS	= 2	40-Bytes Offsets (Offset-0 to -39) + 8-Bytes Parsing Flags (Bit 63:0) accompanying to CPU Port for this combination
GSW_CPU_PARSER_RESERVED	= 3	Reserved - for future use

### 1.10.5.63 GSW\_FCS\_TxOps\_t

#### Description

FCS and Pad Insertion operations for GSWIP 3.1 Used by GSW\_CPU\_PortCfgSet/Get.

#### Prototype

```

enum
{
    GSW_CRC_PAD_INS_EN = = 0,
    GSW_CRC_EN_PAD_DIS = = 1,
    GSW_CRC_PAD_INS_DIS = = 2
} GSW_FCS_TxOps_t;

```

## Parameters

Name	Value	Description
GSW_CRC_PAD_INS_EN	= 0	CRC Pad Insertion Enable
GSW_CRC_EN_PAD_DIS	= 1	CRC Insertion Enable Pad Insertion Disable
GSW_CRC_PAD_INS_DIS	= 2	CRC Pad Insertion Disable

### 1.10.5.64 GSW\_IGMP\_MemberMode\_t

#### Description

Defines the multicast group member mode. Used by [GSW\\_multicastTable\\_t](#) and [GSW\\_multicastTableRead\\_t](#).

#### Prototype

```

enum
{
    GSW_IGMP_MEMBER_INCLUDE = = 0,
    GSW_IGMP_MEMBER_EXCLUDE = = 1,

```

```

        GSW_IGMP_MEMBER_DONT_CARE == 2,
        GSW_IGMP_MEMBER_INVALID
    } GSW_IGMP_MemberMode_t;

```

## Parameters

Name	Value	Description
GSW_IGMP_MEMBER_INCLUDE	= 0	Include source IP address membership mode. Only supported for IGMPv3.
GSW_IGMP_MEMBER_EXCLUDE	= 1	Exclude source IP address membership mode. Only supported for IGMPv2.
GSW_IGMP_MEMBER_DONT_CARE	= 2	Group source IP address is 'don't care'. This means all source IP addresses (*) are included for the multicast group membership. This is the default mode for IGMPv1 and IGMPv2.
GSW_IGMP_MEMBER_INVALID		

### 1.10.5.65 GSW\_LogicalPortMode\_t

#### Description

Logical port mode. Used by [GSW\\_CTP\\_portAssignment\\_t](#).

#### Prototype

```

enum
{
    GSW_LOGICAL_PORT_8BIT_WLAN == 0,
    GSW_LOGICAL_PORT_9BIT_WLAN == 1,
    GSW_LOGICAL_PORT_GPON == 2,
    GSW_LOGICAL_PORT_EPON == 3,
    GSW_LOGICAL_PORT_GINT == 4,
    GSW_LOGICAL_PORT_OTHER == 0xFF
} GSW_LogicalPortMode_t;

```

## Parameters

Name	Value	Description
GSW_LOGICAL_PORT_8BIT_WLAN	= 0	WLAN with 8-bit station ID
GSW_LOGICAL_PORT_9BIT_WLAN	= 1	WLAN with 9-bit station ID
GSW_LOGICAL_PORT_GPON	= 2	GPON OMCI context
GSW_LOGICAL_PORT_EPON	= 3	EPON context
GSW_LOGICAL_PORT_GINT	= 4	G.INT context
GSW_LOGICAL_PORT_OTHER	= 0xFF	Others (sub interface ID is 0 by default)

### 1.10.5.66 GSW\_CtpPortConfigMask\_t

#### Description

CTP Port configuration mask. Used by [GSW\\_CTP\\_portConfig\\_t](#).

