MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

© Marvell Solutions

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Gob * Senao 12156925

SOHO Switch Software Architecture Specification

Doc. No. MV-S800039-00, Rev. --
July 25, 2003

86mazc0oog79my
MARVELL CONFIDE

MOVING FORWARD FASTER®

MARVELL®

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Document Status	
Advanced Information	This document contains design specifications for initial product development. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Preliminary Information	This document contains preliminary data, and a revision of this document will be published at a later date. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Final Information	This document contains specifications on a product that is in final release. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Revision Code: Rev. --	
Advanced	Technical Publication: 0.32

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925 MARVELL CONFIDENTIAL, UNDER

This page to be used for Preliminary or Advanced information product documentation.

Disclaimer
This document provides Preliminary information about the products described, and such information should not be used for purpose of final design. Visit the Marvell® web site at www.marvell.com for the latest information on Marvell products.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document. Marvell makes no commitment either to update or to keep current the information contained in this document. Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications. The user should contact Marvell to obtain the latest specifications before finalizing a product design. Marvell assumes no responsibility, either for use of these products or for any infringements of patents and trademarks, or other rights of third parties resulting from its use. No license is granted under any patents, patent rights, or trademarks of Marvell.These products may include one or more optional functions. The user has the choice of implementing any particular optional function. Should the user choose to implement any of these optional functions, it is possible that the use could be subject to third party intellectual property rights. Marvell recommends that the user investigate whether third party intellectual property rights are relevant to the intended use of these products and obtain licenses as appropriate under relevant intellectual property rights.
Marvell comprises Marvell Technology Group Ltd. (MTGL) and its subsidiaries, Marvell International Ltd. (MIL), Marvell Semiconductor, Inc. (MSI), Marvell Asia Pte Ltd. (MAPL), Marvell Japan K.K. (MJKK), Marvell Semiconductor Israel Ltd. (MSIL), SysKonnect GmbH, and Radlan Computer Communications, Ltd.
Export Controls. With respect to any of Marvell's Information, the user or recipient, in the absence of appropriate U.S. government authorization, agrees: 1) not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2; 2) not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and, 3) in the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML"). At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.
Copyright 2000. Marvell. All rights reserved. Marvell, the Marvell logo, Moving Forward Faster, Alaska, and GalNet are registered trademarks of Marvell. Discovery, Fastwriter, GalTis, Horizon, Libertas, Link Street, NetGX, PHY Advantage, Prestera, Raising The Technology Bar, UniMAC, Virtual Cable Tester, and Yukon are trademarks of Marvell. All other trademarks are the property of their respective owners.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

Table of Contents

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

Section 1. Introduction

This document describes the driver suite of the Marvell® SOHO Switch Product Family. This driver suite supports the complete Marvell SOHO Switch Product Family. Throughout the remainder of this document, the driver suite will be referred to as the "QuarterDeck Driver Suite". The QuarterDeck Driver Suite definitions, driver architecture, driver implementation, and Application Programming Interface (API) definitions are provided. The source code (written in C), QDDriver2.0.zip, is coupled with this document and is provided separately.

Hardware

The SOHO Switch Products are either single chip integrations of complete Fast Ethernet switches with various valuable and cost effective features, or Gigabit Ethernet switches with SERDES, 1000BASE-FX, and/or (G)MII interfaces. Each product in the SOHO Switch Product Family has different features, so please refer to the particular datasheet for each switch device for more information.

SOHO Switch Product Family Hardware Interface to Software

The many operating modes of the SOHO Switch Products can be configured via the IEEE standard Serial Management Interface (SMI). The SMI is otherwise known as the MDC/MDIO. Using this interface, the CPU has full access to the PHY and switch registers to control all of the configurations of the PHYs and the switch.

SOHO Switch Product Family Software

The QuarterDeck Driver Suite is developed to control the SOHO Switch Products hardware through the SMI interface. The driver is designed with a layered architecture, and can be ported to the customers' target platform. The driver suite is composed of a set of comprehensive drivers for managing the SOHO Switch Product systems.

The QuarterDeck Driver Suite serves as a foundation for the customer-developed applications, such as IEEE 802.1 bridging services, IEEE 802.1p based QoS, IPv4 DiffServ, and IPv6 TC based QoS. The API can be utilized to offer customer developed higher layer applications such as IP routing, and NAT.

Based on a modular architecture and comprehensive APIs, the QuarterDeck Driver Suite enables software developers to integrate high-level applications without detailed knowledge of the SOHO Switch Products registers and tables.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

1.1 References

- 88E6051 Datasheet - Integrated 6-Port QoS 10/100 Ethernet Switch
- 88E6052 Datasheet - Integrated 7-Port QoS 10/100 Ethernet Switch
- 88E6021 Datasheet - Integrated 3-Port QoS 10/100 Ethernet Switch
- 88E6063 Datasheet - Integrated 7-Port QoS 10/100 Ethernet Switch
- 88E6083 Datasheet - Integrated 10-Port QoS 10/100 Ethernet Switch
- 88E6181/88E6183 Datasheet - 10 Port 10/100/1000 Ethernet Switch

- 88E3081 Datasheet - Integrated 8-Port 10/100 Fast Ethernet Transceiver
- 88E3082 Datasheet - Integrated 8-Port 10/100 Fast Ethernet Transceiver
- 88E1145 Datasheet - Integrated 4-Port 10/100/1000 Ethernet Transceiver

1.2 Acronyms and Abbreviations

Acronym	Definition
MSAPI	Marvell Semiconductor Application Programming Interface
FDB	Filtering Database
SMI	Serial Management Interface
STP	Spanning Tree Protocol
RTOS	Real Time Operating System

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Section 2. Software Architecture

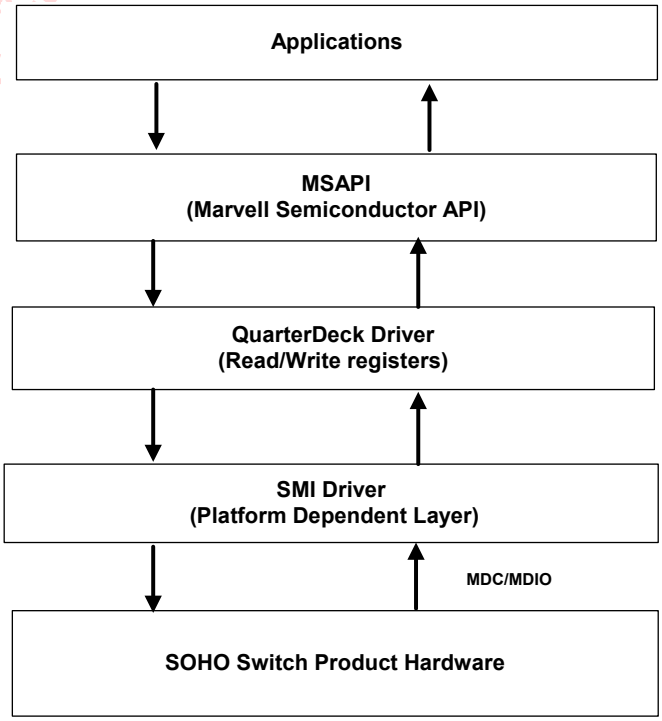
2.1 Overview

The QuarterDeck Driver Suite is designed to support multiple SOHO Switch Products on a single system and consists of the following main components:

- Marvell Semiconductor API Layer
- QuarterDeck Driver Layer
- SMI Driver Layer, which is platform dependent and needs to be provided by BSP.

The QuarterDeck Driver Suite interfaces with the SOHO Switch Products via SMI Driver layer by using the SMI (MDC/MDIO) interface. Applications interface with the driver suite through well-defined Marvell Semiconductor APIs. The applications do not need to be aware of the layers beneath the API.

Figure 1: SOHO Switch Product Software Architecture



MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

2.2 Architecture Goals

The SOHO Switch Product software architecture is designed to be layered, modular, portable, and scalable. It is compatible among the Marvell SOHO switch family members.

2.2.1 Multi-Layer Architecture

The QuarterDeck Driver Suite is based on a multi-layer architecture. This architecture improves maintainability and debug capabilities. It also makes the driver more portable.

2.2.2 Portability

The SOHO Switch Product software is CPU independent. It makes no assumptions about the host CPU on register size and Endianess.

The SOHO Switch Product software is also both OS independent and target hardware platform independent. Its sources are implemented in ANSI C and use none of the other library facilities. Thus, the user can switch RTOS with no modifications of the software.

The SOHO Switch Product software includes well-defined APIs. Marvell will maintain and support this API set for future generations of the device. This will enable customers to leverage application development across multiple generations of the SOHO switch products.

2.2.3 Modularity and Scalability

The Marvell Semiconductor Application Programming Interface is divided into functionally-oriented modules such as STP, VLAN, FDB, and QoS. Modularity allows the user to incorporate only those modules that are required. In other words, adding, enhancing, or removing a module from the API does not affect other modules, e.g., enhancing the VLAN modules for a future device does not affect the QoS module functions, and vice versa.

2.2.4 Compatibility

The SOHO Switch Product software is compatible within the products in the SOHO switch family. A driver may work with newer products, but it will not support the added features available in the newer products. However, a new driver will always work with an older product within the SOHO switch family.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

2.3 Driver Porting

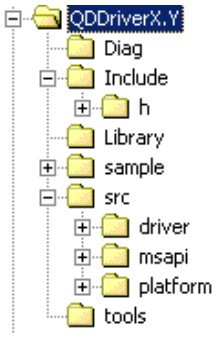
2.3.1 SOHO Switch Product Source Code Organization

The QuarterDeck Driver Suite is distributed in source code format, in C. The source code can be found in the QDDriverX.Y.zip file, where [X.Y] represents the driver's version number.

When the contents of the zip file QDDriverX.Y.zip are extracted, a root folder called QDDriverX.Y is installed on the user's current directory. This folder contains a tree structure of subfolders that correspond to the architecture of the QuarterDeck Driver Suite. The subfolders are: Include, Library, Sample, src, and tools.

- The Include subfolder includes header files used by QuarterDeck driver source files.
- The Library subfolder includes object files.
- The sample subfolder includes various samples that show how to utilize MSAPIs.
- The src subfolder includes QuarterDeck Driver Sources.
- The tools subfolder includes make utilities.
- The Diag subfolder includes a diagnostic program to test both APIs and Switch Device.

Figure 2: Source Code Organization



Users need to copy the files in these folders to their own development environment while maintaining this tree structure. To build the QuarterDeck driver, refer to the README in the QDDriverX.Y directory. If a flat directory is preferred for all the files, some changes need to be made in setenv.bat, makefile, its definition files, and rule files in the tools directory so that the user can build the QuarterDeck driver library. The build environment for the current version supports vxWorks and WinCE for MIPS. If a different RTOS or a different type of CPU is used, the user may reference the setup for vxWorks and WinCE, or the user may incorporate the driver source files into their own build environment.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

2.3.2 SOHO Switch Product Driver Initialization

QuarterDeck Driver should be initialized before accessing any of the MS APIs. It will be initialized by calling the qdLoadDriver() which will perform the following:

1. Registering Platform specific routines.
2. Configuring the SOHO Switch Device.

2.3.2.1 Registering Platform Routines

A user must register platform specific functions to the SOHO Switch Product driver. These functions include SMI interface specific functions and Semaphore specific functions, and can be registered by calling qdLoadDriver().

The following structure will be used to register platform specific functions.

```
typedef struct_BSP_FUNCTIONS
{
        FGT_READ_MII        readMii      /* read MII Registers */
        FGT_WRITE_MII       writeMii     /* write MII Registers */
        FGT_SEM_CREATE      semCreate    /* create semaphore, OPTIONAL */
        FGT_SEM_DELETE      semDelete    /* delete the semaphore, OPTIONAL */
        FGT_SEM_TAKE        semTake      /* try to get a semaphore, OPTIONAL */
        FGT_SEM_GIVE        semGive      /* return semaphore, OPTIONAL */
}BSP_FUNCTIONS;
```

The readMii and writeMii functions are required before using any of the MS APIs; however, semaphore related functions are OPTIONAL. In the QuarterDeck Driver Suite, semaphores are used for accesses of the MAC Address Table (gtBrgFdb.c), 802.1Q VLAN Table (gtBrgVtu.c), and RMON Counter Table (gtPortRmon.c). Therefore, if only one task is accessing any one of the above tables at a given time, none of the optional semaphore functions are required.

Refer to qdLoadDriver() function definition and Section 4 for more information.

2.3.2.2 Configuring the SOHO Switch Product

Function qdLoadDriver() can be used to configure SOHO Switch Product. Required input values for the configuration are CPU port number, initial state of Ports, and Device Scan Mode.

Refer to qdLoadDriver() function definition for more information.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

Section 3. SOHO Switch Product Hardware

The SOHO Switch Product registers are accessible using the IEEE Serial Management Interface (SMI). In this section, the 88E6063 and 88E6181/88E6183 devices will be used as a reference. For other SOHO Switch Products, please refer to the appropriate Datasheet, since registers may be slightly different from 88E6063 and 88E6181/88E6183 devices.

3.1 Register Map for Fast Ethernet Switch Product

The SOHO Switch Products use 16 of the 32 possible device addresses. The 16 device addresses are configurable at reset by use of the EE_CLK/ADDR4 pin (see the appropriate SOHO Switch Product Datasheet).

Figure 3 shows the 88E6063 register map assuming the lower 16 SMI device addresses are used. When the upper 16 SMI device addresses are used, the register map ranges from 0x10 to 0x1F, instead of 0x0 to 0xF as shown.

Note

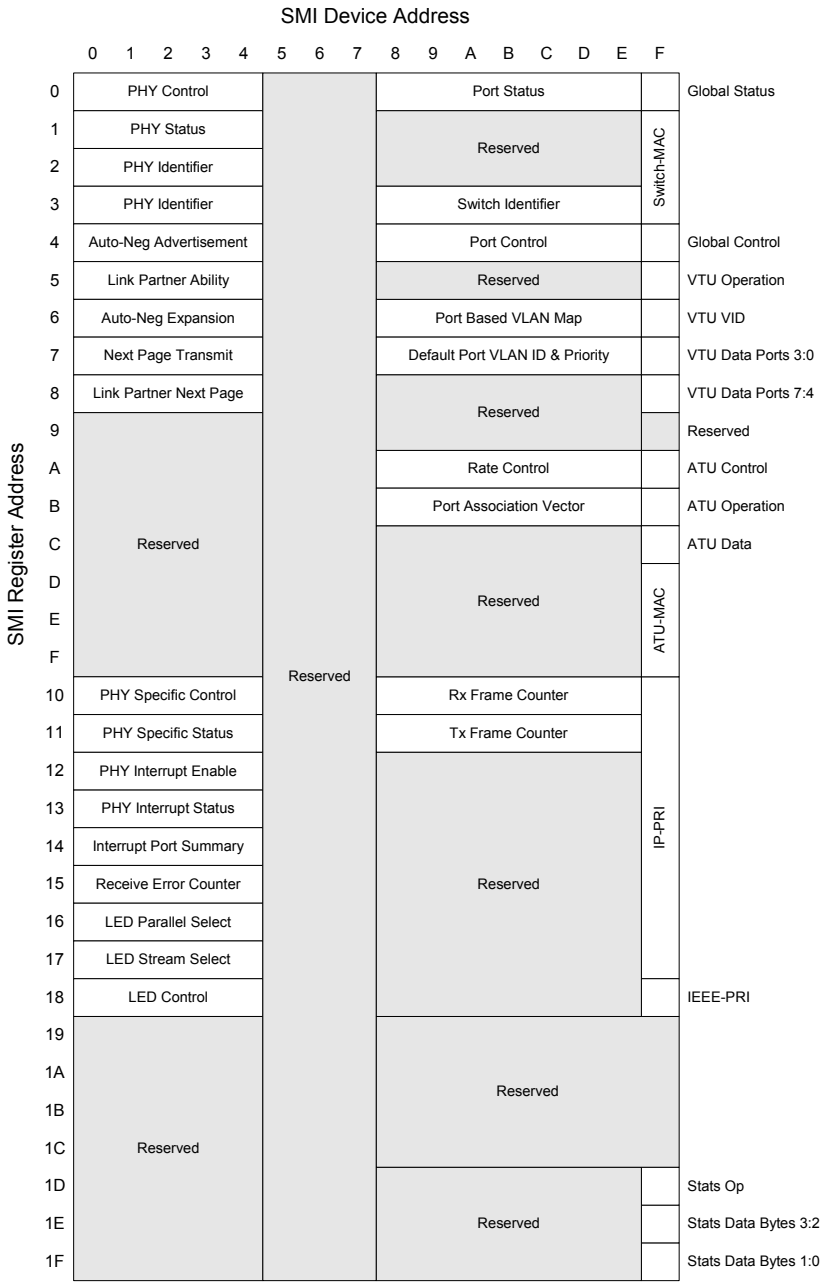
88E6083 (88E6080) contains 10 ports which means that 16 device addresses are not enough for Register access. It uses total of 19 devices addresses: 8 Phy Addresses, 10 Port Addresses, and 1 Global Address. Please refer to 88E6083 Datasheet for the detailed information.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Figure 3: 88E6063 Register Map



MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

The SOHO Switch Product registers fall into three categories:

- Switch port registers
- PHY registers
- Global registers

3.1.1 Switch Port Register

The 88E6063 contains seven switch ports (MACs). These switch ports are accessible using SMI device addresses 0x8 to 0xE or 0x18 to 0x1E, depending upon the value of the EE_CLK/ADDR4 pin at reset. The MACs are fully IEEE 802.3 compliant.

Note

The 88E6083 contains 10 switch ports and their addresses are fixed from 0x10 to 0x19.

3.1.2 PHY Registers

The 88E6063 contains five physical layer devices (PHYs). These devices are accessible using SMI device addresses 0x0 to 0x4 or 0x10 to 0x14, depending upon the value of the EE_CLKADDR4 pin at reset. The PHYs are fully IEEE 802.3 compliant including their register interface.

The PHYs in the 88E6063 are identical to the Marvell® 88E3082 Octal Transceiver except there are only five transceivers (transceivers 5 to 7 do not exist and are not accessible).

Note

The 88E6051/88E6052 and 88E6021 Fast Ethernet switch PHYs are identical to the Marvell 88E3081 Octal Transceivers.

Note

The 88E6083 contains 8 Phys and their addresses are fixed from 0x0 to 0x7.

3.1.3 Switch Global Register

The switch contains many global registers that are used to control features and functions that are common to all ports in the switch. The global registers are accessible using SMI device address 0xF or 0x1F, depending upon the value of the EE_CLK/ADDR4 pin at reset.

Note

Global Register address for 88E6083 device is fixed at 0x1B. Please refer to 88E6083 Datasheet.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

3.2 Register Map for Gigabit Ethernet Switch Product

The Gigabit Ethernet Switch Products supports two different kinds of SMI address usage models. One uses 1 of 32 possible device addresses (Multi Chip Mode). The other uses all of the 32 possible device addresses (Single Chip Mode). The device addresses used and mode is configurable at Reset with the ADDR[4:0] configuration pins (see the appropriate SOHO Gigabit Switch Product Datasheet).

When Single chip addressing mode is used, the Gigabit Switch Product responds to all 32 SMI device addresses so it must be the only device connected to a SMI Master.

When Multi chip addressing mode is used, the Gigabit Switch Product responds to only 1 of 32 possible SMI device addresses so it can share the SMI interface with multiple devices. In this mode only two registers are directly accessible, the SMI Command register (offset 0) and SMI Data register (offset 1). These two registers are used to indirectly access all the Switch registers (along with any PHY registers that may be attached to it.)

Figure 4 shows the 88E6183 device register map, which can be accessed through either Single chip addressing mode or Multi chip addressing mode.

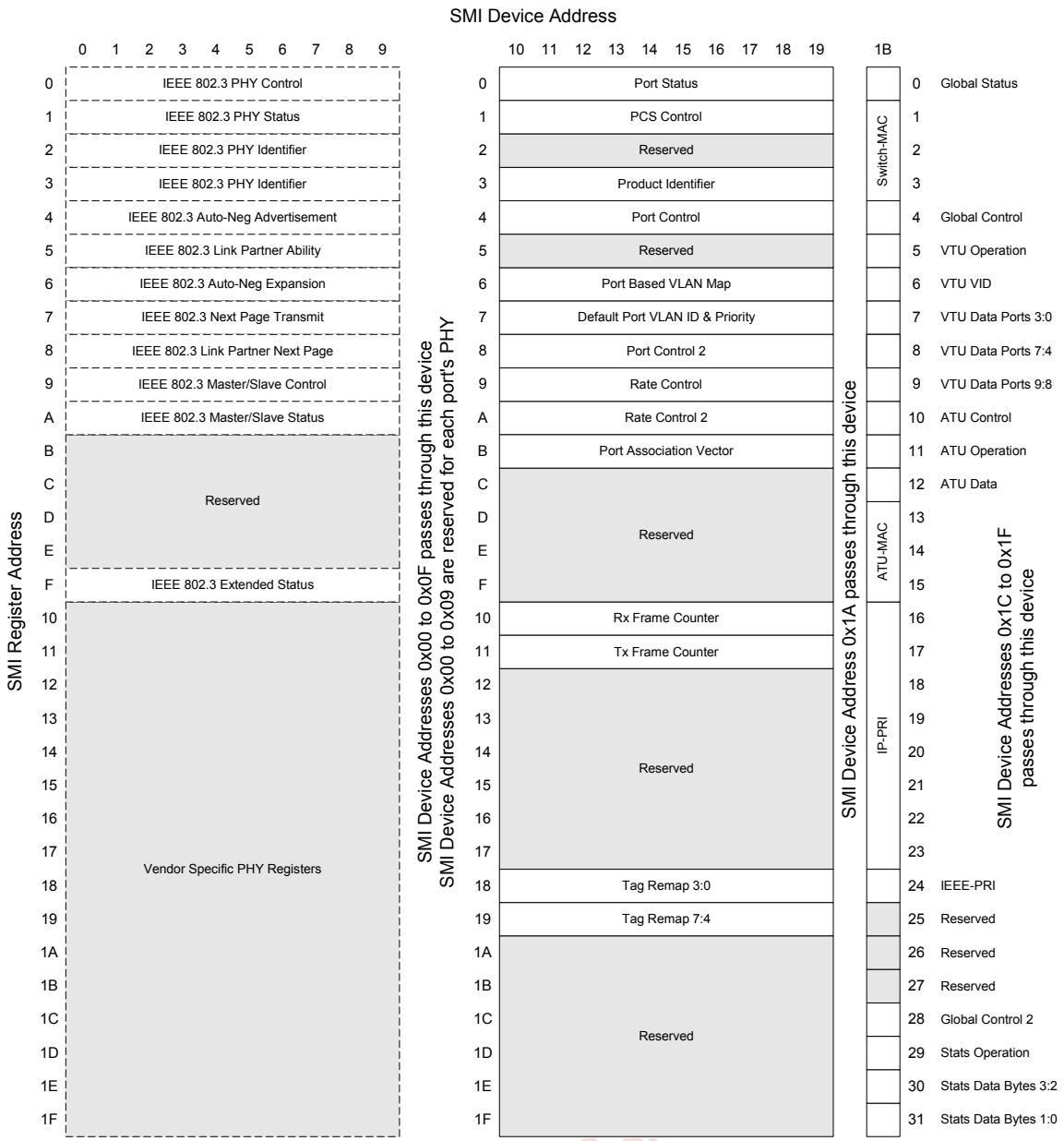
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

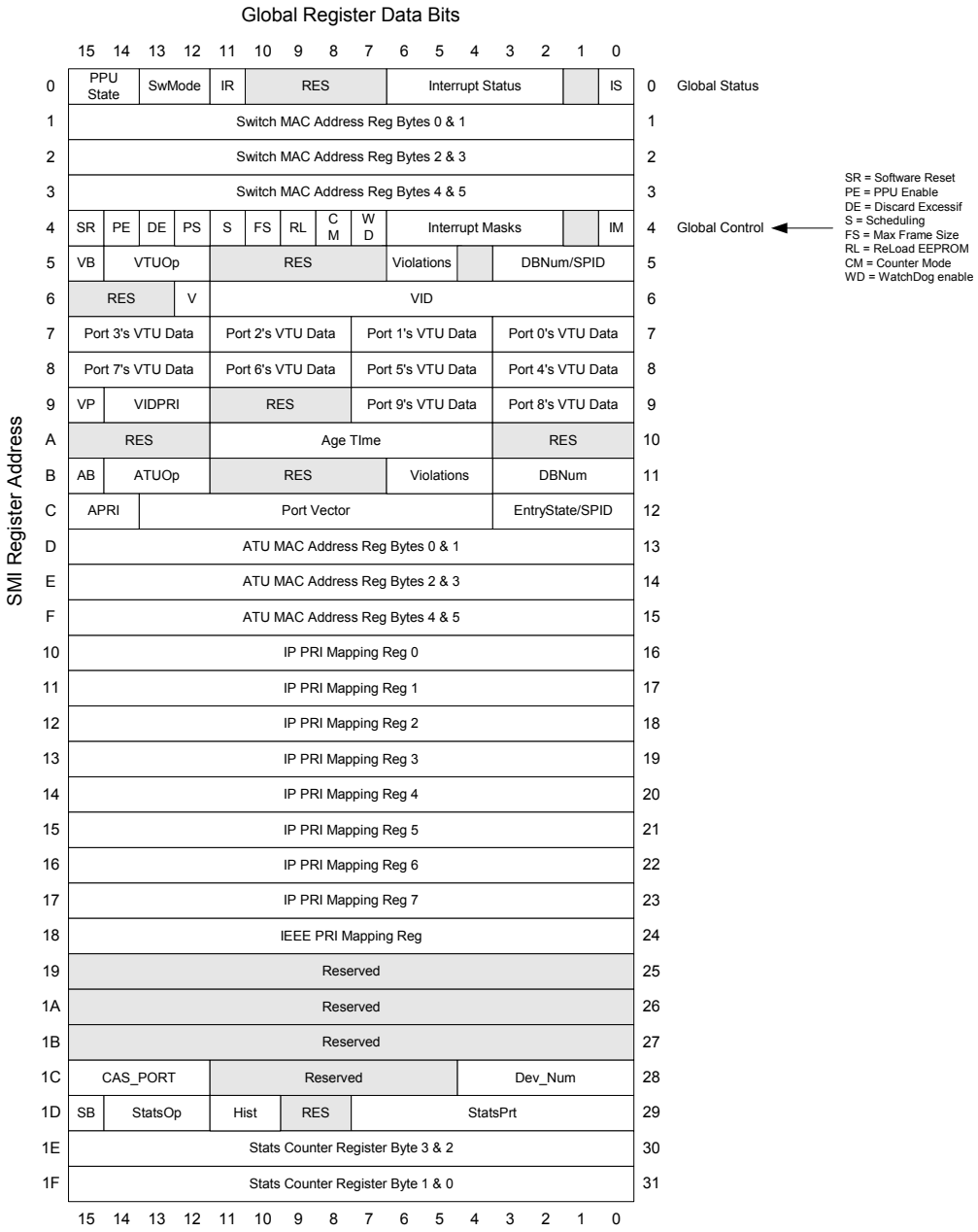
Figure 4: 88E6183 Device Register Map



MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

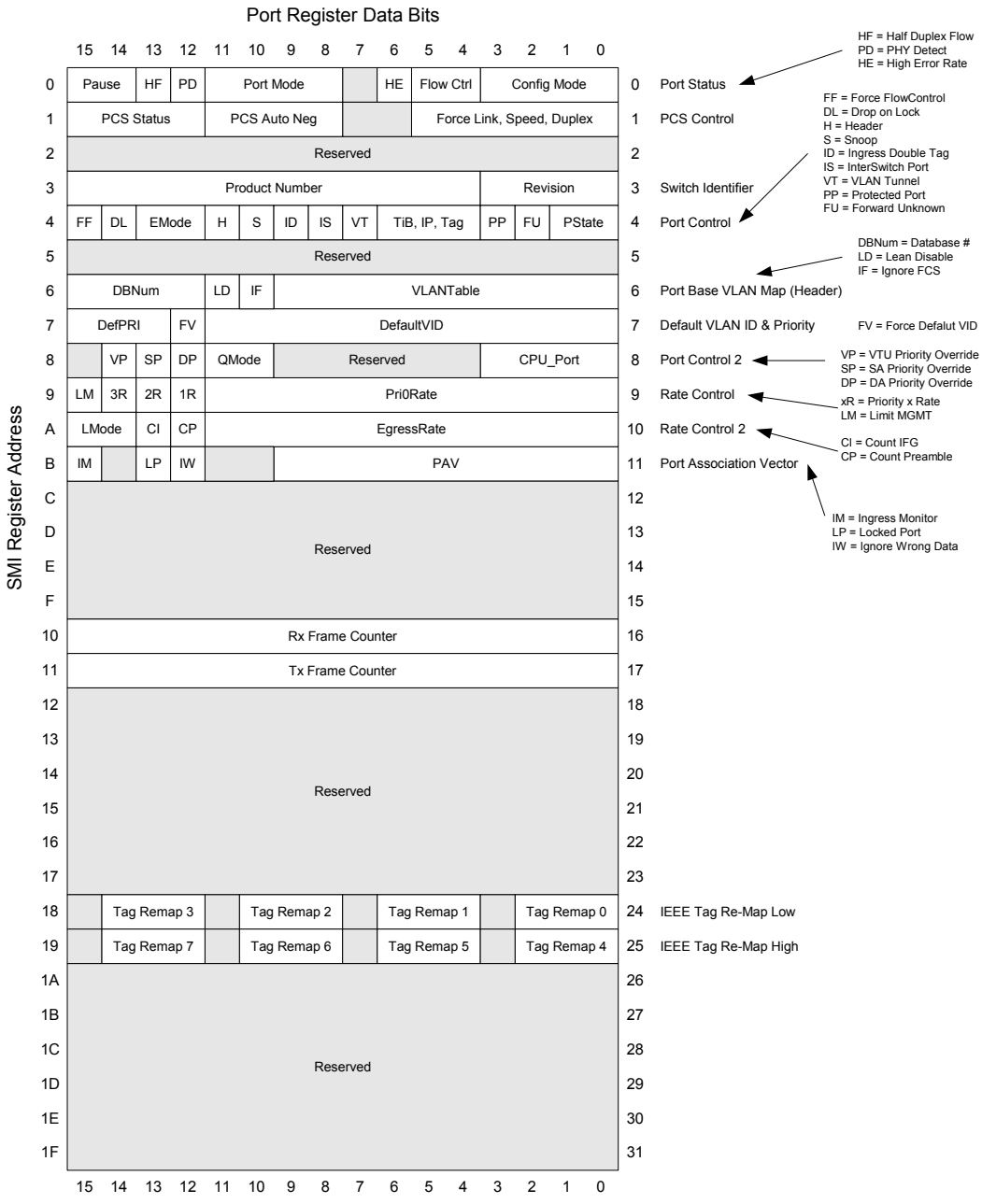
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Figure 5: 88E6183 Global Register Map



Each Ethernet port in the 88E6181/88E6183 device contains its own per port registers. Each per port register is 16-bits wide and their bit assignments are shown in Figure 6.

Figure 6: Per Port Register Bit Map



MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

The SOHO Switch Product registers fall into three categories:

- Switch port registers
- PHY registers
- Global registers

3.2.1 Switch Port Register

The 88E6181/88E6183 devices contain ten switch ports (MACs). The MAC units comply fully with the applicable sections of IEEE 802.3, IEEE 802.3u, and IEEE 802.3x standards. These registers are accessible using SMI device addresses 0x10 to 0x19.

3.2.2 PHY Registers

The 88E6181/88E6183 device can support up to ten external PHYs. These external devices are accessible using SMI device addresses 0x0 to 0x9.

3.2.3 Switch Global Register

The switch contains many global registers that are used to control features and functions that are common to all ports in the switch. The global registers are accessible using SMI device address 0x1B.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Section 4. Platform Specific Routine

SMI interface functions (which are target hardware specific) must be registered to QuarterDeck driver. Semaphore functions (which are OS specific) can also be registered to QuarterDeck driver, but they are optional. Sample programs (Sample\Initialization\msApiInit.c) will explain in detail how to register those routines.

4.1 SMI Interface Functions

SMI (MDC/MDIO) Interface functions are used to read/write SOHO Switch Product device registers. These functions must be provided before using any MS APIs.

4.1.1 SMI (MDC/MDIO) Read

DESCRIPTION

This function reads a data from the given PHY address and Register offset.

SYNOPSIS

```
GT_BOOL miiRead
(
        IN GT_QD_DEV  *dev
        IN unsigned int   phyAddress,
        IN unsigned int   regOffset,
        OUT unsigned int* data
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- phyAddress – Phy address to read the register for
- regOffset - The register offset

OUTPUTS:
- data - Points to a variable where a register data is stored.

RETURNS

GT_TRUE - On success
GT_FALSE - Otherwise

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

4.1.2 SMI (MDC/MDIO) Write

DESCRIPTION

This function writes a data into the given PHY address and Register offset.

SYNOPSIS

```
GT_BOOL miiWrite
(
        IN GT_QD_DEV  *dev
        IN unsigned int     phyAddress,
        IN unsigned int     regOffset,
        IN unsigned int     data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
phyAddress – Phy address to read the register for
regOffset - The register offset
data - Register data to be written.

OUTPUTS:

None

RETURNS

GT_TRUE - On success
GT_FALSE - Otherwise

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

4.2 Semaphore Related Routines

The user may optionally provide Semaphore related routines, which have only two states: either Full, or Empty (also known as binary semaphore). If these OS specific routines are provided, driver will use them when the MAC address table, 802.1Q VLAN table, or RMON counter in the device needs to be accessed. Create semaphore, Delete semaphore, Acquire semaphore, and Release Semaphore are the routines which may be provided.

4.2.1 Create Semaphore

DESCRIPTION

This routine creates semaphore.

SYNOPSIS

```
GT_SEM semCreate
(
        IN GT_SEM_BEGIN_STATE state
);
```

ARGUMENTS

INPUTS:

state – semaphore beginning state, either 0 (GT_SEM_EMPTY) or 1 (GT_SEM_FULL)

OUTPUT:

None.

RETURN

Semaphore ID if success, or 0 if failed.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

4.2.2 Delete Semaphore

DESCRIPTION

This routine deletes a semaphore.

SYNOPSIS

```
GT_STATUS semDelete
(
        IN GT_SEM semId
);
```

ARGUMENTS

INPUTS:
semId – semaphore ID which is given by semCreate.
OUTPUT:
None.

RETURN

GT_OK - On success
GT_FAIL – Otherwise

4.2.3 Acquire Semaphore

DESCRIPTION

This routine tries to take a semaphore.

SYNOPSIS

```
GT_STATUS semTake
(
        IN GT_SEM semId,
        IN GT_U32 time-out
);
```

ARGUMENTS

INPUTS:
semId – semaphore ID which is given by semCreate.
time-out – time-out in seconds
OUTPUT:
None.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

RETURN

GT_OK - On success
GT_FAIL – Otherwise

4.2.4 Release Semaphore

DESCRIPTION

This routine releases a semaphore.

SYNOPSIS

```
GT_STATUS semGive
(
        IN GT_SEM semId
);
```

ARGUMENTS

INPUTS:
semId – semaphore ID which is given by semCreate.
OUTPUT:
None.

RETURN

GT_OK - On success
GT_FAIL – Otherwise

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Section 5. QuarterDeck Driver Layer

The QuarterDeck Driver Layer is between the SMI Driver and Marvell Semiconductor API (MSAPI). This layer receives functionally-oriented requests from the MSAPI layer and invokes the SMI driver layer to perform the hardware register access. The QuarterDeck Driver Layer includes functions to access PHY registers, switch port registers, and switch global registers. It also provides functions to read and write bit fields of the above registers.

5.1 PHY Register Accesses

This section includes functions that access the PHY registers of the SOHO Switch product.

5.1.1 hwReadPhyReg()

DESCRIPTION

This function reads a switch's PHY register with a given port number and a SMI register address.

SYNOPSIS

```
GT_STATUS hwReadPhyReg
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        OUT GT_U16     *data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
portNum - Port number to read the register for
regAddr - The SMI register address

OUTPUTS:

data - The read register's data

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

SOUCE FILE

gtHwcntl.c

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

5.1.2 hwWritePhyReg

DESCRIPTION

This function writes to a PHY register with a given port number and SMI register address.

SYNOPSIS

```
GT_STATUS hwWritePhyReg
(
        IN GT_QD_DEV  *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        IN  GT_U16      data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
portNum - Port number to write the register for
regAddr - The SMI register address
data - The data to be written
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.1.3 hwGetPhyRegField

DESCRIPTION

This function reads a specified field from a PHY register.