#### Prototype

```
enum
{
    GSW_CTP_PORT_CONFIG_MASK_BRIDGE_PORT_ID = 0x00000001,
    GSW_CTP_PORT_CONFIG_MASK_FORCE_TRAFFIC_CLASS = 0x00000002,
    GSW_CTP_PORT_CONFIG_MASK_INGRESS_VLAN = 0x00000004,
    GSW_CTP_PORT_CONFIG_MASK_INGRESS_VLAN_IGMP = 0x00000008,
    GSW_CTP_PORT_CONFIG_MASK_EGRESS_VLAN = 0x00000010,
    GSW_CTP_PORT_CONFIG_MASK_EGRESS_VLAN_IGMP = 0x00000020,
    GSW_CTP_PORT_CONFIG_MASK_INGRESS_NTO1_VLAN = 0x00000040,
    GSW_CTP_PORT_CONFIG_MASK_EGRESS_NTO1_VLAN = 0x00000080,
    GSW_CTP_PORT_CONFIG_INGRESS_MARKING = 0x00000100,
    GSW_CTP_PORT_CONFIG_EGRESS_MARKING = 0x00000200,
    GSW_CTP_PORT_CONFIG_EGRESS_MARKING_OVERRIDE = 0x00000400,
    GSW_CTP_PORT_CONFIG_EGRESS_REMARKING = 0x00000800,
    GSW_CTP_PORT_CONFIG_INGRESS_METER = 0x00001000,
    GSW_CTP_PORT_CONFIG_EGRESS_METER = 0x00002000,
    GSW_CTP_PORT_CONFIG_BRIDGING_BYPASS = 0x00004000,
    GSW_CTP_PORT_CONFIG_FLOW_ENTRY = 0x00008000,
    GSW_CTP_PORT_CONFIG_LOOPBACK_AND_MIRROR = 0x00010000,
    GSW_CTP_PORT_CONFIG_MASK_ALL = 0x7FFFFFFF,
    GSW_CTP_PORT_CONFIG_MASK_FORCE = 0x80000000
} GSW_CtpPortConfigMask_t;
```

#### Parameters

Name	Value	Description
GSW_CTP_PORT_CONFIG_MASK_BRIDGE_PORT_ID	= 0x00000001	Mask for <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_FORCE_TRAFFIC_CLASS	= 0x00000002	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_INGRESS_VLAN	= 0x00000004	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_INGRESS_VLAN_IGMP	= 0x00000008	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_EGRESS_VLAN	= 0x00000010	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_EGRESS_VLAN_IGMP	= 0x00000020	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_INGRESS_NTO1_VLAN	= 0x00000040	Mask for <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_EGRESS_NTO1_VLAN	= 0x00000080	Mask for <a href="#">GSW_CTP_portConfig_t</a>

Name	Value	Description
GSW_CTP_PORT_CONFIG_INGRESS_MARKING	= 0x00000100	Mask for <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_EGRESS_MARKING	= 0x00000200	Mask for <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_EGRESS_MARKING_OVERRIDE	= 0x00000400	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_EGRESS_REMARKING	= 0x00000800	Mask for <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_INGRESS_METER	= 0x00001000	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_EGRESS_METER	= 0x00002000	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_BRIDGING_BYPASS	= 0x00004000	Mask for <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_FLOW_ENTRY	= 0x00008000	Mask for <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_LOOPBACK_AND_MIRROR	= 0x00010000	Mask for <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> , <a href="#">GSW_CTP_portConfig_t</a> and <a href="#">GSW_CTP_portConfig_t</a>
GSW_CTP_PORT_CONFIG_MASK_ALL	= 0x7FFFFFFF	Enable all fields
GSW_CTP_PORT_CONFIG_MASK_FORCE	= 0x80000000	Bypass any check for debug purpose

### 1.10.5.67 GSW\_PCE\_ActionTrafficClass\_t

#### Description

Traffic Class Action Selector. Used by .

#### Prototype

```
enum
{
    GSW_PCE_ACTION_TRAFFIC_CLASS_DISABLE = 0,
    GSW_PCE_ACTION_TRAFFIC_CLASS_REGULAR = 1,
    GSW_PCE_ACTION_TRAFFIC_CLASS_ALTERNATIVE = 2
} GSW_PCE_ActionTrafficClass_t;
```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_TRAFFIC_CLASS_DISABLE	= 0	Disabled. Traffic class action is disabled.
GSW_PCE_ACTION_TRAFFIC_CLASS_REGULAR	= 1	Regular Class. Traffic class action is enabled and the CoS classification traffic class is used.
GSW_PCE_ACTION_TRAFFIC_CLASS_ALTERNATIVE	= 2	Alternative Class. Traffic class action is enabled and the class of the 'nTrafficClassAlter' field is used.

### 1.10.5.68 GSW\_PCE\_ActionIGMP\_Snoop\_t

#### Description

IGMP Snooping Control. Used by .