SYNOPSIS

```
GT_STATUS hwGetPhyRegField
(
        IN  GT_QD_DEV   *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        IN  GT_U8       fieldOffset,
        IN  GT_U8       fieldLength,
        OUT GT_U16      *data
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
portNum - Port number to read the register for
regAddr - The register's SMI address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to read
OUTPUTS:
data - The read register field

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.1.4 hwSetPhyRegField

DESCRIPTION

This function writes to a specified field in a PHY register.

SYNOPSIS

```
GT_STATUS hwSetPhyRegField
(
    IN  GT_QD_DEV  *dev
    IN  GT_U8       portNum,
    IN  GT_U8       regAddr,
    IN  GT_U8       fieldOffset,
    IN  GT_U8       fieldLength,
    IN  GT_U16      data
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
portNum - Port number to write the register for
regAddr - The register's address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to write
data - Data to be written
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

5.2 Switch Port Register Accesses

This section includes functions that access the switch port registers of the SOHO Switch Product device.

5.2.1 hwReadPortReg

DESCRIPTION

This function reads a SOHO Switch Product switch port register with a given port number and SMI register address.

SYNOPSIS

```
GT_STATUS hwReadPortReg
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        OUT GT_U16      *data
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- portNum - Port number to read the register for
- regAddr - The register's address

OUTPUTS:
- data - The read register's data

RETURNS

GT_OK - On success
GT_FAIL - On error

5.2.2 hwWritePortReg

DESCRIPTION

This function writes to a SOHO Switch Product switch port register with a given port number and SMI register address.

SYNOPSIS

```
GT_STATUS hwWritePortReg
(
        IN  GT_QD_DEV  *dev
```

```
        IN  GT_U8   portNum,
        IN  GT_U8   regAddr,
        IN  GT_U16  data
    );
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
portNum - Port number to write the register for
regAddr - The register's address
data - The data to be written

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.2.3 hwGetPortRegField

DESCRIPTION

This function reads a specified field from a switch port register of SOHO Switch Product.

SYNOPSIS

```
GT_STATUS hwGetPortRegField
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        IN  GT_U8       fieldOffset,
        IN  GT_U8       fieldLength,
        OUT GT_U16      *data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
portNum - Port number to read the register for
regAddr - The register's address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to read

OUTPUTS:

data - The read register field

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

5.2.4 hwSetPortRegField

DESCRIPTION

This function writes to specified field in a switch port register of SOHO Switch Product.

SYNOPSIS

```
GT_STATUS hwSetPortRegField
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       portNum,
        IN  GT_U8       regAddr,
        IN  GT_U8       fieldOffset,
        IN  GT_U8       fieldLength,
        IN  GT_U16      data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
portNum - Port number to write the register for
regAddr - The register's address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to write
data - Data to be written

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

5.3 Global Register Accesses

This section includes functions that access the global registers of the SOHO Switch Product device.

5.3.1 hwReadGlobalReg

DESCRIPTION

This function reads a SOHO Switch Product global register.

SYNOPSIS

```
GT_STATUS hwReadGlobalReg
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       regAddr,
        OUT GT_U16      *data
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- regAddr - The register's address

OUTPUTS:
- data - The read register's data

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.3.2 hwWriteGlobalReg

DESCRIPTION

This function writes to a SOHO Switch Product global register.

SYNOPSIS

```
GT_STATUS hwWriteGlobalReg
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       regAddr,
        IN  GT_U16      data
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

regAddr - The register's address
data - The data to be written
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.3.3 hwGetGlobalRegField

DESCRIPTION

This function reads a specified field from a SOHO Switch Product global register.

SYNOPSIS

```
GT_STATUS hwGetGlobalRegField
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       regAddr,
        IN  GT_U8       fieldOffset,
        IN  GT_U8       fieldLength,
        OUT GT_U16      *data
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
regAddr - The register's address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to read
OUTPUTS:
data - The read register field

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

5.3.4 hwSetGlobalRegField

DESCRIPTION

This function writes to specified field in a SOHO Switch Product global register.

SYNOPSIS

```
GT_STATUS hwSetGlobalRegField
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8       regAddr,
        IN  GT_U8       fieldOffset,
        IN  GT_U8       fieldLength,
        IN  GT_U16      data
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
regAddr - The register's address
fieldOffset - The field start bit index (0 - 15)
fieldLength - Number of bits to write
data - Data to be written

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - Otherwise

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

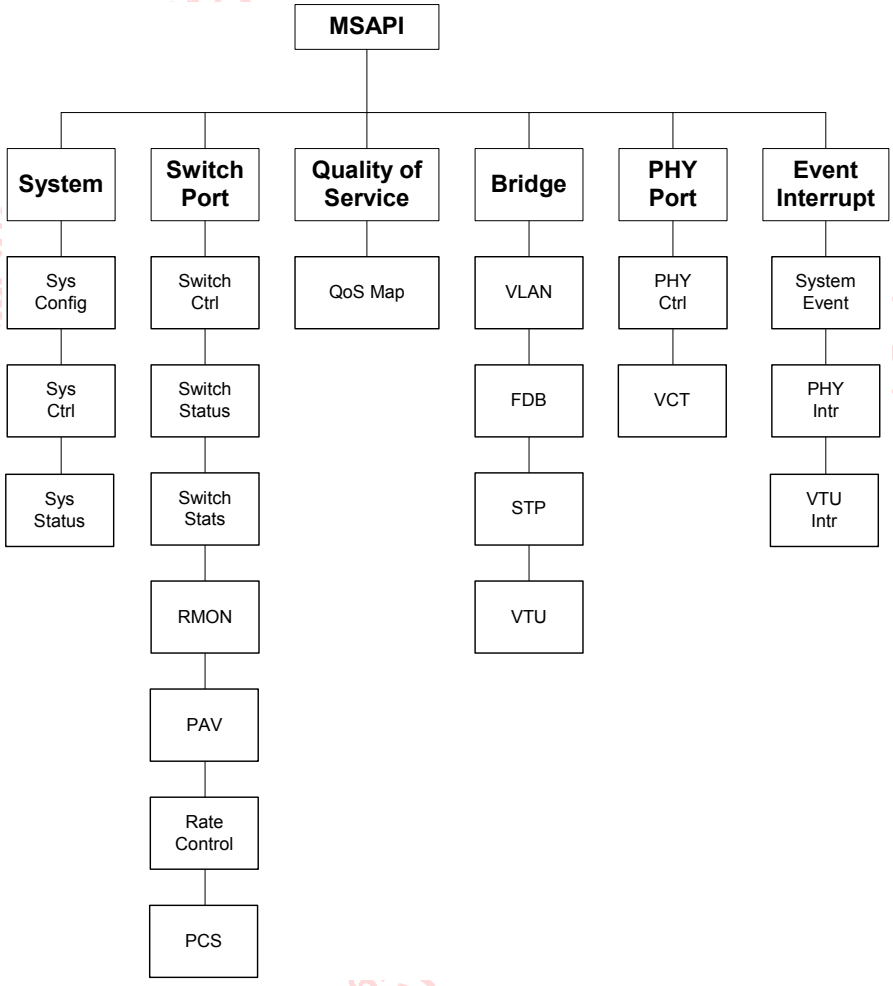
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Section 6. Marvell Semiconductor API Layer

The Marvell Semiconductor API Layer provides the Application Programming Interface functions. The MSAPI is the top layer of the QuarterDeck Driver Suite and the only software interface exposed to user applications. Through the well-defined MSAPI, the application development can be leveraged across multiple generations of products in the SOHO switching family.

The MSAPI consists of functionally-oriented modules or libraries. These modules are designed to be independent of each other, i.e., adding, removing, or making changes in one module has no effect on other modules. Consequently, users need only to include modules that are required for their applications. This feature adds to the software's scalability. Figure 7 shows the MSAPI module structure.

Figure 7: MSAPI/QD Modules



MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1 System Library

Table 1: System Library H files

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes	MS API prototypes

6.1.1 System Control

The Marvell Semiconductor API for system control is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtSysCtrl.c.

6.1.1.1 gsysSwReset

DESCRIPTION

This routine performs switch software reset.

SYNOPSIS

```
GT_STATUS gsysSwReset
(
        IN_GT_QD_DEV  *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.2 gsysSetDiscardExcessive

DESCRIPTION

This routine sets the Discard Excessive state.

SYNOPSIS

```
GT_STATUS gsysSetDiscardExcessive
(
        IN GT_QD_DEV  *dev
        IN GT_BOOL     en
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- en - GT_TRUE Discard is enabled, GT_FALSE otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.1.1.3 gsysGetDiscardExcessive

DESCRIPTION

This routine gets the Discard Excessive state.

SYNOPSIS

```
GT_STATUS gsysGetDiscardExcessive
(
        IN GT_QD_DEV  *dev
        IN GT_BOOL     en
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()

OUTPUTS:
- en - GT_TRUE Discard is enabled, GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.1.1.4 gsysSetSchedulingMode

DESCRIPTION

This routine sets the Scheduling Mode.

SYNOPSIS

```
GT_STATUS gsysSetSchedulingMode
(
IN GT_QD_DEV  *dev
IN GT_BOOL    mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
mode - GT_TRUE weighted fair queuing scheme,
GT_FALSE strict priority scheme

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.1.1.5 gsysGetSchedulingMode

DESCRIPTION

This routine gets the Scheduling Mode.

SYNOPSIS

```
GT_STATUS gsysGetSchedulingMode
(
IN GT_QD_DEV  *dev
OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

mode - GT_TRUE weighted fair queuing scheme,
GT_FALSE strict priority scheme

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.6 gsysSetMaxFrameSize

DESCRIPTION

This routine sets the maximum frame size allowed.

SYNOPSIS

```
GT_STATUS gsysSetMaxFrameSize
(
        IN GT_QD_DEV  *dev
        IN GT_BOOL    mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
mode - GT_TRUE maximum size 1522, GT_FALSE maximum size 1535

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.1.1.7 gsysGetMaxFrameSize

DESCRIPTION

This routine gets the maximum frame size allowed.

SYNOPSIS

```
GT_STATUS gsysGetMaxFrameSize
(
        IN  GT_QD_DEV  *dev
        OUT GT_BOOL    *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

mode - GT_TRUE maximum size 1522, GT_FALSE maximum size 1535

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.8 gsysReLoad

DESCRIPTION

This routine causes the switch to reload configurations from EEPROM.

SYNOPSIS

```
GT_STATUS gsysReLoad
(
        IN GT_QD_DEV  *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.1.1.9 gsysSetWatchDog

DESCRIPTION

This routine sets the WatchDog mode.

SYNOPSIS

```
GT_STATUS gsysSetWatchDog
(
IN GT_QD_DEV  *dev
IN GT_BOOL      en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
en - GT_TRUE enables, GT_FALSE disable
OUTPUTS: None

RETURNS

GT_OK   - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.10 gsysGetWatchDog

DESCRIPTION

This routine gets the WatchDog mode.

SYNOPSIS

```
GT_STATUS gsysGetWatchDog
(
        IN GT_QD_DEV  *dev
        OUT GT_BOOL  *en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
en - GT_TRUE enables, GT_FALSE disable

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

6.1.1.11 gsysSetDuplexPauseMac

DESCRIPTION

This routine sets the full-duplex pause source Mac Address.

SYNOPSIS

```
GT_STATUS gsysSetDuplexPauseMac
(
        IN GT_QD_DEV       *dev
        IN GT_ETHERADDR *mac
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
mac - The Mac address to be set
OUTPUTS: None

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.12 gsysGetDuplexPauseMac

DESCRIPTION

This routine gets the full-duplex pause source Mac Address.

SYNOPSIS

```
GT_STATUS gsysGetDuplexPauseMac
(
        IN GT_QD_DEV        *dev
        OUT GT_ETHERADDR *mac
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
mac - The Mac address

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

6.1.1.13 gsysSetPerPortDuplexPauseMac

DESCRIPTION

This routine sets whether the full-duplex pause source Mac Address is per port or per device.

SYNOPSIS

```
GT_STATUS gsysSetPerPortDuplexPauseMac
(
        IN GT_QD_DEV  *dev
        IN GT_BOOL      en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
en - GT_TURE per port mac, GT_FALSE global mac
OUTPUTS: None

RETURNS

GT_OK   - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.14 gsysGetPerPortDuplexPauseMac

DESCRIPTION

This routine gets whether the full-duplex pause source Mac Address is per port or per device.

SYNOPSIS

```
GT_STATUS gsysGetPerPortDuplexPauseMac
(
        IN GT_QD_DEV  *dev
        IN GT_BOOL    en
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
en - GT_TURE per port mac, GT_FALSE global mac

OUTPUTS: None

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error

6.1.1.15 gsysSetPPUEn

DESCRIPTION

This routine enables/disables Phy Polling Unit.

SYNOPSIS

```
GT_STATUS gsysSetPPUEn
(
        IN GT_QD_DEV  *dev,
        IN GT_BOOL    en
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
en - GT_TRUE to enable PPU, GT_FALSE otherwise.

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.16 gsysGetPPUEn

DESCRIPTION

This routine get the PPU state.

SYNOPSIS

```
GT_STATUS gsysGetPPUEn
(
        IN GT_QD_DEV  *dev,
        OUT GT_BOOL  *en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
en - GT_TRUE if PPU is enabled, GT_FALSE otherwise.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.1.1.17 gsysSetCascadePort

DESCRIPTION

This routine sets Cascade Port number.
In multichip systems frames coming from a CPU need to know when they have reached their destination chip.

SYNOPSIS

```
GT_STATUS gsysSetCascadePort
(
        IN GT_QD_DEV  *dev,
        IN GT_LPORT    port
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Cascade Port
OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.1.18 gsysGetCascadePort

DESCRIPTION

This routine gets Cascade Port number.
In multichip systems frames coming from a CPU need to know when they have reached their destination chip.

SYNOPSIS

```
GT_STATUS gsysGetCascadePort
(
        IN GT_QD_DEV  *dev,
        OUT GT_LPORT *port
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
port - Cascade Port

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.1.1.19 gsysSetDeviceNumber

DESCRIPTION

This routine sets Device Number.
In multichip systems frames coming from a CPU need to know when they have reached their destination chip. From CPU frames whose Dev_Num fieldmatches these bits have reached their destination chip and are sent out this chip using the port number indicated in the frame's Trg_Port field.

SYNOPSIS

```
GT_STATUS gsysSetDeviceNumber
(
        IN GT_QD_DEV  *dev,
        IN GT_U32        devNum
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

devNum - Device Number (0 ~ 31)

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.1.1.20 gsysGetDeviceNumber

DESCRIPTION

This routine gets Device Number.

SYNOPSIS

```
GT_STATUS gsysGetDeviceNumber
(
        IN GT_QD_DEV  *dev,
        OUT GT_U32  *devNum
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

devNum - Device Number (0 ~ 31)

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.2 System Status

6.1.2.1 gsysGetSW_Mode

DESCRIPTION

This routine gets the Switch mode. These two bits return the current value of the SW_MODE[1:0] pins.

SYNOPSIS

```
GT_STATUS gsysGetSW_Mode
(
        IN GT_QD_DEV     *dev
        IN GT_SW_MODE *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

mode - GT_TRUE Discard is enabled, GT_FALSE otherwise.

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.1.2.2 gsysGetInitReady

DESCRIPTION

This routine get the InitReady bit. This bit is set to a one when the ATU, the Queue Controller and the Statistics Controller are done with their initialization and are ready to accept frames.

SYNOPSIS

```
GT_STATUS gsysGetInitReady
(
        IN GT_QD_DEV     *dev
        IN GT_GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

mode - GT_TRUE Discard is enabled, GT_FALSE otherwise.

RETURNS

GT_OK - On success

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

GT_BAD_PARAM - On bad parameter
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.1.2.3 gsysGetPPUState

DESCRIPTION

This routine get the PPU State. These two bits return the current value of the PPU.

SYNOPSIS

```
GT_STATUS gsysGetSW_Mode
(
        IN GT_QD_DEV     *dev,
        OUT GT_PPU_STATE*mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
mode - GT_PPU_STATE

RETURNS

GT_OK - On success
GT_BAD_PARAM - On bad parameter
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.3 System Configuration

The API for system configuration is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtSysConfig.c.

6.1.3.1 qdLoadDriver

DESCRIPTION

QuarterDeck Driver Initialization Routine. This is the first routine that needs to be called by system software. It takes sysCfg from system software, and returns the information related to this SOHO Switch Product device into the GT_QD_DEV data structure which space is provided by system software. This structure pointer (*dev) is then used for all the API functions.

Current Driver supports the following Methods to scan the SMI Bus for SOHO Switch Product:

SMI_AUTO_SCAN_MODE,
SMI_MANUAL_MODE, and
SMI_MULTI_ADDR_MODE.

SMI_AUTO_SCAN_MODE can be used when there is only one Fast Ethernet SOHO Switch Product or only one Gigabit Switch Product with Single Address Mode setup. SMI_MANUAL_MODE can be used by providing baseAddr in GT_SCAN_MODE structure which is a field in GT_SYS_CONFIG structure.
SMI_MULTI_ADDR_MODE should be used when the given Gigabit Switch device is configured as Multi Address Mode. SMI_MULTI_ADDR_MODE also requires for baseAddr field in GT_SCAN_MODE structure to be set to the device's SMI address.

SYNOPSIS

```
GT_STATUS qdLoadDriver
(
        IN GT_SYS_CONFIG   *sysCfg
        OUT GT_QD_DEV      *dev
);
```

ARGUMENTS

INPUTS:
sysCfg - Holds system configuration parameters.
OUTPUTS:
dev - Holds general system information.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_ALREADY_EXIST - if device already started
GT_BAD_PARAM - on bad parameters

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.1.3.2 qdUnloadDriver

DESCRIPTION

This function unloads the QuarterDeck Driver.

SYNOPSIS

```
GT_STATUS qdUnloadDriver
(
        IN  GT_QD_DEV  *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
None.

RETURNS

GT_OK - On success
GT_FAIL - On error

6.1.3.3 sysEnable

DESCRIPTION

This function does nothing for this release, but may be used in future releases.