#### Prototype

```
enum
{
    GSW_PCE_ACTION_IGMP_SNOOP_DISABLE = 0,
    GSW_PCE_ACTION_IGMP_SNOOP_REGULAR = 1,
    GSW_PCE_ACTION_IGMP_SNOOP_REPORT = 2,
    GSW_PCE_ACTION_IGMP_SNOOP_LEAVE = 3,
    GSW_PCE_ACTION_IGMP_SNOOP_AD = 4,
    GSW_PCE_ACTION_IGMP_SNOOP_QUERY = 5,
    GSW_PCE_ACTION_IGMP_SNOOP_QUERY_GROUP = 6,
    GSW_PCE_ACTION_IGMP_SNOOP_QUERY_NO_ROUTER = 7
} GSW_PCE_ActionIGMP_Snoop_t;
```

#### Parameters

Name	Value	Description
GSW_PCE_ACTION_IGMP_SNOOP_DISABLE	= 0	Disabled. IGMP Snooping is disabled.
GSW_PCE_ACTION_IGMP_SNOOP_REGULAR	= 1	Default. Regular Packet. No IGMP Snooping action required.
GSW_PCE_ACTION_IGMP_SNOOP_REPORT	= 2	IGMP Report/Join Message.
GSW_PCE_ACTION_IGMP_SNOOP_LEAVE	= 3	IGMP Leave Message.
GSW_PCE_ACTION_IGMP_SNOOP_AD	= 4	Router Solicitation/Advertisement message.
GSW_PCE_ACTION_IGMP_SNOOP_QUERY	= 5	IGMP Query Message.



Name	Value	Description
GSW_PCE_ACTION_IGMP_SNOOP_QUERY_GROUP	= 6	IGMP Group Specific Query Message.
GSW_PCE_ACTION_IGMP_SNOOP_QUERY_NO_ROUTER	= 7	IGMP General Query message without Router Solicitation.

### 1.10.5.69 GSW\_PCE\_ActionLearning\_t

#### Description

MAC Address Learning control. Used by .

#### Prototype

```
enum
{
    GSW_PCE_ACTION_LEARNING_DISABLE = 0,
    GSW_PCE_ACTION_LEARNING_REGULAR = 1,
    GSW_PCE_ACTION_LEARNING_FORCE_NOT = 2,
    GSW_PCE_ACTION_LEARNING_FORCE = 3
} GSW_PCE_ActionLearning_t;
```

#### Parameters

Name	Value	Description
GSW_PCE_ACTION_LEARNING_DISABLE	= 0	Learning is based on the forwarding decision. If the packet is discarded, the address is not learned. If the packet is forwarded to any egress port, the address is learned.
GSW_PCE_ACTION_LEARNING_REGULAR	= 1	Reserved
GSW_PCE_ACTION_LEARNING_FORCE_NOT	= 2	Force No Learning. The address is not learned; forwarding decision ignored.
GSW_PCE_ACTION_LEARNING_FORCE	= 3	Force Learning. The address is learned, the forwarding decision ignored. Note: The MAC Learning Control signals delivered to Port-Map filtering and combined with Final Forwarding Decision. The result is used as a feedback for MAC Address learning in the Bridging Table.

### 1.10.5.70 GSW\_PCE\_ActionIrq\_t

#### Description

Interrupt Control Action Selector. Used by .

#### Prototype

```
enum
{
    GSW_PCE_ACTION_IRQ_DISABLE = 0,
```

```

        GSW_PCE_ACTION_IRQ_REGULAR == 1,
        GSW_PCE_ACTION_IRQ_EVENT == 2
    } GSW_PCE_ActionIrq_t;

```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_IRQ_DISABLE	= 0	Disabled. Interrupt Control Action is disabled for this rule.
GSW_PCE_ACTION_IRQ_REGULAR	= 1	Regular Packet. The Interrupt Control Action is enabled, the packet is treated as a regular packet and no interrupt event is generated.
GSW_PCE_ACTION_IRQ_EVENT	= 2	Interrupt Event. The Interrupt Control Action is enabled and an interrupt event is generated.

### 1.10.5.71 GSW\_PCE\_ActionCrossState\_t

#### Description

Cross State Action Selector. Used by .

#### Prototype

```

enum
{
    GSW_PCE_ACTION_CROSS_STATE_DISABLE == 0,
    GSW_PCE_ACTION_CROSS_STATE_REGULAR == 1,
    GSW_PCE_ACTION_CROSS_STATE_CROSS == 2
} GSW_PCE_ActionCrossState_t;

```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_CROSS_STATE_DISABLE	= 0	Disable. The Cross State Action is disabled.
GSW_PCE_ACTION_CROSS_STATE_REGULAR	= 1	Regular Packet. The Cross State Action is enabled and the packet is treated as a non-Cross-State packet (regular packet). Therefore it does not ignore Port-State filtering rules.
GSW_PCE_ACTION_CROSS_STATE_CROSS	= 2	Cross-State packet. The Cross State Action is enabled and the packet is treated as a Cross-State packet. It ignores the Port-State filtering rules.