SYNOPSIS

```
GT_STATUS sysEnable
(
        IN GT_QD_DEV   *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2 Switch Port Library

Table 2: Switch Port Library H files

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes	MS API prototypes

6.2.1 Switch Port Control

The API for Switch Port Control is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPortCtrl.c.

6.2.1.1 gprtSetIGMPSnoop

DESCRIPTION

This routine sets the IGMP Snoop. When it is set to one and this port receives IGMP frame, the frame is switched to the CPU port, overriding all other switching decisions, with exception for CPU's Trailer.

SYNOPOSIS

```
GT_STATUS gprtSetIGMPSnoop
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- mode -GT_TRUE for IGMP Snoop or GT_FALSE otherwise

OUTPUTS:
- None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.2 gprtGetIGMPSnoop

DESCRIPTION

This routine gets the IGMP Snoop mode.

SYNOPOSIS

```
GT_STATUS gprtGetIGMPSnoop
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

mode - GT_TRUE for IGMP Snoop enabled
GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device.

6.2.1.3 gprtSetHeaderMode

DESCRIPTION

This routine sets the ingress and egress header mode of a switch port.

SYNOPOSIS

```
GT_STATUS gprtSetHeaderMode
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - GT_TRUE for header mode
GT_FALSE otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device.

6.2.1.4 gprtGetHeaderMode

DESCRIPTION

This routine gets the ingress and egress header mode of a switch port.

SYNOPOSIS

```
GT_STATUS gprtGetHeaderMode
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- mode - GT_TRUE header mode enabled
  GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device.

6.2.1.5 gprtSetForceFc

DESCRIPTION

This routine sets the force flow control state.

SYNOPOSIS

```
GT_STATUS gprtSetForceFc
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     force
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- force -GT_TRUE for force flow control or
  GT_FALSE otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.6 gprtGetForceFc

DESCRIPTION

This routine gets the force flow control state.

SYNOPOSIS

```
GT_STATUS gprtGetForceFc
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *force
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

force - GT_TRUE for force flow control or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.1.7 gprtSetTrailerMode

DESCRIPTION

This routine sets the egress trailer mode.

SYNOPOSIS

```
GT_STATUS gprtSetTrailerMode
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - GT_TRUE for add trailer or GT_FALSE otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.8 gprtGetTrailerMode

DESCRIPTION

This routine gets the egress trailer mode.

SYNOPOSIS

```
GT_STATUS gprtGetTrailerMode
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies the device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

mode - GT_TRUE for add trailer or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.1.9 gprtSetIngressMode

DESCRIPTION

This routine sets the ingress mode.

SYNOPOSIS

```
GT_STATUS gprtSetIngressMode
(
        IN GT_QD_DEV          *dev
        IN GT_LPORT           port,
        IN GT_INGRESS_MODE mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - The ingress mode

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.10 gprtGetIngressMode

DESCRIPTION

This routine gets the ingress mode.

SYNOPOSIS

```
GT_STATUS gprtGetIngressMode
(
        IN GT_QD_DEV            *dev
        IN  GT_LPORT            port,
        OUT GT_INGRESS_MODE *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

mode - The ingress mode

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.1.11 gprtSetMcRateLimit

DESCRIPTION

This routine sets the port multicast rate limit.

SYNOPOSIS

```
GT_STATUS gprtSetMcRateLimit
(
        IN GT_QD_DEV   *dev
        IN GT_LPORT      port,
        IN GT_MC_RATE  rate
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
rate - GT_TRUE to Enable, GT_FALSE for otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.12 gprtGetMcRateLimit

DESCRIPTION

This routine gets the port multicast rate limit.

SYNOPOSIS

```
GT_STATUS gprtGetMcRateLimit
(
        IN  GT_QD_DEV       *dev
        IN  GT_LPORT         port,
        OUT GT_MC_RATE  *rate
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number.

OUTPUTS:
- rate - GT_TRUE to Enable, GT_FALSE for otherwise.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.1.13 gprtSetProtectedMode

DESCRIPTION

This routine set protected mode of a switch port.
When this mode is set to GT_TRUE, frames are allowed to egress port defined by the 802.1Q VLAN membership for the frame's VID 'AND' by the port's VLANTable if 802.1Q is enabled on the port. Both must allow the frame to Egress.

SYNOPOSIS

```
GT_STATUS gprtSetProtectedMode
(
        IN  GT_QD_DEV   *dev,
        IN GT_LPORT     port,
        IN GT_BOOL      mode
);
```

ARGUMENTS

port - the logical port number.
mode - GT_TRUE for protected mode or GT_FALSE otherwise

RETURNS

GT_OK   - on success
GT_FAIL - on error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.14 gprtGetProtectedMode

DESCRIPTION

This routine gets protected mode of a switch port.
When this mode is set to GT_TRUE, frames are allowed to egress port defined by the 802.1Q VLAN membership for the frame's VID 'AND' by the port's VLANTable if 802.1Q is enabled on the port. Both must allow the frame to Egress.

SYNOPOSIS

```
GT_STATUS gprtGetProtectedMode
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        OUT GT_BOOL     *mode
);
```

ARGUMENTS

port - the logical port number.
mode - GT_TRUE: header mode enabled, GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error

6.2.1.15 gprtSetForwardUnknown

DESCRIPTION

This routine set Forward Unknown mode of a switch port.
When this mode is set to GT_TRUE, normal switch operation occurs.
When this mode is set to GT_FALSE, unicast frame with unknown DA addresses will not egress out this port.

SYNOPOSIS

```
GT_STATUS gprtSetForwardUnknown
(
        IN  GT_QD_DEV   *dev,
        IN GT_LPORT     port,
        IN GT_BOOL      mode
);
```

ARGUMENTS

port - the logical port number.
mode - GT_TRUE for protected mode, or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.16 gprtGetForwardUnknown

DESCRIPTION

This routine gets Forward Unknown mode of a switch port.
When this mode is set to GT_TRUE, normal switch operation occurs.
When this mode is set to GT_FALSE, unicast frame with unknown DA addresses will not egress out this port.

SYNOPSIS

```
GT_STATUS gprtGetForwardUnknown
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        OUT GT_BOOL     *mode
);
```

ARGUMENTS

port - the logical port number.
mode - GT_TRUE for protected mode, or GT_FALSE otherwise

RETURNS

GT_OK   - on success
GT_FAIL - on error

6.2.1.17 gprtSetDropOnLock

DESCRIPTION

This routine set the Drop on Lock. When set to GT_TRUE, Ingress frames will be discarded if their SA field is not in the ATU's address database.

SYNOPSIS

```
GT_STATUS gprtSetDropOnLock
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for Unknown SA drop or GT_FALSE otherwise

OUTPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.18 gprtGetDropOnLock

DESCRIPTION

This routine gets DropOnLock mode. When set to GT_TRUE, Ingress frames will be discarded if their SA field is not in the ATU's address database.

SYNOPSIS

```
GT_STATUS gprtGetDropOnLock
(
    IN  GT_QD_DEV    *dev,
    IN  GT_LPORT     port,
    OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:
mode - GT_TRUE for Unknown SA drop or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.19 gprtSetDoubleTag

DESCRIPTION

This routine set the Ingress Double Tag Mode. When set to GT_TRUE, ingressing frames are examined to see if they contain an 802.3ac tag. If they do, the tag is removed and then the frame is processed from

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

there (i.e., removed tag is ignored). Essentially, untagged frames remain untagged, single tagged frames become untagged and double tagged frames become single tagged.

SYNOPSIS

```
GT_STATUS gprtSetDoubleTag
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for DoulbeTag mode or GT_FALSE otherwise

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.20 gprtGetDoubleTag

DESCRIPTION

This routine gets the Ingress Double Tag Mode. When set to GT_TRUE, ingressing frames are examined to see if they contain an 802.3ac tag. If they do, the tag is removed and then the frame is processed from there (i.e., removed tag is ignored). Essentially, untagged frames remain untagged, single tagged frames become untagged and double tagged frames become single tagged.

SYNOPSIS

```
GT_STATUS gprtGetDoubleTag
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL      *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

port - The logical port number.

OUTPUTS:
mode - GT_TRUE for DoulbeTag mode or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.21 gprtSetInterswitchPort

DESCRIPTION

This routine set Interswitch Port. When set to GT_TRUE,
it indicates this port is a interswitch port used to communicated with
CPU or to cascade with another switch device.

SYNOPSIS

```
GT_STATUS gprtSetInterswitchPort
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for Interswitch port or GT_FALSE otherwise

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.22 gprtGetInterswithPort

DESCRIPTION

This routine gets Interswitch Port. When set to GT_TRUE,
it indicates this port is a interswitch port used to communicated with
CPU or to cascade with another switch device.

SYNOPSIS

```
GT_STATUS gprtGetInterswithPort
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

mode - GT_TRUE for Interswitch port or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.23 gprtSetLearnDisable

DESCRIPTION

This routine enables/disables automatic learning of new source MAC addresses
on the given port ingress

SYNOPSIS

```
GT_STATUS gprtSetLearnDisable
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for disable or GT_FALSE otherwise

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.24 gprtGetLearnDisable

DESCRIPTION

This routine gets LearnDisable setup

SYNOPSIS

```
GT_STATUS gprtGetLearnDisable
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:
mode - GT_TRUE for disable or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.25 gprtSetIgnoreFCS

DESCRIPTION

This routine sets FCS Ignore mode. When this bit is set to GT_TRUE, the last four bytes of frames received on this port are overwritten with a good CRC and the frames will be accepted by the switch.

SYNOPSIS

```
GT_STATUS gprtSetIgnoreFCS
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for ignore FCS or GT_FALSE otherwise

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.26 gprtGetIgnoreFCS

DESCRIPTION

This routine gets Ignore FCS setup.

SYNOPSIS

```
GT_STATUS gprtGetIgnoreFCS
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL      *mode
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:
mode - GT_TRUE for ignore FCS or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.27 gprtSetVTUPriOverride

DESCRIPTION

This routine sets VTU Priority Override. When this bit is set to GT_TRUE, VTU priority overrides can occur on this port.

SYNOPSIS

```
GT_STATUS gprtSetVTUPriOverride
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        IN  GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for VTU Priority Override or GT_FALSE otherwise

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.28 gprtGetVTUPriOverride

DESCRIPTION

This routine gets VTU Priority Override setup.

SYNOPSIS

```
GT_STATUS gprtGetVTUPriOverride
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

mode - GT_TRUE for VTU Priority Override or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.29 gprtSetSAPriOverride

DESCRIPTION

This routine sets SA Priority Override. When this bit is set to GT_TRUE, SA priority overrides can occur on this port.

SYNOPSIS

```
GT_STATUS gprtSetSAPriOverride
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for SA Priority Override or GT_FALSE otherwise

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.30 gprtGetSAPriOverride

DESCRIPTION

This routine gets SA Priority Override setup.

SYNOPSIS

```
GT_STATUS gprtGetSAPriOverride
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:
mode - GT_TRUE for SA Priority Override or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.31 gprtSetDAPriOverride

DESCRIPTION

This routine sets DA Priority Override. When this bit is set to GT_TRUE, DA priority overrides can occur on this port.

SYNOPSIS

```
GT_STATUS gprtSetDAPriOverride
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for DA Priority Override or GT_FALSE otherwise

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.32 gprtGetDAPriOverride

DESCRIPTION

This routine gets DA Priority Override setup.

SYNOPSIS

```
GT_STATUS gprtGetDAPriOverride
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:
mode - GT_TRUE for DA Priority Override or GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.33 gprtSetCPUPort

DESCRIPTION

This routine sets CPU Port number. When Snooping is enabled on this port or when this port is configured as an Interswitch Port and it receives a To_CPU frame, the switch needs to know what port on this device the frame should egress.

SYNOPSIS

```
GT_STATUS gprtSetCPUPort
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        IN  GT_LPORT    cpuPort
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
cpuPort - CPU Port number or interswitch port where CPU Port is connected

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.34 gprtGetCPUPort

DESCRIPTION

SYNOPSIS

```
GT_STATUS gprtGetCPUPort
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_LPORT *cpuLPort
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

cpuPort - CPU Port number or interswitch port where CPU Port is connected

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.35 gprtSetLockedPort

DESCRIPTION

This routine sets LockedPort. When it's set to one, CPU directed learning for 802.1x MAC authentication is enabled on this port. In this mode, an ATU Miss Violation interrupt will occur when a new SA address is received in a frame on this port. Automatically SA learning and refreshing is disabled in this mode.

SYNOPSIS

```
GT_STATUS gprtSetLockedPort
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for Locked Port, GT_FALSE otherwise

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.36 gprtGetLockedPort

DESCRIPTION

This routine gets Locked Port mode for the given port.

SYNOPSIS

```
GT_STATUS gprtGetLockedPort
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

mode - GT_TRUE for Locked Port, GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.1.37 gprtSetIgnoreWrongData

DESCRIPTION

This routine sets Ignore Wrong Data. If the frame's SA address is found in the database and if the entry is 'static' or if the port is 'locked' the source port's bit is checked to insure the SA has been assigned to this port. If the SA is NOT assigned to this port, it is considered an ATU Member Violation. If the IgnoreWrongData is set to GT_FALSE, an ATU Member Violation interrupt will be generated. If it's set to GT_TRUE, the ATU Member Violation error will be masked and ignored.

SYNOPSIS

```
GT_STATUS gprtSetIgnoreWrongData
(
        IN  GT_QD_DEV   *dev,
        IN  GT_LPORT    port,
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.
mode - GT_TRUE for IgnoreWrongData, GT_FALSE otherwise

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

6.2.1.38 gprtGetIgnoreWrongData

DESCRIPTION

This routine gets Ignore Wrong Data mode for the given port.

SYNOPSIS

```
GT_STATUS gprtGetIgnoreWrongData
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_BOOL   *mode
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number.

OUTPUTS:

mode - GT_TRUE for IgnoreWrongData, GT_FALSE otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.2 Switch Port Status

The API for Switch Port Status accesses is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPortStatus.c.

6.2.2.1 gprtGetPartnerLinkPause

DESCRIPTION

This routine retrieves the link partner pause state.

SYNOPSIS

```
GT_STATUS gprtGetPartnerLinkPause
(
        IN  GT_QD_DEV   *dev
        IN  GT_LPORT     port,
        OUT GT_BOOL     *state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE for enable or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.2 gprtGetSelfLinkPause

DESCRIPTION

This routine retrieves the link pause state.

SYNOPSIS

```
GT_STATUS gprtGetSelfLinkPause
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL    *state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

state - GT_TRUE for enable or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.3 gprtGetResolve

DESCRIPTION

This routine retrieves the resolve state.

SYNOPSIS

```
GT_STATUS gprtGetResolve
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number
OUTPUTS:
state - GT_TRUE for Done or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.4 gprtGetLinkState

DESCRIPTION

This routine retrieves the link state.

SYNOPSIS

```
GT_STATUS gprtGetLinkState
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *state
);
```

ARGUMENTS

INPUT:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

state - GT_TRUE for Up or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.5 gprtGetPortMode

DESCRIPTION

This routine retrieves the port mode.

SYNOPSIS

```
GT_STATUS gprtGetPortMode
(
        IN  GT_QD_DEV   *dev
        IN  GT_LPORT     port,
        OUT GT_BOOL     *mode
);
```

ARGUMENTS:

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
mode - GT_TRUE for MII or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.6 gprtGetPhyMode

DESCRIPTION

This routine retrieves the PHY mode.

SYNOPSIS

```
GT_STATUS gprtGetPhyMode
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL    *mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

port - The logical port number

OUTPUTS:

mode - GT_TRUE for MII PHY or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.7 gprtGetDuplex

DESCRIPTION

This routine retrieves the port duplex mode.

SYNOPSIS

```
GT_STATUS gprtGetDuplex
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

mode - GT_TRUE for Full or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.8 gprtSetDuplex

DESCRIPTION

This routine sets the duplex mode of MII/SNI/RMII ports.

SYNOPSIS

```
GT_STATUS gprtSetDuplex
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

port - The logical port number, either port 5 or port 6

OUTPUTS:

mode - GT_TRUE for Full-duplex or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.9 gprtGetSpeed

DESCRIPTION

This routine retrieves the port speed.

SYNOPSIS

```
GT_STATUS gprtGetSpeed
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL    *speed
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number, either port 5 or port 6

OUTPUTS:

speed - the logical number

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.2.10 gprtGetPauseEn

DESCRIPTION

This routine retrieves the pause state which indicates that Full Duplex flow control will be used on this port if the port is in Full Duplex mode. This value is valid only if gprtGetLinkState returns GT_TRUE.

SYNOPSIS

```
GT_STATUS
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

```
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
state - GT_TRUE for enable or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.11 gprtGetHdFlow

DESCRIPTION

This routine retrieves the half duplex flow control value.
If set, Half Duplex back pressure will be used on this port if this port is in a half duplex mode.

SYNOPSIS

```
GT_STATUS gprtGetHdFlow
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
state - GT_TRUE for enable or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.2.12 gprtGetPHYDetect

DESCRIPTION

This routine retrieves the information regarding PHY detection.
If set to GT_TRUE, An 802.3 PHY is attached to this port.

SYNOPSIS

```
GT_STATUS gprtGetPHYDetect
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if connected or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.13 gprtSetPHYDetect

DESCRIPTION

This routine sets PHYDetect bit which make PPU change its polling.
PPU's pool routine uses these bits to determine which port's to poll
PHYs on for Link, Duplex, Speed, and Flow Control.
This function should not be called if gsysGetPPUState returns PPU_STATE_ACTIVE.

SYNOPSIS

```
GT_STATUS gprtSetPHYDetect
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        IN  GT_BOOL     state
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
state - GT_TRUE or GT_FALSE
OUTPUTS:
None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.14 gprtGetSpeedMode

DESCRIPTION

This routine retrieves the port speed.

SYNOPSIS

```
GT_STATUS gprtGetSpeedMode
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_PORT_SPEED_MODE   *speed
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
mode - GT_PORT_SPEED_MODE type.
(PORT_SPEED_1000_MBPS,PORT_SPEED_100_MBPS, or PORT_SPEED_10_MBPS)

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.2.15 gprtGetHighErrorRate

DESCRIPTION

This routine retrieves the PCS High Error Rate.
This routine returns GT_TRUE if the rate of invalid code groups seen by PCS has exceeded 10 to the power of -11.

SYNOPSIS

```
GT_STATUS gprtGetHighErrorRate
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE or GT_FALSE

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.16 gprtGetTxPaused

DESCRIPTION

This routine retrieves Transmit Pause state.

SYNOPSIS

```
GT_STATUS gprtGetTxPaused
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE if Rx MAC receives a PAUSE frame with none-zero Puase Time
GT_FALSE otherwise.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.17 gprtGetFlowCtrl

DESCRIPTION

This routine retrieves Flow control state.