### 1.10.5.72 GSW\_PCE\_ActionCriticalFrame\_t

#### Description

Critical Frame Action Selector. Used by .

## Prototype

```
enum
{
    GSW_PCE_ACTION_CRITICAL_FRAME_DISABLE = 0,
    GSW_PCE_ACTION_CRITICAL_FRAME_REGULAR = 1,
    GSW_PCE_ACTION_CRITICAL_FRAME_CRITICAL = 2
} GSW_PCE_ActionCriticalFrame_t;
```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_CRITICAL_FRAME_DISABLE	= 0	Disable. The Critical Frame Action is disabled.
GSW_PCE_ACTION_CRITICAL_FRAME_REGULAR	= 1	Regular Packet. The Critical Frame Action is enabled and the packet is treated as a non-Critical Frame.
GSW_PCE_ACTION_CRITICAL_FRAME_CRITICAL	= 2	Critical Packet. The Critical Frame Action is enabled and the packet is treated as a Critical Frame.

### 1.10.5.73 GSW\_PCE\_ActionColorFrame\_t

## Description

Color Frame Action Selector. Used by .

## Prototype

```
enum
{
    GSW_PCE_ACTION_COLOR_FRAME_DISABLE = 0,
    GSW_PCE_ACTION_COLOR_FRAME_NO_CHANGE = 1,
    GSW_PCE_ACTION_COLOR_FRAME_CRITICAL = 2,
    GSW_PCE_ACTION_COLOR_FRAME_GREEN = 3,
    GSW_PCE_ACTION_COLOR_FRAME_YELLOW = 4,
    GSW_PCE_ACTION_COLOR_FRAME_RED = 5
} GSW_PCE_ActionColorFrame_t;
```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_COLOR_FRAME_DISABLE	= 0	Disable. No color frame action.
GSW_PCE_ACTION_COLOR_FRAME_NO_CHANGE	= 1	Do not change color.
GSW_PCE_ACTION_COLOR_FRAME_CRITICAL	= 2	Identity packet as critical which bypass active congestion management (ACM).
GSW_PCE_ACTION_COLOR_FRAME_GREEN	= 3	Change to green color.

Name	Value	Description
GSW_PCE_ACTION_COLOR_FRAME_YELLOW	= 4	Change to yellow color.
GSW_PCE_ACTION_COLOR_FRAME_RED	= 5	Change to red color.

#### 1.10.5.74 GSW\_PCE\_ActionTimestamp\_t

##### Description

Timestamp Action Selector. Used by .

##### Prototype

```
enum
{
    GSW_PCE_ACTION_TIMESTAMP_DISABLE = 0,
    GSW_PCE_ACTION_TIMESTAMP_REGULAR = 1,
    GSW_PCE_ACTION_TIMESTAMP_STORED = 2
} GSW_PCE_ActionTimestamp_t;
```

##### Parameters

Name	Value	Description
GSW_PCE_ACTION_TIMESTAMP_DISABLE	= 0	Disable. Timestamp Action is disabled for this rule.
GSW_PCE_ACTION_TIMESTAMP_REGULAR	= 1	Regular Packet. The Timestamp Action is enabled for this rule. The packet is treated as a regular packet and no timing information is stored.
GSW_PCE_ACTION_TIMESTAMP_STORED	= 2	Receive/Transmit Timing packet. Ingress and Egress Timestamps for this packet should be stored.

#### 1.10.5.75 GSW\_PCE\_ActionPortmap\_t

##### Description

Forwarding Group Action Selector. This flow table action and the 'bFlowID\_Action' action can be used exclusively. Used by .

##### Prototype

```
enum
{
    GSW_PCE_ACTION_PORTMAP_DISABLE = 0,
    GSW_PCE_ACTION_PORTMAP_REGULAR = 1,
    GSW_PCE_ACTION_PORTMAP_DISCARD = 2,
    GSW_PCE_ACTION_PORTMAP_CPU = 3,
    GSW_PCE_ACTION_PORTMAP_ALTERNATIVE = 4,
    GSW_PCE_ACTION_PORTMAP_MULTICAST_ROUTER = 5,
```

```

        GSW_PCE_ACTION_PORTMAP_MULTICAST_HW_TABLE = = 6,
        GSW_PCE_ACTION_PORTMAP_ALTERNATIVE_VLAN = = 7,
        GSW_PCE_ACTION_PORTMAP_ALTERNATIVE_STAG_VLAN = = 8
    } GSW_PCE_ActionPortmap_t;

```

## Parameters

Name	Value	Description
GSW_PCE_ACTION_PORTMAP_DISABLE	= 0	Disable. Forwarding Group Action is disabled.
GSW_PCE_ACTION_PORTMAP_REGULAR	= 1	Regular Packet. Forwarding Action enabled. Select Default Port-Map (result of Default Forwarding Classification).
GSW_PCE_ACTION_PORTMAP_DISCARD	= 2	Discard. Discard the packets.
GSW_PCE_ACTION_PORTMAP_CPU	= 3	Forward to the CPU port. This requires that the CPU port is previously set by calling GSW_CPU_PORT_CFG_SET.
GSW_PCE_ACTION_PORTMAP_ALTERNATIVE	= 4	Forward to a portmap, selected by the parameter 'nForwardPortMap'. Please note that this feature is not supported by all hardware platforms.
GSW_PCE_ACTION_PORTMAP_MULTICAST_ROUTER	= 5	The packet is treated as Multicast Router Solicitation/Advertisement or Query packet.
GSW_PCE_ACTION_PORTMAP_MULTICAST_HW_TABLE	= 6	The packet is interpreted as Multicast packet and learned in the multicast group table.
GSW_PCE_ACTION_PORTMAP_ALTERNATIVE_VLAN	= 7	The CTAG VLAN portmap classification result is replaced by the portmap parameter 'nForwardPortMap'. All other classification results stay unchanged and will be combined together with the overwritten portmap.
GSW_PCE_ACTION_PORTMAP_ALTERNATIVE_STAG_VLAN	= 8	Add STAG VLAN portmap 'nForwardPortMap' to the overall portmap classification result (AND'ed with the portmap).

### 1.10.5.76 GSW\_PCE\_ActionMeter\_t

#### Description

Flow Meter Assignment control. Used by .

#### Prototype

```

enum
{
    GSW_PCE_ACTION_METER_DISABLE = = 0,
    GSW_PCE_ACTION_METER_REGULAR = = 1,
    GSW_PCE_ACTION_METER_1 = = 2,
    GSW_PCE_ACTION_METER_1_2 = = 3
}

```

```
} GSW_PCE_ActionMeter_t;
```

### Parameters

Name	Value	Description
GSW_PCE_ACTION_METER_DISABLE	= 0	Action Disable.
GSW_PCE_ACTION_METER_REGULAR	= 1	Action Enable. The action is enabled but no dedicated metering instance is assigned by the rule.
GSW_PCE_ACTION_METER_1	= 2	Action Enable. Assign one meter instance as given in parameter "nMeterId".
GSW_PCE_ACTION_METER_1_2	= 3	Action Enable. Assign pair of meter instances. These instances are "nMeterId" and the next following meter instance index.

### 1.10.5.77 GSW\_PCE\_ActionVLAN\_t

#### Description

VLAN Group Action Selector. Used by .

#### Prototype

```
enum
{
    GSW_PCE_ACTION_VLAN_DISABLE = 0,
    GSW_PCE_ACTION_VLAN_REGULAR = 1,
    GSW_PCE_ACTION_VLAN_ALTERNATIVE = 2
} GSW_PCE_ActionVLAN_t;
```

### Parameters

Name	Value	Description
GSW_PCE_ACTION_VLAN_DISABLE	= 0	Disabled. The VLAN Action is disabled.
GSW_PCE_ACTION_VLAN_REGULAR	= 1	Regular VLAN. VLAN Action enabled. Select Default VLAN ID.
GSW_PCE_ACTION_VLAN_ALTERNATIVE	= 2	Alternative VLAN. VLAN Action enabled. Select Alternative VLAN as configured in 'nVLAN_Id' or 'nSVLAN_Id'. For CTAG VLAN it requires that this VLAN ID is configured by calling GSW_VLAN_ID_CREATE in advance. This additional call is not required for STAG VLAN.

### 1.10.5.78 GSW\_PCE\_ActionCrossVLAN\_t

#### Description

Cross VLAN Action Selector. Used by .

### Prototype

```
enum
{
    GSW_PCE_ACTION_CROSS_VLAN_DISABLE = 0,
    GSW_PCE_ACTION_CROSS_VLAN_REGULAR = 1,
    GSW_PCE_ACTION_CROSS_VLAN_CROSS = 2
} GSW_PCE_ActionCrossVLAN_t;
```

### Parameters

Name	Value	Description
GSW_PCE_ACTION_CROSS_VLAN_DISABLE	= 0	Disabled. The Cross VLAN Action is disabled.
GSW_PCE_ACTION_CROSS_VLAN_REGULAR	= 1	Regular VLAN Packet. Do not ignore VLAN filtering rules.
GSW_PCE_ACTION_CROSS_VLAN_CROSS	= 2	Cross-VLAN packet. Ignore VLAN filtering rules.