SYNOPSIS

```
GT_STATUS gprtGetFlowCtrl
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE if Rx MAC determines that no more data should be entering this port.
GT_FALSE otherwise.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.2.18 gprtGetC_Duplex

DESCRIPTION

This routine retrieves Port 9's duplex configuration mode determined at reset.

SYNOPSIS

```
GT_STATUS gprtGetC_Duplex
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE if configured as Full duplex operation
GT_FALSE otherwise.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.2.19 gprtGetC_Mode

DESCRIPTION

This routine retrieves port's interface type configuration mode determined at reset.

SYNOPSIS

```
GT_STATUS gprtGetC_Mode
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_PORT_CONFIG_MODE   *state
);
```

ARGUMENTS

INPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - one of value in GT_PORT_CONFIG_MODE enum type :
PORTCFG_GMII_125MHZ,/* Px_GTXCLK = 125MHz, 1000BASE - Port 9 Only */
PORTCFG_FD_MII_0MHZ,/* Px_GTXCLK = 0 MHz, Power Save - Port 9 Only */
PORTCFG_FDHD_MII_25MHZ,/* Px_GTXCLK = 25MHz, 100BASE - Port 9 Only */
PORTCFG_FDHD_MII_2_5MHZ,/* Px_GTXCLK = 2.5MHz, 10BASE - Port 9 Only */
PORTCFG_FD_SERDES,/* Default value */
PORTCFG_FD_1000BASE_X,/* Port 7,8,9 only */
PORTCFG_MGMII,/* duplex, speed determined by the PPU */
PORTCFG_DISABLED/* Port 9 Only */

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.3 Switch Port RMON Counter

The API for Port RMON Counter accesses is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPortRmon.c.

6.2.3.1 gstatsFlushAll

DESCRIPTION

Flush All RMON counters for all ports.

SYNOPSIS

```
GT_STATUS gstatsFlushAll
(
        IN GT_QD_DEV    *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.2 gstatsFlushPort

DESCRIPTION

Flush All RMON counters for a given port.

SYNOPSIS

```
GT_STATUS gstatsFlushPort
(
        IN GT_QD_DEV    *dev
        GT_LPORT        port
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
OUTPUTS:
None.

RETURNS

GT_OK - On success

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.3 gstatsGetPortCounter

DESCRIPTION

This routine gets a specific RMON counter of the given port

SYNOPSIS

```
GT_STATUS gstatsGetPortCounter
(
    IN  GT_QD_DEV           *dev
    IN  GT_LPORT            port,
    IN  GT_STATS_COUNTERS   counter,
    IN  GT_U32              *statsData
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver
- port - the logical port number.
- counter - the counter which will be read

OUTPUTS:
- statsData - points to 32bit data storage for the MIB counter

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.4 gstatsGetPortAllCounters

DESCRIPTION

This routine gets all RMON counters of the given port

SYNOPSIS

```
GT_STATUS gstatsGetPortAllCounters
(
    IN  GT_QD_DEV                   *dev
    IN  GT_LPORT                    port,
    OUT GT_STATS_COUNTER_SET        *statsCounterSet
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - the logical port number.

OUTPUTS:
statsCounterSet - points to GT_STATS_COUNTER_SET for the MIB counters

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.5 gstatsGetPortCounter2

DESCRIPTION

This routine gets a specific counter of the given port. GT_STATS_COUNTERS2 structure should be used for this routine, which will be supported by Gigabit Switch Family.

SYNOPSIS

```
GT_STATUS gstatsGetPortCounter2
(
        IN  GT_QD_DEV               *dev
        IN  GT_LPORT                port,
        IN  GT_STATS_COUNTERS2      counter,
        OUT GT_U32                  *statsData
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver
port - the logical port number.
counter - the counter which will be read
OUTPUTS:
statsData - points to 32bit data storage for the MIB counter

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.6 gstatsGetPortAllCounters2

DESCRIPTION

This routine gets all counters of the given port. GT_STATS_COUNTER_SET2 structure

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

should be used for this routine, which will be supported by Gigabit Switch Family.

SYNOPSIS

```
GT_STATUS gstatsGetPortAllCounters2
(
        IN  GT_QD_DEV               *dev
        IN  GT_LPORT                port,
        OUT GT_STATS_COUNTER_SET2*statsCounterSet
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver
port - the logical port number.

OUTPUTS:

statsCounterSet - points to GT_STATS_COUNTER_SET2 for the MIB counters

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.7 gstatsSetHistogramMode

DESCRIPTION

This routine sets the Histogram Counters Mode. Histogram Counters, such as 64Octets, 65to127Octets, 128to255Octests, and so forth, can have one of the following modes:
1. Count received frames only.
2. Count transmitted frames only.
3. Count both receive and transmitted frames.

SYNOPSIS

```
GT_STATUS gstatsSetHistogramMode
(
        IN  GT_QD_DEV               *dev,
        IN GT_HISTOGRAM_MODEmode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver
mode - Histogram Mode (GT_COUNT_RX_ONLY, GT_COUNT_TX_ONLY, and GT_COUNT_RX_TX)

OUTPUTS:

None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.3.8 gstatsGetHistogramMode

DESCRIPTION

This routine gets the Histogram Counters Mode. Histogram Counters, such as 64Octets, 65to127Octets, 128to255Octests, and so forth, can have one of the following modes:
1. Count received frames only.
2. Count transmitted frames only.
3. Count both receive and transmitted frames.

SYNOPSIS

```
GT_STATUS gstatsGetHistogramMode
(
        IN  GT_QD_DEV                   *dev,
        OUT GT_HISTOGRAM_MODE*mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver
OUTPUTS:
mode - Histogram Mode (GT_COUNT_RX_ONLY, GT_COUNT_TX_ONLY, and GT_COUNT_RX_TX)

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.2.4 Switch Port Rate Control

The API for switch Port Rate Control is defined in msApiDefs.h and msApiPrototypes.h. It is implemented in gtPortRateCtrl.c.

6.2.4.1 grcSetLimitMode

DESCRIPTION

This routine sets the port's rate control ingress limit mode.

SYNOPSIS

```
GT_STATUS grcSetLimitMode
(
        IN GT_QD_DEV            *dev
        IN GT_LPORT             port
        IN GT_RATE_LIMIT_MODE   mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number
- mode - rate control ingress limit mode

OUTPUTS: None

RETURNS

- GT_OK - On success
- GT_FAIL - On error
- GT_BAD_PARAM - On bad parameters
- GT_NOT_SUPPORTED - If not supported by the device

6.2.4.2 grcGetLimitMode

DESCRIPTION

This routine gets the port's rate control ingress limit mode.

SYNOPSIS

```
GT_STATUS grcGetLimitMode
(
        IN  GT_QD_DEV               *dev
        IN  GT_LPORT                port,
        OUT GT_RATE_LIMIT_MODE      *mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number.

OUTPUTS:
- mode - rate control ingress limit mode.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.3 grcSetPri3Rate

DESCRIPTION

This routine sets the ingress data rate limit for priority 3 frames. Priority 3 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

```
GT_STATUS grcSetPri3Rate
(
        IN  GT_QD_DEV  *dev
        IN GT_LPORT     port,
        IN GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
mode - the priority 3 frame rate limit mode
GT_FALSE: use the same rate as Pri2Rate
GT_TRUE: use twice the rate as Pri2Rate

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.4 grcGetPri3Rate

DESCRIPTION

This routine gets the ingress data rate limit for priority 3 frames. Priority 3 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

```
GT_STATUS grcGetPri3Rate
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
OUTPUTS:
mode - the priority 3 frame rate limit mode
GT_FALSE: use the same rate as Pri2Rate
GT_TRUE: use twice the rate as Pri2Rate

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.5 grcSetPri2Rate

DESCRIPTION

This routine sets the ingress data rate limit for priority 2 frames. Priority 2 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

```
GT_STATUS grcSetPri2Rate
(
    IN GT_QD_DEV  *dev
    IN GT_LPORT    port,
    IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
mode - the priority 2 frame rate limit mode
GT_FALSE: use the same rate as Pri1Rate
GT_TRUE: use twice the rate as Pri1Rate
OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORTED - If not supported by the device.

6.2.4.6 grcGetPri2Rate

DESCRIPTION

This routine gets the ingress data rate limit for priority 2 frames. Priority 2 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
GT_STATUS grcGetPri2Rate
(
        IN  GT_QD_DEV   *dev
        IN  GT_LPORT     port,
        OUT GT_BOOL    *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - the logical port number.

OUTPUTS:

mode - the priority 2 frame rate limit mode
GT_FALSE: use the same rate as Pri1Rate
GT_TRUE: use twice the rate as Pri1Rate

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device.

6.2.4.7 grcSetPri1Rate

DESCRIPTION

This routine sets the ingress data rate limit for priority 1 frames. Priority 1 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

```
GT_STATUS grcSetPri1Rate
(
        IN GT_QD_DEV   *dev
        IN GT_LPORT     port,
        IN GT_BOOL      mode
);*
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
mode - the priority 1 frame rate limit mode
GT_FALSE: use the same rate as Pri0Rate
GT_TRUE: use twice the rate as Pri0Rate

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORTED - If not supported by the device.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.4.8 grcGetPri1Rate

DESCRIPTION

This routine gets the ingress data rate limit for priority 1 frames. Priority 1 frames will be discarded after the ingress rate selection is reached or exceeded.

SYNOPSIS

```
GT_STATUS grcGetPri1Rate
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS

dev - specifies device context returned by qdLoadDriver()
port - the logical port number.

OUTPUTS:

mode - the priority 1 frame rate limit mode
GT_FALSE: use the same rate as Pri0Rate
GT_TRUE: use twice the rate as Pri0Rate

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device.

6.2.4.9 grcSetPri0Rate

DESCRIPTION

This routine sets the port's ingress data limit for priority 0 frames.

SYNOPSIS

```
GT_STATUS grcSetPri0Rate
(
        IN GT_QD_DEV     *dev
        IN GT_LPORT       port,
        IN GT_PRI0_RATE  rate
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - logical port number.
rate - ingress data rate limit for priority 0 frames. These frames
will be discarded after the ingress rate selected is reached or exceeded.

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.10 grcGetPri0Rate

DESCRIPTION

This routine gets the port's ingress data limit for priority 0 frames.

SYNOPSIS

```
GT_STATUS grcGetPri0Rate
(
        IN  GT_QD_DEV       *dev
        IN  GT_LPORT        port,
        OUT GT_PRI0_RATE    *rate
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - logical port number to set.
OUTPUTS:
rate - ingress data rate limit for priority 0 frames. These frames will be discarded after the ingress rate selected is reached or exceeded.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.11 grcSetBytesCount

DESCRIPTION

This routine sets the bytes to count for limiting needs to be determined.

SYNOPSIS

```
GT_STATUS grcSetBytesCount
(
        IN GT_QD_DEV      *dev
        IN GT_LPORT       port,
        IN GT_BOOL limitMGMT,
        IN GT_BOOL countIFG,
        IN GT_BOOL countPre
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - logical port number to set.
limitMGMT - GT_TRUE: To limit and count MGMT frame bytes
GT_FALSE: otherwise

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

countIFG - GT_TRUE: To count IFG bytes
GT_FALSE: otherwise
countPre - GT_TRUE: To count Preamble bytes
GT_FALSE: otherwise

OUTPUTS:
None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.12 grcGetBytesCount

DESCRIPTION

This routine gets the bytes to count for limiting needs to be determined.

SYNOPSIS

```
GT_STATUS grcGetBytesCount
(
    IN GT_QD_DEV     *dev
    IN GT_LPORT      port,
    IN GT_BOOL       *limitMGMT,
    IN GT_BOOL       *countIFG,
    IN GT_BOOL       *countPre
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - logical port number

OUTPUTS:
limitMGMT - GT_TRUE: To limit and count MGMT frame bytes
GT_FALSE: otherwise
countIFG - GT_TRUE: To count IFG bytes
GT_FALSE: otherwise
countPre - GT_TRUE: To count Preamble bytes
GT_FALSE: otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.13 grcSetEgressRate

DESCRIPTION

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

This routine sets the port's egress data limit.

SYNOPSIS

```
GT_STATUS grcSetEgressRate
(
        IN GT_QD_DEV            *dev
        IN GT_LPORT             port,
        IN GT_EGRESS_RATE  rate
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number.
- rate - egress data rate limit.

OUTPUTS:
- None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.4.14 grcGetEgressRate

DESCRIPTION

This routine gets the port's egress data limit.

SYNOPSIS

```
GT_STATUS grcGetEgressRate
(
        In GT_QD_DEV            *dev
        IN  GT_LPORT             port,
        OUT GT_EGRESS_RATE  *rate
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number.

OUTPUTS:
- rate - egress data rate limit.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.5 Switch Port Association Vector

The API for switch Port Association Vector is defined in msApiDefs.h and msApiPrototypes.h. It is implemented in gtPortPav.c

6.2.5.1 gpavSetPAV

DESCRIPTION

This routine sets the Port Association Vector.

SYNOPSIS

```
GT_STATUS gpavSetPAV
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_U16      pav
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - logical port number.
pav - Port Association Vector

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.5.2 gpavGetPAV

DESCRIPTION

This routine gets the Port Association Vector.

SYNOPSIS

```
GT_STATUS gpavGetPAV
(
        IN GT_QD_DEV *dev
        IN  GT_LPORT   port,
        OUT GT_U16     *pav
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - logical port number.

OUTPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

pav - Port Association Vector

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.2.5.3 gpavSetIngressMonitor

DESCRIPTION

This routine sets the Ingress Monitor bit in the PAV.

SYNOPSIS

```
GT_STATUS gpavSetIngressMonitor
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
mode - the ingress monitor bit in the PAV
    GT_FALSE: Ingress Monitor enabled
    GT_TRUE: Ingress Monitor disabled

OUTPUTS:

None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORTED - If not supported by the device

6.2.5.4 gpavGetIngressMonitor

DESCRIPTION

This routine gets the Ingress Monitor bit in the PAV.

SYNOPSIS

```
GT_STATUS gpavGetIngressMonitor
(
        IN  GT_QD_DEV   *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL     *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - the logical port number.

OUTPUTS:
mode - the ingress monitor bit in the PAV
GT_FALSE: Ingress Monitor enabled
GT_TRUE: Ingress Monitor disabled

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.6 Switch Port Statistics

The API for switch port statistics accesses is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPortStat.c.

6.2.6.1 gprtSetCtrMode

DESCRIPTION

This routine sets the port Rx/Tx counters mode of operation.

SYNOPSIS

```
GT_STATUS gprtSetCtrMode
(
        IN GT_QD_DEV    *dev
        IN GT_CTR_MODE  mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
mode - The counter mode

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.2.6.2 gprtClearAllCtr

DESCRIPTION

This routine clears all port Rx/Tx counters.

SYNOPSIS

```
GT_STATUS gprtClearAllCtr
(
        IN GT_QD_DEV   *dev
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.6.3 gprtGetPortCtr

DESCRIPTION

This routine gets the port Rx/Tx counters.

SYNOPSIS

```
GT_STATUS gprtGetPortCtr
(
        IN  GT_QD_DEV           *dev
        IN  GT_LPORT            port,
        OUT GT_PORT_STAT        *ctr
);
```

ARGUMENTS

INPUTS:

dev - specifies the device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

ctr - The counters value

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.7 PCS Control

The API for PCS Control is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPCSCtrl.c.

6.2.7.1 gpcsGetCommaDet

DESCRIPTION

This routine retrieves Comma Detection status in PCS.

SYNOPOSIS

```
GT_STATUS gpcsGetCommaDet
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT   port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
state - GT_TRUE for Comma Detected or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.2 gpcsGetSyncOK

DESCRIPTION

This routine retrieves SynOK bit. It is set to a one when the PCS has detected a few comma patterns and is synchronized with its peer PCS layer.

SYNOPOSIS

```
GT_STATUS gpcsGetSyncOK
(
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if synchronized or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.3 gpcsGetSyncFail

DESCRIPTION

This routine retrieves SynFail status.

SYNOPOSIS

```
GT_STATUS gpcsGetSyncFail
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if synchronization failed or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.7.4 gpcsGetAnBypassed

DESCRIPTION

This routine retrieves Inband Auto-Negotiation bypass status.

SYNOPOSIS

```
GT_STATUS gpcsGetAnBypassed
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE if AN is bypassed or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.5 gpcsGetAnBypassMode

DESCRIPTION

This routine retrieves Enable mode of Inband Auto-Negotiation bypass.

SYNOPOSIS

```
GT_STATUS gpcsGetAnBypassMode
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

OUTPUTS:
mode - GT_TRUE if AN bypass is enabled or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.6 gpcsSetAnBypassMode

DESCRIPTION

This routine enables/disables mode of Inband Auto-Negotiation bypass.

SYNOPOSIS

```
GT_STATUS gpcsSetAnBypassMode
(
    IN GT_QD_DEV *dev,
    IN GT_LPORT    port,
    IN GT_BOOL  mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - GT_TRUE to enable AN bypass mode or GT_FALSE otherwise
OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.7 gpcsGetPCSAnEn

DESCRIPTION

This routine retrieves Enable mode of PCS Inband Auto-Negotiation.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

SYNOPOSIS

```
GT_STATUS gpcsGetPCSAnEn
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- mode - GT_TRUE if PCS AN is enabled or GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.8 gpcsSetPCSAnEn

DESCRIPTION

This routine sets Enable mode of PCS Inband Auto-Negotiation.

SYNOPOSIS

```
GT_STATUS gpcsSetPCSAnEn
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        IN GT_BOOL  mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- mode - GT_TRUE to enable PCS AN mode or GT_FALSE otherwise

OUTPUTS:
- None

RETURNS

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.9 gpcsSetRestartPCSAn

DESCRIPTION

This routine restarts PCS Inband Auto-Negotiation.

SYNOPOSIS

```
GT_STATUS gpcsSetRestartPCSAn
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.10 gpcsGetPCSAnDone

DESCRIPTION

This routine retrieves completion information of PCS Auto-Negotiation.

SYNOPOSIS

```
GT_STATUS gpcsGetPCSAnDone
(
        IN GT_QD_DEV *dev,
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
        IN GT_LPORT    port,
        OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- mode - GT_TRUE if PCS AN is done or never done
  GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.11 gpcsSetLinkValue

DESCRIPTION

This routine sets Link's force value.

SYNOPOSIS

```
GT_STATUS gpcsSetLinkValue
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        IN GT_BOOL     state

);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- state - GT_TRUE to force link up, GT_FALSE otherwise

OUTPUTS:
- None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.7.12 gpcsGetLinkValue

DESCRIPTION

This routine retrieves Link Value which will be used for Forcing Link up or down.

SYNOPOSIS

```
GT_STATUS gpcsGetLinkValue
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if Link Force value is one (link up)
  GT_FALSE otherwise (link down)

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.13 gpcsSetForcedLink

DESCRIPTION

This routine forces Link. If LinkValue is set to one, calling this routine with GT_TRUE will force Link to be up.

SYNOPOSIS

```
GT_STATUS gpcsSetForcedLink
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
        IN GT_BOOL     state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- state - GT_TRUE to force link (up or down), GT_FALSE otherwise

OUTPUTS:
- None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.14 gpcsGetForcedLink

DESCRIPTION

This routine retrieves Forced Link bit.

SYNOPOSIS

```
GT_STATUS gpcsGetForcedLink
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if ForcedLink bit is one,
  GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.7.15 gpcsSetDpxValue

DESCRIPTION

This routine sets Duplex's Forced value. This function needs to be called prior to gpcsSetForcedDpx.

SYNOPOSIS

```
GT_STATUS gpcsSetDpxValue
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        IN GT_BOOL     state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
state - GT_TRUE to force full duplex, GT_FALSE otherwise

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.16 gpcsGetDpxValue

DESCRIPTION

This routine retrieves Duplex's Forced value.

SYNOPOSIS

```
GT_STATUS gpcsGetDpxValue
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_TRUE if Duplex's Forced value is set to Full duplex,
GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.17 gpcsSetForcedDpx

DESCRIPTION

This routine forces duplex mode. If DpxValue is set to one, calling this routine with GT_TRUE will force duplex mode to be full duplex.

SYNOPOSIS

```
GT_STATUS gpcsSetForcedDpx
(
    IN GT_QD_DEV *dev,
    IN GT_LPORT    port,
    IN GT_BOOL     state
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
state - GT_TRUE to force duplex mode, GT_FALSE otherwise

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.2.7.18 gpcsGetForcedDpx

DESCRIPTION

This routine retrieves Forced Duplex.

SYNOPOSIS

```
GT_STATUS gpcsGetForcedDpx
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_BOOL  *state
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- state - GT_TRUE if ForcedDpx bit is one,
  GT_FALSE otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.19 gpcsSetForceSpeed

DESCRIPTION

This routine forces speed.

SYNOPOSIS

```
GT_STATUS gpcsSetForceSpeed
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        IN GT_PORT_FORCED_SPEED_MODE  mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

mode - GT_PORT_FORCED_SPEED_MODE (10, 100, 1000, or No Speed Force)

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.2.7.20 gpcsGetForceSpeed

DESCRIPTION

This routine retrieves Force Speed value.

SYNOPOSIS

```
GT_STATUS gpcsGetForceSpeed
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT    port,
        OUT GT_PORT_FORCED_SPEED_MODE   *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

state - GT_PORT_FORCED_SPEED_MODE (10, 100, 1000, or no force speed)

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.3 Quality of Service Library

Table 3: QoS Library H File

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes.h	MS API Prototypes

6.3.1 QoS Map

The API for QoS mapping is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtQosMap.c.

6.3.1.1 gcosSetPortDefaultTc

DESCRIPTION

This routine sets the default traffic class for a specific port.

SYNOPSIS

```
GT_STATUS gcosSetPortDefaultTc
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_U8       trafClass
);
```

ARGUMENTS

INPUTS:
- port - Logical port number
- trafClass - Default traffic class of a port

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.2 gcosGetPortDefaultTc

DESCRIPTION

This routine gets the default traffic class for a specific port.

SYNOPSIS

```
GT_STATUS gcosGetPortDefaultTc
```

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_U8        *trafClass
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver().
port - logical port numberOUTPUTS: None

OUTPUTS:

trafClass - default traffic class of a port

RETURNS

GT_OK - On success
GT_FAIL - On error

6.3.1.3 gqosSetPrioMapRule

DESCRIPTION

This routine sets priority-mapping rule.

SYNOPSIS

```
GT_STATUS gqosSetPrioMapRule
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - GT_TRUE for user priority rule,
GT_FALSE for otherwise.

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.4 gqosGetPrioMapRule

DESCRIPTION

This routine gets the priority-mapping rule.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

SYNOPSIS

```
GT_STATUS gqosGetPrioMapRule
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver
port - The logical port number

OUTPUTS

mode - GT_TRUE for user prio rule,
GT_FALSE for otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.5 gqosIpPrioMapEn

DESCRIPTION

This routine enables the IP priority mapping.

SYNOPSIS

```
GT_STATUS gqosIpPrioMapEn
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     *en
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
en - GT_TRUE to Enable, GT_FALSE for otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.3.1.6 gqosGetIpPrioMapEn

DESCRIPTION

This routine returns the IP priority mapping state.

SYNOPSIS

```
GT_STATUS gqosGetIpPrioMapEn
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL   *en
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- en - GT_TRUE for user prio rule, GT_FALSE, otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.7 gqosUserPrioMapEn

DESCRIPTION

This routine enables the user priority mapping.

SYNOPSIS

```
GT_STATUS gqosUserPrioMapEn
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     en
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number
- en - GT_TRUE to Enable, GT_FALSE for otherwise

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.3.1.8 gqosGetUserPrioMapEn

DESCRIPTION

This routine returns the user priority mapping state.

SYNOPSIS

```
GT_STATUS gqosGetUserPrioMapEn
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *en
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - The logical port number

OUTPUTS:
- en - GT_TRUE for user prio rule, GT_FALSE for otherwise

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.9 gcosGetUserPrio2Tc

DESCRIPTION

This routine gets the traffic class number for a specific 802.1p user priority.

SYNOPSIS

```
GT_STATUS gcosGetUserPrio2Tc
(
        IN  GT_QD_DEV  *dev
        IN  GT_U8          userPrior,
        OUT GT_U8       *trClass
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- userPrior - User priority

OUTPUTS:
- trClass - The Traffic Class the received frame is assigned

RETURNS

GT_OK - On success
GT_FAIL - On error

GT_NOT_SUPPORTED - If not supported by the device

6.3.1.10 gcosSetUserPrio2Tc

DESCRIPTION

This routine sets the traffic class number for a specific 802.1p user priority.

SYNOPSIS

```
GT_STATUS gcosSetUserPrio2Tc
(
        IN GT_QD_DEV *dev
        IN GT_U8     userPrior,
        IN GT_U8     trClass
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- userPrior - User priority of a port
- trClass - The Traffic Class the received frame is assigned

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.11 gcosGetDscp2Tc

DESCRIPTION

This routine retrieves the traffic class assigned for a specific IPv4 Description.

SYNOPSIS

```
GT_STATUS gcosGetDscp2Tc
(
        IN GT_QD_DEV *dev
        IN  GT_U8    dscp,
        OUT GT_U8    *trClass
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- dscp - The IPv4 frame description to query

OUTPUTS:
- trClass - The Traffic Class the received frame is assigned

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.12 gcosSetDscp2Tc

DESCRIPTION

This routine sets the traffic class assigned for a specific IPv4 Dscp.

SYNOPSIS

```
GT_STATUS gcosSetDscp2Tc
(
        IN GT_QD_DEV *dev
        IN GT_U8      dscp,
        IN GT_U8      trClass
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
dscp - The IPv4 frame description to map
trClass - The Traffic Class the received frame is assigned
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.13 gqosSetTagRemap

DESCRIPTION

Sets the remapped priority value for a specific 802.1p priority on a given port.

SYNOPSIS

```
GT_STATUS gqosSetTagRemap
(
        IN GT_QD_DEV *dev,
        IN GT_LPORT port,
        IN GT_U8    pri,
        IN GT_U8    remappedPri
);
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
pri - 802.1p priority
remappedPri - remapped Priority
OUTPUTS:
None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.3.1.14 gqosGetTagRemap

DESCRIPTION

Gets the remapped priority value for a specific 802.1p priority on a given port.

SYNOPSIS

```
GT_STATUS gqosGetTagRemap
(
        IN GT_QD_DEV *dev,
        IN  GT_LPORT port,
        IN  GT_U8      pri,
        OUT GT_U8   *remappedPri
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - the logical port number.
pri - 802.1p priority
OUTPUTS:
remappedPri - remapped Priority

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.4 Bridge Library

Table 4: Bridge Library H Files

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes	MS API prototypes

6.4.1 Port Based VLAN

The API for switch port based VLAN configuration is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtBrgVlan.c.

6.4.1.1 gprtSetEgressMode

DESCRIPTION

This routine sets the egress mode.

SYNOPSIS

```
GT_STATUS gprtSetEgressMode
(
        IN GT_QD_DEV        *dev
        IN GT_LPORT         port,
        IN GT_EGRESS_MODE   mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - The egress mode

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.4.1.2 gprtGetEgressMode

DESCRIPTION

This routine gets the egress mode.

SYNOPSIS

```
GT_STATUS gprtGetEgressMode
(
        IN GT_QD_DEV            *dev
        IN  GT_LPORT            port,
        OUT GT_EGRESS_MODE      *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

mode - The egress mode

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.1.3 gprtSetVlanTunnel

DESCRIPTION

This routine sets the vlan tunnel mode.

SYNOPSIS

```
GT_STATUS gprtSetVlanTunnel
(
        IN GT_QD_DEV    *dev
        IN GT_LPORT     port,
        IN GT_BOOL      mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number
mode - The vlan tunnel mode

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.4.1.4 gprtGetVlanTunnel

DESCRIPTION

This routine gets the vlan tunnel mode.

SYNOPSIS

```
GT_STATUS gprtGetVlanTunnel
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_BOOL  *mode
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS:

mode - The vlan tunnel mode

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.1.5 gvlnSetPortVlanDBNum

DESCRIPTION

This routine sets the port's default VLAN database number (DBNum).

SYNOPSIS

```
GT_STATUS gvlnSetPortVlanDBNum
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT     port,
        IN GT_U8          DBNum
);
```

ARGUMENTS

INPUTS

dev - specifies device context returned by qdLoadDriver()
port - logical port number to set.
DBNum - default database number for this port

OUTPUTS:

None.

RETURNS

IN GT_INGRESS_MODE mode
GT_OK - On success

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_FAIL - On error
GT_BAD_PARAM - on bad parameters

6.4.1.6 gvlnGetPortVlanDBNum

DESCRIPTION

This routine gets the port's default VLAN database number (DBNum).

SYNOPSIS

```
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_U8       *DBNum
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - logical port number to get.

OUTPUTS:
DBNum - default database number for this port

RETURNS:

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters

6.4.1.7 gvlnSetPortVlanDot1qMode

DESCRIPTION

This routine sets the IEEE 802.1q mode for this port.

SYNOPSIS

```
GT_STATUS gvlnSetPortVlanDot1qMode
(
        IN GT_QD_DEV          *dev
        IN GT_LPORT           port,
        IN GT_DOT1Q_MODE  mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

port - logical port number to set.

OUTPUTS:
mode - 802.1q mode for this port

RETURNS

IN GT_INGRESS_MODE mode
GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.4.1.8 gvlnGetPortVlanDot1qMode

DESCRIPTION

This routine gets the IEEE 802.1q mode for this port.

SYNOPSIS

```
GT_STATUS gvlnSetPortVlanDot1qMode
(
        IN GT_QD_DEV        *dev
        IN GT_LPORT         port,
        IN GT_DOT1Q_MODE    mode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - logical port number to set.

OUTPUTS:
mode - 802.1q mode for this port

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.4.1.9 gvlnSetPortVlanForceDefaultVID

DESCRIPTION

This routine sets the mode for forcing to use default VID.

SYNOPSIS

```
GT_STATUS gvlnSetPortVlanForceDefaultVID
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number to set.
- mode - GT_TRUE, force to use default VID
  GT_FALSE, otherwise

OUTPUTS:
- None.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT_SUPPORTED - If not supported by the device

6.4.1.10 gvlnGetPortVlanForceDefaultVID

DESCRIPTION

This routine gets the port mode for ForceDefaultVID bit.

SYNOPSIS

```
GT_STATUS gvlnGetPortVlanForceDefaultVID
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     mode
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - logical port number to set.

OUTPUTS:
- mode - GT_TRUE, force to use default VID
  GT_FALSE, otherwise

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_BAD_PARAM - on bad parameters
GT_NOT _SUPPORTED - If not supported by the device

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.4.1.11 gvlnSetPortVlanPorts

DESCRIPTION

This routine sets the port VLAN group port membership list.

SYNOPSIS

```
GT_STATUS gvlnSetPortVlanPorts
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_LPORT    memPorts[ ],
        IN GT_U8       memPortsLen
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - Logical port number to set
memPorts - Array of logical ports
memPortsLen - Number of members in memPorts array

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

6.4.1.12 gvlnGetPortVlanPorts

DESCRIPTION

This routine gets the port VLAN group port membership list.

SYNOPSIS

```
GT_STATUS gvlnGetPortVlanPorts
(
        IN GT_QD_DEV   *dev
        IN  GT_LPORT     port,
        OUT GT_LPORT memPorts[ ],
        OUT GT_U8       *memPortsLen
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - Logical port number to set

OUTPUTS:

memPorts - Array of logical ports
memPortsLen - Number of members in memPorts array

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

6.4.1.13 gvlnSetPortUserPriLsb

DESCRIPTION

This routine sets the user priority (VPT) LSB bit, to be added to the user priority on the egress.

SYNOPSIS

```
GT_STATUS gvlnSetPortUserPriLsb
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     userPriLsb
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number to set
userPriLsb - GT_TRUE for 1, GT_FALSE for 0
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

6.4.1.14 gvlnGetPortUserPriLsb

DESCRIPTION

This routine gets the user priority (VPT) LSB bit.

SYNOPSIS

```
GT_STATUS gvlnGetPortUserPriLsb
(
        IN  GT_QD_DEV   *dev
        IN  GT_LPORT     port,
        OUT GT_BOOL     *userPriLsb
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number to set
OUTPUTS:
userPriLsb - GT_TRUE for 1, GT_FALSE for 0

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

6.4.1.15 gvlnSetPortVid

DESCRIPTION

This routine sets the port default vlan id.

SYNOPSIS

```
GT_STATUS gvlnSetPortVid
(
        IN GT_QD_DEV *dev
        IN GT_LPORT   port,
        IN GT_U16     vid
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver
port - Logical port number to set
vid - The port vlan id
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

6.4.1.16 gvlnGetPortVid

DESCRIPTION

This routine gets the port default vlan id.

SYNOPSIS

```
GT_STATUS gvlnGetPortVid
(
        IN  GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        OUT GT_U16     *vid
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number to set

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

OUTPUTS:
vid - The port vlan id

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_BAD_PARAM - On bad parameters

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.4.2 VLAN Translation Unit (802.1Q)

The API for VTU (VLAN Translation Unit for 802.1Q) is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtBrgVtu.c.

6.4.2.1 gvtuGetEntryCount

DESCRIPTION

Gets the current number of entries in the VTU table.

SYNOPSIS

```
GT_STATUS gvtuGetEntryCount
(
        IN GT_QD_DEV *dev
        OUT GT_U32   *numEntries
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS: numEntries - number of VTU entries.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NO_SUCH - vlan does not exist
GT_NOT_SUPPORTED - If not supported by the device

6.4.2.2 gvtuGetEntryFirst

DESCRIPTION

Gets first lexicographic entry from the VTU.

SYNOPSIS

```
GT_STATUS gvtuGetEntryFirst
(
        IN GT_QD_DEV        *dev
        OUT GT_VTU_ENTRY    *vtuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS: vtuEntry - match VTU entry.

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_NO_SUCH - table is empty.
GT_NOT_SUPPORTED - If not supported by the device

6.4.2.3 gvtuGetEntryNext

DESCRIPTION

Gets next lexicographic VTU entry from the specified VID.

SYNOPSIS

```
GT_STATUS gvtuGetEntryNext
(
        IN  GT_QD_DEV           *dev
        INOUT GT_VTU_ENTRY  *vtuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
vtuEntry - the VID to start the search.
OUTPUTS:
vtuEntry - match VTU entry.

RETURNS

GT_OK - On success
GT_FAIL - On error or entry does not exist.
GT_NO_SUCH - no more entries.
GT_NOT_SUPPORTED - If not supported by the device

6.4.2.4 gvtuFindVidEntry

DESCRIPTION

Find VTU entry for a specific VID, it will return the entry, if found, along with its associated data

SYNOPSIS

```
GT_STATUS gvtuGetEntryNext
(
        IN GT_QD_DEV            *dev
        INOUT GT_VTU_ENTRY  *vtuEntry
        OUT GT_BOOL             *found
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
vtuEntry - contains the VID to search for.
OUTPUTS:
found - GT_TRUE, if the appropriate entry exists.
vtuEntry - the entry parameters.

RETURNS

GT_OK - On success

Copyright © 2003 Marvell

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

GT_FAIL - On error or entry does not exist.
GT_NO_SUCH - no more entries.
GT_NOT_SUPPORTED - If not supported by the device

6.4.2.5 gvtuFlush

DESCRIPTION

This routine removes all entries from VTU Table.

SYNOPSIS

```
GT_STATUS gvtuFlush
(
        IN GT_QD_DEV  *dev
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.4.2.6 gvtuAddEntry

DESCRIPTION

Creates the new entry in VTU table based on user input.

SYNOPSIS

```
GT_STATUS gvtuAddEntry
(
        IN GT_QD_DEV      *dev
        IN GT_VTU_ENTRY *vtuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
vtuEntry - vtu entry to insert to the VTU.
OUTPUTS:
None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_OUT_OF_CPU_MEM - oaMalloc failed
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.4.2.7 gvtuDelEntry

DESCRIPTION

Deletes VTU entry specified by user.

SYNOPSIS

```
GT_STATUS gvtuDelEntry
(
        IN GT_QD_DEV      *dev
        IN GT_VTU_ENTRY *vtuEntry
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
vtuEntry - the VTU entry to be deleted

OUTPUTS:

None.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NO_SUCH - if specified address entry does not exist
GT_NOT_SUPPORTED - If not supported by the device

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.4.3 FDB/ATU (Filtering Database/Address Translation Unit)

The API for the Filtering database, also known as Address Translation Unit, is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtBrgFdb.c.

6.4.3.1 gfdbSetAtuSize

DESCRIPTION

This routine sets the Mac address table size.

SYNOPSIS

```
GT_STATUS gfdbSetAtuSize
(
        IN GT_QD_DEV *dev
        IN ATU_SIZE     size
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
size - Mac address table size
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.3.2 gfdbGetAgingTimeRange

DESCRIPTION

This routine gets the maximal and minimum age times that the hardware can support.