## 1.10.5.79 GSW\_PCE\_ProcessingPathAction\_t

### Description

MPE Processing Path Assignment Selector - used for GSWIP-3.0 only. Used by .

### Prototype

```
enum
{
    GSW_PCE_PROCESSING_PATH_UNUSED = 0,
    GSW_PCE_PROCESSING_PATH_1 = 1,
    GSW_PCE_PROCESSING_PATH_2 = 2,
    GSW_PCE_PROCESSING_PATH_BOTH = 3
} GSW_PCE_ProcessingPathAction_t;
```

### Parameters

Name	Value	Description
GSW_PCE_PROCESSING_PATH_UNUSED	= 0	Processing Path is not enabled.
GSW_PCE_PROCESSING_PATH_1	= 1	Processing Path-1 is used for MPE-1.
GSW_PCE_PROCESSING_PATH_2	= 2	Processing Path-2 is used for MPE-2.
GSW_PCE_PROCESSING_PATH_BOTH	= 3	Processing Path-1 and -2 are used for MPE-1 & MPE-2.

## 1.10.5.80 GSW\_PCE\_PortFilterAction\_t

### Description

Port Filter Action-1/2/3/4/5/6 Selector - used for GSWIP-3.0 only. This can be used only along with PortMember config. Used by .

## Prototype

```
enum
{
    GSW_PCE_PORT_FILTER_ACTION_UNUSED = 0,
    GSW_PCE_PORT_FILTER_ACTION_1 = 1,
    GSW_PCE_PORT_FILTER_ACTION_2 = 2,
    GSW_PCE_PORT_FILTER_ACTION_3 = 3,
    GSW_PCE_PORT_FILTER_ACTION_4 = 4,
    GSW_PCE_PORT_FILTER_ACTION_5 = 5,
    GSW_PCE_PORT_FILTER_ACTION_6 = 6
} GSW_PCE_PortFilterAction_t;
```

## Parameters

Name	Value	Description
GSW_PCE_PORT_FILTER_ACTION_UNUSED	= 0	Port Filter Action is Unused.
GSW_PCE_PORT_FILTER_ACTION_1	= 1	Port Filter Action Type-1 is used.
GSW_PCE_PORT_FILTER_ACTION_2	= 2	Port Filter Action Type-2 is used.
GSW_PCE_PORT_FILTER_ACTION_3	= 3	Port Filter Action Type-3 is used.
GSW_PCE_PORT_FILTER_ACTION_4	= 4	Port Filter Action Type-4 is used.
GSW_PCE_PORT_FILTER_ACTION_5	= 5	Port Filter Action Type-5 (Unknown Unicast) is used.
GSW_PCE_PORT_FILTER_ACTION_6	= 6	Port Filter Action Type-6 (Unknown Multicast) is used.

### 1.10.5.81 GSW\_PCE\_SUBIFID\_TYPE\_t

#### Description

Select Mode of Sub-Interface ID Field. Used by [GSW\\_PCE\\_pattern\\_t](#).

## Prototype

```
enum
{
    GSW_PCE_SUBIFID_TYPE_GROUP = 0,
    GSW_PCE_SUBIFID_TYPE_BRIDGEPORT = 1
} GSW_PCE_SUBIFID_TYPE_t;
```

## Parameters

Name	Value	Description
GSW_PCE_SUBIFID_TYPE_GROUP	= 0	Sub Interface ID group as defined by GSWIP-3.1.
GSW_PCE_SUBIFID_TYPE_BRIDGEPORT	= 1	Bridge Port ID as defined by GSWIP-3.1.



### 1.10.5.82 GSW\_PCE\_IP\_t

#### Description

Rule selection for IPv4/IPv6. Used by [GSW\\_PCE\\_pattern\\_t](#).

#### Prototype

```
enum
{
    GSW_PCE_IP_DISABLED = 0,
    GSW_PCE_IP_V4 = 1,
    GSW_PCE_IP_V6 = 2
} GSW_PCE_IP_t;
```

#### Parameters

Name	Value	Description
GSW_PCE_IP_DISABLED	= 0	Rule Pattern for IP selection disabled.
GSW_PCE_IP_V4	= 1	Rule Pattern for IPv4.
GSW_PCE_IP_V6	= 2	Rule Pattern for IPv6.