SYNOPSIS

```
GT_STATUS gfdbGetAgingTimeRange
(
        IN GT_QD_DEV *dev
        OUT GT_U32 *maxTimeout,
        OUT GT_U32 *minTimeout
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
maxTimeout - Maximum aging time in seconds
minTimeout - Minimum aging time in seconds

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.3.3 gfdbSetAgingTimeout

DESCRIPTION

This routine sets the time-out period in seconds for aging out dynamically learned forwarding information. The standard recommends 300 sec.

SYNOPSIS

```
GT_STATUS gfdbSetAgingTimeout
(
        IN GT_QD_DEV *dev
        IN GT_U32    timeout
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
timeout - Aging time in seconds
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.3.4 gfdbGetAtuDynamicCount

DESCRIPTION

Gets the current number of dynamic unicast entries in the Filtering Database.

SYNOPSIS

```
GT_STATUS gfdbGetAtuDynamicCount
(
        IN GT_QD_DEV *dev
        OUT GT_U32   *numDynEntries
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
numDynEntries - Number of dynamic entries

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NO_SUCH - VLAN does not exist

6.4.3.5 gfdbGetAtuEntryFirst

DESCRIPTION

This routine gets first lexicographic MAC address entry from the ATU.

SYNOPSIS

```
GT_STATUS gfdbGetAtuEntryFirst
(
        IN GT_QD_DEV        *dev
        OUT GT_ATU_ENTRY    *atuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
OUTPUTS:
atuEntry - Match Address translate unit entry

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NO_SUCH - Table is empty

6.4.3.6 gfdbGetAtuEntryNext

DESCRIPTION

This routine gets the next lexicographic MAC address from the specified Mac Address.

SYNOPSIS

```
GT_STATUS gfdbGetAtuEntryNext
(
        IN GT_QD_DEV        *dev
        INOUT GT_ATU_ENTRY  *atuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
atuEntry - The Mac Address to start the search
OUTPUTS:
atuEntry - Match Address translate unit entry

RETURNS

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_OK - On success
GT_FAIL - On error or entry does not exist
GT_NO_SUCH - No more entries.

6.4.3.7 gfdbFindAtuMacEntry

DESCRIPTION

This routine finds FDB entry for specific MAC address from the ATU.

SYNOPSIS

```
GT_STATUS gfdbFindAtuMacEntry
(
        IN GT_QD_DEV        *dev
        INOUT GT_ATU_ENTRY  *atuEntry,
        OUT GT_BOOL *found
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
atuEntry - The Mac address to search
OUTPUTS:
found - GT_TRUE, if the appropriate entry exists
atuEntry - The entry parameters

RETURNS

GT_OK - On success
GT_FAIL - On error or entry does not exist
GT_NO_SUCH - No more entries

6.4.3.8 gfdbFlush

DESCRIPTION

This routine flushes all or unblocked addresses from the MAC Address Table.

SYNOPSIS

```
GT_STATUS gfdbFlush
(
        IN GT_QD_DEV       *dev
        IN GT_FLUSH_CMD flushCmd
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
flushCmd - The flush operation type
OUTPUTS: None

RETURNS

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.4.3.9 gfdbFlushInDB

DESCRIPTION

This routine flushes all or unblocked addresses from the particular ATU Database (DBNum).
If multiple address databases are being used, this API can be used to flush entries in a particular DBNum database.

SYNOPSIS

```
GT_STATUS gfdbFlushInDB
(
        IN GT_QD_DEV      *dev
        IN GT_FLUSH_CMD flushCmd,
        IN GT_U8 DBNum
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
flushCmd - The flush operation type.
DBNum - ATU MAC Address Database Number
OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - if current device does not support this feature.

6.4.3.10 gfdbAddMacEntry

DESCRIPTION

This routine creates a new entry in the MAC address table.

SYNOPSIS

```
GT_STATUS gfdbAddMacEntry
(
        IN BT_QD_DEV      *dev
        IN GT_ATU_ENTRY *macEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
macEntry - Mac address entry to insert to the ATU
OUTPUTS: None

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_OUT_OF_CPU_MEM - OaMalloc failed
GT_NOT_SUPPORTED - If not supported by the device

6.4.3.11 gfdbDelMacEntry

DESCRIPTION

This routine deletes the MAC address entry.

SYNOPSIS

```
GT_STATUS gfdbDelMacEntry
(
        IN GT_QD_DEV        *dev
        IN GT_ETHERADDR  *macAddress
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
macAddress - MAC address
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NO_SUCH - If specified address entry does not exist
GT_NOT_SUPPORTED - If not supported by the device

6.4.3.12 gfdbDelAtuEntry

DESCRIPTION

This routine deletes the ATU entry.

SYNOPSIS

```
GT_STATUS gfdbDelAtuEntry
(
        IN GT_QD_DEV       *dev
        IN GT_ATU_ENTRY  *atuEntry
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
atuEntry - the ATU entry to be deleted.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

OUTPUTS:
None

RETURNS

GT_OK - On success
GT_FAIL - On error

COMMENTS

DBNum in atuEntry -
ATU MAC Address Database number. If multiple address databases are not being used, DBNum should be zero. If multiple address databases are being used, this value should be set to the desired address database number.

6.4.3.13 gfdbLearnEnable

DESCRIPTION

Enable/disable automatic learning of new source MAC addresses on port ingress.

SYNOPSIS

```
GT_STATUS gfdbLearnEnable
(
        IN GT_QD_DEV *dev
        IN GT_BOOL      en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
en - GT_TRUE for enable or GT_FALSE otherwise
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.3.14 gfdbMove

DESCRIPTION

This routine moves all or unblocked addresses in the MAC Address Table from a port to another.

SYNOPSIS

```
GT_STATUS gfdbMove
(
        IN GT_QD_DEV       *dev
        IN GT_MOVE_CMD moveCmd,
```

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

```
    IN GT_U32moveFrom,
    IN GT_U32moveTo
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
moveCmd - The move operation type
moveFrom - port where moving from
moveTo - port where moving to

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.4.3.15 gfdbMoveInDB

DESCRIPTION

This routine moves all or unblocked addresses in the particular ATU Database (DBNum) from a port to another. If multiple address databases are being used, this API can be used to move entries in a particular DBNumdatabase.

SYNOPSIS

```
GT_STATUS gfdbMoveInDB
(
    IN GT_QD_DEV       *dev
    IN GT_MOVE_CMD moveCmd,
    IN GT_U8DBNum,
    IN GT_U32moveFrom,
    IN GT_U32moveTo
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
moveCmd - The move operation type.
DBNum - ATU MAC Address Database Number
moveFrom - port where moving from
moveTo - port where moving to

OUTPUTS:

None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - if current device does not support this feature.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.4.4 Spanning Tree Protocol (STP)

The API used for the states of Spanning Tree Protocol is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtBrgStp.c.

6.4.4.1 gstpSetMode

DESCRIPTION

This routine enables or disables Spanning tree. When enabled, this function sets all ports to blocking state, and inserts the BPDU MAC into the ATU to be captured to CPU. On disable all ports is being modified to be in forwarding state.

SYNOPSIS

```
GT_STATUS gstpSetMode
(
IN GT_QD_DEV *dev
IN GT_BOOL    en
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
en - GT_TRUE for enable,
GT_FALSE for disable
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORTED - If not supported by the device

6.4.4.2 gstpSetPortState

DESCRIPTION

This routine sets the port state.

SYNOPSIS

```
GT_STATUS gstpSetPortState
(
        IN GT_QD_DEV *dev
        IN GT_LPORT port,
        IN GT_PORT_STP_STATE  state
);
```

ARGUMENTS

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
state - The port state to set
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.4.4.3 gstpGetPortState

DESCRIPTION

This routine returns the port state.

SYNOPSIS

```
GT_STATUS gstpGetPortState
(
        IN GT_QD_DEV                *dev
        IN  GT_LPORT                port,
        OUT GT_PORT_STP_STATE  *state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - The logical port number
OUTPUTS:
state - The current port state

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.5 PHY Port Library

Table 5: PHY Port Library H files

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes.h	MS API prototypes

6.5.1 PHY Control

The Marvell Semiconductor API for PHY control is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtPhyCtrl.c.

6.5.1.1 gprtPhyReset

DESCRIPTION

This routine performs PHY reset. After reset, Auto-Negotiation will be enabled.

SYNOPSIS

```
GT_STATUS gprtPhyReset
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - The logical port number

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.2 gprtSetPortLoopback

DESCRIPTION

Enable/Disable Internal Port Loopback. Enabling Loopback will disable the Auto-Negotiation and set the PHY mode to 10 Mbps half-duplex.
To test Loopback on a different mode (such as 100 Mbps full-duplex), the user may need to call gprtSetPortSpeed and gprtSetPortDuplexMode.

Disabling Loopback does not enable the Auto-Negotiation, so the user may need to call gprtPortAutoNegEnable in order to enable Auto-Negotiation.

SYNOPSIS

```
GT_STATUS gprtSetPortLoopback
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_BOOL     enable
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - Logical port number
- enable - If GT_TRUE, enable loopback
  If GT_FALSE, disable loopback mode

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.3 gprtSetPortSpeed

DESCRIPTION

Sets speed for a specific logical port. This function will keep the duplex mode and Loopback mode to the previous value, but disable others (such as Auto-Negotiation).

SYNOPSIS

```
GT_STATUS gprtSetPortSpeed
(
        IN GT_QD_DEV *dev
        IN GT_LPORT    port,
        IN GT_BOOL     speed
);
```

ARGUMENTS

INPUTS:
- dev - specifies device context returned by qdLoadDriver()
- port - Logical port number
- speed - Port speed, GT_TRUE=100 Mbps
  GT_FALSE=10 Mbps

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.5.1.4 gprtPortAutoNegEnable

DESCRIPTION

Enable/disable an Auto-Negotiation for duplex mode on specific logical port. When Auto-Negotiation is disabled, the PHY will be in 10 Mbps half-duplex mode.
Enabling Auto-Negotiation will set 100BASE-TX full-duplex, 100BASE-TX full-duplex, 100BASE-TX full-duplex, and 100BASE-TX full-duplex in Auto-Negotiation Advertisement register.

SYNOPSIS

```
GT_STATUS gprtPortAutoNegEnable
(
        IN GT_QD_DEV *dev
        IN GT_LPORT   port,
        IN GT_BOOL    state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
state - GT_TRUE for enable Auto-Negotiation for duplex mode, GT_FALSE otherwise
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.5 gprtPortPowerDown

DESCRIPTION

Enable/disable (power down) on specific port. When this function is called with normal operation re quest, the PHY will set to Auto-Negotiation mode.

SYNOPSIS

```
GT_STATUS gprtPortPowerDown
(
        IN GT_QD_DEV  *dev
        IN GT_LPORT    port,
        IN GT_BOOL     state
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
state - GT_TRUE: power down
GT_FALSE: normal operation
OUTPUTS: None

RETURNS

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

GT_OK - On success
GT_FAIL - On error

6.5.1.6 gprtPortRestartAutoNeg

DESCRIPTION

Restart Auto-Negotiation. If Auto-Negotiation is not enabled, this routine enables Auto-Negotiation of interface speed on a specific port. Loopback and Power Down will be disabled by this routine.

SYNOPSIS

```
GT_STATUS gprtPortRestartAutoNeg
(
        IN GT_QD_DEV *dev
        IN GT_LPORT   port
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.7 gprtSetPortDuplexMode

DESCRIPTION

Sets duplex mode for a specific logical port. This function will keep the speed and Loopback mode to the previous value, but disable others (such as Auto-Negotiation).

SYNOPSIS

```
GT_STATUS gprtSetPortDuplexMode
(
        IN GT_QD_DEV        *dev
        IN GT_LPORT         port,
        IN GT_PORT_DUPLEX dMode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
dMode - Dulpex mode
OUTPUTS: None

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.8 gprtSetPortAutoMode

DESCRIPTION

This routine sets up the port with given Auto Mode. The modes supported are as follows:

- Auto for both speed and duplex
- Auto for speed only and full-duplex
- Auto for speed only and half-duplex
- Auto for duplex only and 100 Mbps speed
- Auto for duplex only and 10 Mbps speed

SYNOPSIS

```
GT_STATUS gprtSetPortAutoMode
(
        IN GT_QD_DEV        *dev
        IN GT_LPORT         port,
        IN GT_PORT_DUPLEX dMode
);
```

ARGUMENTS

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
dMode - Dulpex mode
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.9 gprtSetPause

DESCRIPTION

This routine will set the pause bit in the Auto-Negotiation Advertisement Register and restart Auto-Negotiation.

SYNOPSIS

```
GT_STATUS gprtSetPause
(
        IN GT_QD_DEV      *dev
        IN GT_LPORT       port,
        IN GT_PORT_BOOL state
);
```

ARGUMENTS

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

INPUTS:
dev - specifies device context returned by qdLoadDriver()
port - Logical port number
state - set/reset Pause bit
OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

6.5.1.10 gprtSetPhyReg

DESCRIPTION

This routine writes Phy Registers.

SYNOPSIS

```
GT_STATUS gprtSetPhyReg
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN  GT_U32       regAddr,
        IN  GT_U16       data
);
```

ARGUMENTS

port - logical port number.
regAddr - The register's address.
data - The read register's data.

RETURNS

GT_OK - on success
GT_FAIL - on error

6.5.1.11 gprtGetPhyReg

DESCRIPTION

This routine reads Phy Registers.

SYNOPSIS

```
GT_STATUS gprtGetPhyReg
(
        IN  GT_QD_DEV    *dev,
```

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

```
        IN  GT_LPORT    port,
        IN  GT_U32      regAddr,
        OUT GT_U16      *data
);
```

ARGUMENTS

port - logical port number.
regAddr - The register's address.
data - The read register's data.

RETURNS

GT_OK - on success
GT_FAIL - on error

6.5.1.12 gprtSet1000TMasterMode

DESCRIPTION

This routine sets the Phy's 1000BaseT Master/Slave mode.

SYNOPSIS

```
GT_STATUS gprtSet1000TMasterMode
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        IN GT_1000T_MASTER_SLAVE*mode
);
```

ARGUMENTS

port - logical port number.
mode - GT_1000T_MASTER_SLAVE structure
    autoConfig - GT_TRUE for auto, GT_FALSE for manual setup.
    masterPrefer - GT_TRUE if Master configuration is preferred.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.5.1.13 gprtGet1000TMasterMode

DESCRIPTION

This routine gets the Phy's 1000BaseT Master/Slave mode.

SYNOPSIS

```
GT_STATUS gprtGet1000TMasterMode
(
        IN  GT_QD_DEV    *dev,
        IN  GT_LPORT     port,
        OUT GT_1000T_MASTER_SLAVE*mode
);
```

ARGUMENTS

port - logical port number.
mode - GT_1000T_MASTER_SLAVE structure
    autoConfig - GT_TRUE for auto, GT_FALSE for manual setup.
    masterPrefer - GT_TRUE if Master configuration is preferred.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.5.2 Virtual Cable Tester™ (VCT)

The API for Marvell Virtual Cable Tester™ is defined in msApiDefs.h and msApiPrototypes.h. It is implemented in gtVct.c.

6.5.2.1 gvctGetCableStatus

DESCRIPTION

This routine perform the virtual cable test for the requested port, and returns the status per MDI pair.

SYNOPSIS

```
GT_STATUS gvctGetCableDiag
(
        IN GT_QD_DEV            *dev
        IN  GT_LPORT            port,
        OUT GT_CABLE_STATUS *cableStatus
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - logical port number.

OUTPUTS:

cableStatus - the port copper cable status.
cableLen - the port copper cable length.

RETURNS

GT_OK - on success
GT_FAIL - on error

6.5.2.2 gvctGet1000BTExtendedStatus

DESCRIPTION

This routine retrieves extended cable status, such as Pair Poloarity, Pair Swap, and Pair Skew. Note that this routine will be success only if 1000Base-T Link is up.

SYNOPSIS

```
GT_STATUS gvctGet1000BTExtendedStatus
(
        IN GT_QD_DEV            *dev
        IN  GT_LPORT            port,
        OUT GT_1000BT_EXTENDED_STATUS *extendedStatus
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - logical port number.

OUTPUTS:

extendedStatus - the extended cable status.

RETURNS

GT_OK - on success
GT_FAIL - on error
GT_NOT_SUPPORTED - if current device does not support this feature.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.6 Event/Interrupt Library

Table 6: Event/Interrupt Library H files

File Name	Description
msApiDefs.h	MS API structures and definitions
MsApiPrototypes.h	MS API prototypes

The API for the SOHO Switch Products device Interrupt is defined in msApiDefs.h and msApiPrototypes.h and implemented in gtEvent.c and gtPhyInt.c. If the target hardware supports interrupt, BSP for the target should register its own Interrupt Service Routing to the RTOS, and inside the ISR routine these APIs can be called to get the interrupt cause and de-assert INTn. For more information, refer to the Sample\VxWorks\Interrupt\gtInt.c file.

6.6.1 System Interrupt

6.6.1.1 eventSetActive

DESCRIPTION

This routine enables/disables the receive of an hardware driven event.

SYNOPSIS

```
GT_STATUS eventSetActive
(
        IN GT_QD_DEV  *dev
        IN GT_U32     eventType
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
eventType - the event type. any combination of the following:
GT_STATS_DONE, GT_VTU_PROB, GT_VTU_DONE, GT_ATU_FULL,
GT_ATU_DONE, GT_PHY_INTERRUPT, and GT_EE_INTERRUPT.

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.6.1.2 eventGetIntStatus

DESCRIPTION

This routine reads the hardware driven event status.

SYNOPSIS

```
GT_STATUS eventGetIntStatus
(
        IN GT_QD_DEV *dev
        OUT  GT_U16    *intCause
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

intCause - It provides the source of interrupt of the following:
GT_STATS_DONE, GT_VTU_PROB, GT_VTU_DONE, GT_ATU_FULL,
GT_ATU_DONE, GT_PHY_INTERRUPT, and GT_EE_INTERRUPT

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.6.2 PHY Interrupt

6.6.2.1 gprtPhyIntEnable

DESCRIPTION

Enable/Disable on PHY Interrupt. This register determines whether the INT# pin is asserted when an interrupt event occurs. When an interrupt occurs, the corresponding bit is set and remains set until Register 19 is read via the SMI. When interrupt enable bits are not set in Register 18, interrupt status bits in Register 19 are still set when the corresponding interrupt events occur; however, the INT# is not asserted.