### 1.10.5.83 GSW\_PCE\_RuleRegion\_t

#### Description

Traffic Flow Table Mangement. Used by [GSW\\_PCE\\_rule\\_t](#).

#### Prototype

```
enum
{
    GSW_PCE_RULE_COMMON = 0,
    GSW_PCE_RULE_CTP = 1,
    GSW_PCE_RULE_DEBUG = 2
} GSW_PCE_RuleRegion_t;
```

#### Parameters

Name	Value	Description
GSW_PCE_RULE_COMMON	= 0	PCE Rule Region common for all CTP
GSW_PCE_RULE_CTP	= 1	PCE Rule Region for specific CTP
GSW_PCE_RULE_DEBUG	= 2	PCE Rule Debug (HW direct mapping)

### 1.10.5.84 GSW\_PMAC\_Short\_Frame\_Chk\_t

#### Description

Short Length Received Frame Check Type for PMAC. Used by PMAC structure [GSW\\_PMAC\\_Glbl\\_Cfg\\_t](#).

### Prototype

```
enum
{
    GSW_PMAC_SHORT_LEN_DIS == 0,
    GSW_PMAC_SHORT_LEN_ENA_UNTAG == 1,
    GSW_PMAC_SHORT_LEN_ENA_TAG == 2,
    GSW_PMAC_SHORT_LEN_RESERVED == 3
} GSW_PMAC_Short_Frame_Chk_t;
```

### Parameters

Name	Value	Description
GSW_PMAC_SHORT_LEN_DIS	= 0	Short frame length check is disabled.
GSW_PMAC_SHORT_LEN_ENA_UNTAG	= 1	Short frame length check is enabled without considering VLAN Tags.
GSW_PMAC_SHORT_LEN_ENA_TAG	= 2	Short frame length check is enabled including VLAN Tags.
GSW_PMAC_SHORT_LEN_RESERVED	= 3	Reserved - Currently unused

## 1.10.5.85 GSW\_PMAC\_Proc\_Flags\_Eg\_Cfg\_t

### Description

Egress PMAC Config Table Selector.

### Prototype

```
enum
{
    GSW_PMAC_PROC_FLAGS_NONE == 0,
    GSW_PMAC_PROC_FLAGS_TC == 1,
    GSW_PMAC_PROC_FLAGS_FLAG == 2,
    GSW_PMAC_PROC_FLAGS_MIX == 3
} GSW_PMAC_Proc_Flags_Eg_Cfg_t;
```

### Parameters

Name	Value	Description
GSW_PMAC_PROC_FLAGS_NONE	= 0	Use value of <a href="#">GSW_PMAC_Glbl_Cfg_t</a>
GSW_PMAC_PROC_FLAGS_TC	= 1	Use traffic class for egress config table addressing
GSW_PMAC_PROC_FLAGS_FLAG	= 2	Use flags (MPE1, MPE2, DEC, ENC) for egress config table addressing
GSW_PMAC_PROC_FLAGS_MIX	= 3	Use reduced traffic class (saturated to 3) and flags (MPE1, MPE2) for egress config table addressing

### 1.10.5.86 GSW\_PMAC\_Ig\_Cfg\_Src\_t

#### Description

PMAC Ingress Configuration Source Source of the corresponding field.

#### Prototype

```
enum
{
    GSW_PMAC_IG_CFG_SRC_DMA_DESC = 0,
    GSW_PMAC_IG_CFG_SRC_DEF_PMAC = 1,
    GSW_PMAC_IG_CFG_SRC_PMAC = 2
} GSW_PMAC_Ig_Cfg_Src_t;
```

#### Parameters

Name	Value	Description
GSW_PMAC_IG_CFG_SRC_DMA_DESC	= 0	Field is from DMA descriptor
GSW_PMAC_IG_CFG_SRC_DEF_PMAC	= 1	Field is from default PMAC header
GSW_PMAC_IG_CFG_SRC_PMAC	= 2	Field is from PMAC header of packet

### 1.10.5.87 GSW\_portType\_t

#### Description

Port Type - GSWIP-3.1 only. Used by [GSW\\_portCfg\\_t](#).

#### Prototype

```
enum
{
    GSW_LOGICAL_PORT = 0,
    GSW_PHYSICAL_PORT = 1,
    GSW_CTP_PORT = 2,
    GSW_BRIDGE_PORT = 3
} GSW_portType_t;
```

#### Parameters

Name	Value	Description
GSW_LOGICAL_PORT	= 0	Logical Port
GSW_PHYSICAL_PORT	= 1	Physical Port Applicable only for GSWIP-3.1/3.2
GSW_CTP_PORT	= 2	Connectivity Termination Port (CTP) Applicable only for GSWIP-3.1/3.2
GSW_BRIDGE_PORT	= 3	Bridge Port Applicable only for GSWIP-3.1/3.2

### 1.10.5.88 GSW\_RMON\_type\_t

#### Description

RMON Counters Type enumeration. Used by [GSW\\_RMON\\_clear\\_t](#) and [GSW\\_RMON\\_mode\\_t](#).

#### Prototype

```
enum
{
    GSW_RMON_ALL_TYPE = 0,
    GSW_RMON_PMAC_TYPE = 1,
    GSW_RMON_PORT_TYPE = 2,
    GSW_RMON_METER_TYPE = 3,
    GSW_RMON_IF_TYPE = 4,
    GSW_RMON_ROUTE_TYPE = 5,
    GSW_RMON_REDIRECT_TYPE = 6,
    GSW_RMON_BRIDGE_TYPE = 7,
    GSW_RMON_CTP_TYPE = 8
} GSW_RMON_type_t;
```

#### Parameters

Name	Value	Description
GSW_RMON_ALL_TYPE	= 0	All RMON Types Counters
GSW_RMON_PMAC_TYPE	= 1	All PMAC RMON Counters
GSW_RMON_PORT_TYPE	= 2	Port based RMON Counters
GSW_RMON_METER_TYPE	= 3	Meter based RMON Counters
GSW_RMON_IF_TYPE	= 4	Interface based RMON Counters
GSW_RMON_ROUTE_TYPE	= 5	Route based RMON Counters
GSW_RMON_REDIRECT_TYPE	= 6	Redirected Traffic based RMON Counters
GSW_RMON_BRIDGE_TYPE	= 7	Bridge Port based RMON Counters
GSW_RMON_CTP_TYPE	= 8	CTP Port based RMON Counters

### 1.10.5.89 GSW\_RMON\_CountMode\_t

#### Description

RMON Counters Mode Enumeration. This enumeration defines Counters mode - Packets based or Bytes based counting. Metering and Routing Sessions RMON counting support either Byte based or packets based only.

#### Prototype

```
enum
{
    GSW_RMON_COUNT_PKTS = 0,
    GSW_RMON_COUNT_BYTES = 1,
    GSW_RMON_DROP_COUNT = 2
} GSW_RMON_CountMode_t;
```

## Parameters

Name	Value	Description
GSW_RMON_COUNT_PKTS	= 0	Packet based RMON Counters
GSW_RMON_COUNT_BYTES	= 1	Bytes based RMON Counters
GSW_RMON_DROP_COUNT	= 2	number of dropped frames, supported only for interface counters

### 1.10.5.90 GSW\_RmonMeterColor\_t

#### Description

Used for getting metering RMON counters. Used by GSW\_RMON\_METER\_GET.

#### Prototype

```
enum
{
    GSW_RMON_METER_COLOR_RES = 0,
    GSW_RMON_METER_COLOR_GREEN = 1,
    GSW_RMON_METER_COLOR_YELLOW = 2,
    GSW_RMON_METER_COLOR_RED = 3
} GSW_RmonMeterColor_t;
```

## Parameters

Name	Value	Description
GSW_RMON_METER_COLOR_RES	= 0	
GSW_RMON_METER_COLOR_GREEN	= 1	
GSW_RMON_METER_COLOR_YELLOW	= 2	
GSW_RMON_METER_COLOR_RED	= 3	

### 1.10.5.91 @6

#### Description

This enumeration type defines two boolean states: False and True.

#### Prototype

```
enum
{
    GSW_FALSE = 0,
    GSW_TRUE = 1
} @6;
```

## Parameters

Name	Value	Description
GSW_FALSE	= 0	Boolean False.
GSW_TRUE	= 1	Boolean True.

### 1.10.5.92 GSW\_IP\_Select\_t

#### Description

Selection to use IPv4 or IPv6. Used along with [GSW\\_IP\\_t](#) to denote which union member to be accessed.

#### Prototype

```
enum
{
    GSW_IP_SELECT_IPV4 = 0,
    GSW_IP_SELECT_IPV6 = 1
} GSW_IP_Select_t;
```

## Parameters

Name	Value	Description
GSW_IP_SELECT_IPV4	= 0	IPv4 Type
GSW_IP_SELECT_IPV6	= 1	IPv6 Type