SYNOPSIS

```
GT_STATUS gprtGetPhyIntEnable
(
        IN GT_QD_DEV  *dev
        IN  GT_LPORT    port,
        IN  GT_U16       intType
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port - Logical port number
intType - The type of interrupt to enable/disable. Any of the following combinations may be disabled:

- GT_SPEED_CHANGED
- GT_DUPLEX_CHANGED
- GT_PAGE_RECEIVED
- GT_AUTO_NEG_COMPLETED
- GT_LINK_STATUS_CHANGED
- GT_SYMBOL ERROR
- GT_FALSE_CARRIER
- GT_FIFO_FLOW
- GT_CROSSOVER_CHANGED
- GT_POLARITY-CHANGED
- GT_JABBER

To disable PHY Interrupt, use value 0.

OUTPUTS: None

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.6.2.2 gprtGetPhyIntStatus

DESCRIPTION

This register determines which interrupt occurred. Calling this function will de-assert the INT# pin.

SYNOPSIS

```
GT_STATUS gprtGetPhyIntStatus
(
        IN GT_QD_DEV *dev
        IN OUT GT_U8   port,
        OUT GT_U16     *intCause
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()
port -Logical port number

OUTPUTS:

intCause - The type of interrupt from which to get the status. Any of the following combinations may be disabled:

- GT_SPEED_CHANGED
- GT_DUPLEX_CHANGED
- GT_PAGE_RECEIVED
- GT_AUTO_NEG_COMPLETED
- GT_LINK_STATUS_CHANGED
- GT_SYMBOL ERROR
- GT_FALSE_CARRIER
- GT_FIFO_FLOW
- GT_CROSSOVER_CHANGED
- GT_POLARITY-CHANGED
- GT_JABBER

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.6.2.3 gprtGetPhyIntPortSummary

DESCRIPTION

This register lists the ports that have active interrupts. It provides a quick way to isolate the interrupt so that the MAC or switch does not have to poll the interrupt status register (Register 19) for all ports. Reading this register does not de-assert the INT# pin.

SYNOPSIS

```
GT_STATUS gprtGetPhyIntPortSummary
(
        IN GT_QD_DEV *dev
        OUT GT_U16     *intPortMask
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

intPortMask - Bit Mask with the bits set for the corresponding PHYs with active interrupt (e.g., the bit number 0 and 2 are set when port number 0 and 2 have active interrupt.

RETURNS

GT_OK - On success
GT_FAIL - On error

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao
MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

6.6.3 VTU Interrupt

6.6.3.1 gvtuGetIntStatus

DESCRIPTION

Get VTU Violation Interrupt Status

SYNOPSIS

```
GT_STATUS gvtuGetIntStatus
(
        IN GT_QD_DEV            *dev
        OUT GT_VTU_INT_STATUS *vtuIntStatus
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

vtuIntStatus - interrupt cause, source portID, and vid.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

6.6.4 ATU Interrupt

6.6.4.1 gatuGetIntStatus

DESCRIPTION

Get ATU Violation Interrupt Status

SYNOPSIS

```
GT_STATUS gatuGetIntStatus
(
        IN GT_QD_DEV            *dev
        OUT GT_ATU_INT_STATUS *atuIntStatus
);
```

ARGUMENTS

INPUTS:

dev - specifies device context returned by qdLoadDriver()

OUTPUTS:

atuIntStatus - interrupt cause, source portID, DBNum, and MAC Address.

RETURNS

GT_OK - On success
GT_FAIL - On error
GT_NOT_SUPPORT - if current device does not support this feature.

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

Appendix A. Supported API List

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
System Control									
	gsysSwReset	√	√	√	√	√	√	√	√
	gsysSetDiscardEcessive	√	√	√	√	√	√	√	√
	gsysGetDiscardEcessive	√	√	√	√	√	√	√	√
	gsysSetSchedulingMode	√	√	√	√	√	√	√	√
	gsysGetSchedulingMode	√	√	√	√	√	√	√	√
	gsysSetMaxFrameSize	√	√	√	√	√	√	√	√
	gsysGetMaxFrameSize	√	√	√	√	√	√	√	√
	gsysReLoad	√	√	√	√	√	√	√	√
	gsysSetWatchDog	√	√	√	√	√	√	√	√
	gsysGetWatchDog	√	√	√	√	√	√	√	√
	gsysSetDuplexPauseMac	√	√	√	√	√	√	√	√
	gsysGetDuplexPauseMac	√	√	√	√	√	√	√	√
	gsysSetPerPortDuplexPauseMac	√	√	√	√	√	√	√	√
	gsysGetPerPortDuplexPauseMac	√	√	√	√	√	√	√	√
	gsysReadMiiReg	√	√	√	√	√	√	√	√
	gsysWriteMiiReg	√	√	√	√	√	√	√	√
	gsysSetPPUEn								√
	gsysGetPPUEn								√
	gsysSetCascadePort								√
	gsysGetCascadePort								√
	gsysSetDeviceNumber								√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
gsysGetDeviceNumber								√
System Status								
gsysGetSW_Mode	√			√	√	√	√	√
gsysGetInitReady	√			√	√	√	√	√
gsysGetPPUState								√
System Config								
qdLoadDriver	√	√	√	√	√	√	√	√
qdUnloadDriver	√	√	√	√	√	√	√	√
sysEnable	√	√	√	√	√	√	√	√
Port Control								
gprtSetForceFc	√	√	√	√	√	√	√	√
gprtGetForceFc	√	√	√	√	√	√	√	√
gprtSetTrailerMode	√		√	√	√	√	√	
gprtGetTrailerMode	√		√	√	√	√	√	
gprtSetIngressMode	√	√	√	√	√	√	√	
gprtGetIngressMode	√	√	√	√	√	√	√	
gprtSetMcRateLimit	√	√	√					
gprtGetMcRateLimit	√	√	√					
gprtSetIGMPSnoop	√				√		√	√
gprtGetIGMPSnoop	√				√		√	√
gprtSetHeaderMode				√	√	√	√	√
gprtGetHeaderMode				√	√	√	√	√
gprtSetProtectedMode							√	√
gprtGetProtectedMode							√	√
gprtSetForwardUnknown							√	√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gprtGetForwardUnknown							√	√
	gprtSetDropOnLock								√
	gprtGetDropOnLock								√
	gprtSetDoubleTag								√
	gprtGetDoubleTag								√
	gprtSetInterswitchPort								√
	gprtGetInterswitchPort								√
	gprtSetLearnDisable								√
	gprtGetLearnDisable								√
	gprtSetIgnoreFCS								√
	gprtGetIgnoreFCS								√
	gprtSetVTUPriOverride								√
	gprtGetVTUPriOverride								√
	gprtSetSAPriOverride								√
	gprtGetSAPriOverride								√
	gprtSetDAPriOverride								√
	gprtGetDAPriOverride								√
	gprtSetCPUPort								√
	gprtGetCPUPort								√
	gprtSetLockedPort								√
	gprtGetLockedPort								√
	gprtSetIgnoreWrongData								√
	gprtGetIgnoreWrongData								√
Port Status									
	gprtGetPartnerLinkPause	√	√	√	√	√	√	√	

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gprtGetSelfLinkPause	√	√	√	√	√	√	√	√
	gprtGetResolve	√	√	√	√	√	√	√	
	gprtGetLinkState	√	√	√	√	√	√	√	√
	gprtGetPortMode	√	√	√	√	√	√	√	
	gprtGetPhyMode	√	√	√	√	√	√	√	
	gprtGetDuplex	√	√	√	√	√	√	√	√
	gprtSetDuplex	√	√		√	√	√	√	
	gprtGetSpeed	√	√	√	√	√	√	√	
	gprtGetPauseEn								√
	gprtGetHdFlow								√
	gprtGetPHYDetect								√
	gprtSetPHYDetect								√
	gprtGetSpeedMode								√
	gprtGetHighErrorRate								√
	gprtGetTxPaused								√
	gprtGetFlowCtrl								√
	gprtGetC_Duplex								√
	gprtGetC_Mode								√
RMON									
	gstatsFlushAll	√				√		√	√
	gstatsFlushPort	√				√		√	√
	gstatsGetPortCounter	√				√		√	
	gstatsGetPortAllCounters	√				√		√	
	gstatsGetPortCounter2								√
	gstatsGetPortAllCounters2								√
	gstatsGetHistogramMode								√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gstatsSetHistogramMode								√
Switch Statistics									
	gprtSetCtrMode	√	√	√	√	√	√	√	√
	gprtClearAllCtr	√	√	√	√	√	√	√	√
	gprtGetPortCtr	√	√	√	√	√	√	√	√
PCS Control									
	gpcsGetCommaDet								√
	gpcsGetSyncOK								√
	gpcsGetSyncFail								√
	gpcsGetAnBypassed								√
	gpcsGetAnBypassMode								√
	gpcsSetAnBypassMode								√
	gpcsGetPCSAnEn								√
	gpcsSetPCSAnEn								√
	gpcsSetRestartPCSAn								√
	gpcsGetPCSAnDone								√
	gpcsSetLinkValue								√
	gpcsGetLinkValue								√
	gpcsSetForcedLink								√
	gpcsGetForcedLink								√
	gpcsSetDpxValue								√
	gpcsGetDpxValue								√
	gpcsSetForcedDpx								√
	gpcsGetForcedDpx								√
	gpcsSetForceSpeed								√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MSAPI		88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gpcsGetForceSpeed								√
QoS Map									
	gqosSetPrioMapRule	√		√		√	√	√	√
	gqosGetPrioMapRule	√		√		√	√	√	√
	gqosIpPrioMapEn	√		√		√	√	√	√
	gqosGetIpPrioMapEn	√		√		√	√	√	√
	gqosUserPrioMapEn	√		√		√	√	√	√
	gqosGetUserPrioMapEn	√		√		√	√	√	√
	gcosGetPortDefaultTc	√		√		√	√	√	√
	gcosSetPortDefaultTc	√		√		√	√	√	√
	gcosGetUserPrio2Tc	√		√		√	√	√	√
	gcosSetUserPrio2Tc	√		√		√	√	√	√
	gcosGetDscp2Tc	√		√		√	√	√	√
	gcosSetDscp2Tc	√		√		√	√	√	√
	gqosGetTagRemap								√
	gqosSetTagRemap								√
VLAN									
	gprtSetEgressMode	√	√	√	√	√	√	√	√
	gprtGetEgressMode	√	√	√	√	√	√	√	√
	gprtSetVlanTunnel	√	√	√	√	√	√	√	√
	gprtGetVlanTunnel	√	√	√	√	√	√	√	√
	gvlnSetPortVlanPorts	√	√	√	√	√	√	√	√
	gvlnGetPortVlanPorts	√	√	√	√	√	√	√	√
	gvlnSetPortUserPriLsb	√	√	√	√	√	√	√	√
	gvlnGetPortUserPriLsb	√	√	√	√	√	√	√	√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gvlnSetPortVid	√	√	√	√	√	√	√	√
	gvlnGetPortVid	√	√	√	√	√	√	√	√
	gvlnSetPortVlanDBNum	√		√	√	√	√	√	√
	gvlnGetPortVlanDBNum	√		√	√	√	√	√	√
	gvlnSetPortVlanDot1qMode	√				√		√	√
	gvlnGetPortVlanDot1qMode	√				√		√	√
	gvlnSetPortVlanForceDefaultVID	√				√		√	√
	gvlnGetPortVlanForceDefaultVID	√				√		√	√
VTU									
	gvtuGetEntryCount	√				√		√	√
	gvtuGetEntryFirst	√				√		√	√
	gvtuGetEntryNext	√				√		√	√
	gvtuFindVidEntry	√				√		√	√
	gvtuFlush	√				√		√	√
	gvtuAddEntry	√				√		√	√
	gvtuDelEntry	√				√		√	√
FDB									
	gfdbSetAtuSize	√	√	√	√	√	√	√	
	gfdbGetAgingTimeRange	√	√	√	√	√	√	√	√
	gfdbSetAgingTimeout	√	√	√	√	√	√	√	√
	gfdbGetAtuDynamicCount	√	√	√	√	√	√	√	√
	gfdbGetAtuEntryFirst	√	√	√	√	√	√	√	√
	gfdbGetAtuEntryNext	√	√	√	√	√	√	√	√
	gfdbFindAtuMacEntry	√	√	√	√	√	√	√	√
	gfdbFlush	√		√	√	√	√	√	√
	gfdbFlushInDB	√			√	√		√	√

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
	gfdbAddMacEntry	√		√	√	√	√	√	√
	gfdbDelMacEntry	√		√	√	√	√	√	√
	gfdbDelAtuEntry	√		√	√	√	√	√	√
	gfdbLearnEnable	√	√	√	√	√	√	√	√
	gfdbMove								√
	gfdbMoveInDB								√
STP									
	gstpSetMode	√		√	√	√	√	√	√
	gstpSetPortState	√	√	√	√	√	√	√	√
	gstpGetPortState	√	√	√	√	√	√	√	√
PHY Control									
	gprtPhyReset	√	√	√	√	√	√	√	√
	gprtSetPortLoopback	√	√	√	√	√	√	√	√
	gprtSetPortSpeed	√	√	√	√	√	√	√	√
	gprtPortAutoNegEnable	√	√	√	√	√	√	√	√
	gprtPortPowerDown	√	√	√	√	√	√	√	√
	gprtPortRestartAutoNeg	√	√	√	√	√	√	√	√
	gprtSetPortDuple Mode	√	√	√	√	√	√	√	√
	gprtSetPortAutoMode	√	√	√	√	√	√	√	√
	gprtSetPause	√	√	√	√	√	√	√	√
	gprtSetPhyReg	√	√	√	√	√	√	√	√
	gprtGetPhyReg	√	√	√	√	√	√	√	√
	gprtSet1000TMasterMode								√
	gprtGet1000TMasterMode								√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
Rate Control								
grcSetLimitMode					√	√	√	√
grcGetLimitMode					√	√	√	√
grcSetPri3Rate					√	√	√	√
grcGetPri3Rate					√	√	√	√
grcSetPri2Rate					√	√	√	√
grcGetPri2Rate					√	√	√	√
grcSetPri1Rate					√	√	√	√
grcGetPri1Rate					√	√	√	√
grcSetPri0Rate					√	√	√	√
grcGetPri0Rate					√	√	√	√
grcSetBytesCount					√	√	√	√
grcGetBytesCount					√	√	√	√
grcSetEgressRate					√	√	√	√
grcGetEgressRate					√	√	√	√
PAV								
gpavSetPAV				√	√		√	√
gpavGetPAV				√	√		√	√
gpavSetIngressMonitor				√	√		√	√
gpavGetIngressMonitor				√	√		√	√
Virtual Cable Test								
gvctGetCableDiag				√	√	√	√	√
gvctGet1000BTExtendedStatus								√
Version								
gtVersion	√	√	√	√	√	√	√	√

	MSAPI	88E6021	88E6051	88E6052	88E6060	88E6063/88E6318	88E6218	88E6083	88E6181/88E6183
System Event									
	eventSetActive	√	√	√	√	√	√	√	√
	eventGetIntStatus	√	√	√	√	√	√	√	√
PHY Interrupt									
	gprtPhyIntEnable	√	√	√	√	√	√	√	
	gprtGetPhyIntStatus	√	√	√	√	√	√	√	
	gprtGetPhyIntPortSummary	√	√	√	√	√	√	√	
VTU Interrupt									
	gvtuGetIntStatus	√				√		√	√
ATU Interrupt									
	gatuGetIntStatus								√

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925
86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

MARVELL®

MOVING FORWARD FASTER®

Marvell Semiconductor, Inc.

700 First Avenue
Sunnyvale, CA 94089

Phone 408.222.2500
Fax 408.752.9028

www.marvell.com

US and Worldwide Offices

Marvell Semiconductor, Inc.
700 First Avenue
Sunnyvale, CA 94089
Tel: 1.408.222.2500
Fax: 1.408.752.9028

Marvell Asia Pte, Ltd.
151 Lorong Chuan, #02-05
New Tech Park
Singapore 556741
Tel: 65.6756.1600
Fax: 65.6756.7600

Marvell Japan K.K.
Shinjuku Center Bldg. 50F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0650
Tel: 81.(0).3.5324.0355
Fax: 81.(0).3.5324.0354

Marvell Semiconductor Israel, Ltd.
Moshav Manof
D.N. Misgav 20184
Israel
Tel: 972.4.999.9555
Fax: 972.4.999.9574

Worldwide Sales Offices

Western US Sales Office
Marvell
700 First Avenue
Sunnyvale, CA 94089
Tel: 1.408.222.2500
Fax: 1.408.752.9028
Sales Fax: 1.408.752.9029

Central US Sales Office
Marvell
11709 Boulder Lane, Ste. #220
Austin, TX 78726
Tel: 1.512.336.1551
Fax: 1.512.336.1552

Eastern US/Canada Sales Office
Marvell
Knox Trail Office Bldg.
2352 Main Street
Concord, MA 01742
Tel: 1.978.461.0563
Tel: 1.978.461.1406
Fax: 1.978.461.1405

Europe Sales Office
Marvell
3 Clifton Court
Corner Hall
Hemel Hempstead
Hertfordshire, HP3 9XY
United Kingdom
Tel: 44.(0).1442.211668
Fax: 44.(0).1442.211543

Marvell
Fagerstagatan 4
163 08 Spanga
Stockholm, Sweden
Tel: 46.16.146348
Fax: 46.16.482425

Marvell
5 Rue Poincare
56400 Le Bono
France
Tel: 33.297.579697
Fax: 33.297.578933

Israel Sales Office
Marvell
Ofek Center Bldg. 2, Floor 2
Northern Industrial Zone
LOD 71293
Israel
Tel: 972.8.924.7555
Fax: 972.8.924.7554

China Sales Office
Marvell
5J, 1800 Zhong Shan West Road
Shanghai, China 200233
Tel: 86.21.6440.1350
Fax: 86.21.6440.0799

Japan Sales Office
Marvell
Helios Kannai Bldg. 12F
3-21-2 Motohama-cho, Naka-ku
Yokohama, Kanagawa
Japan 231-0004
Tel: 81.45.222.8811
Fax: 81.45.222.8812

Taiwan Sales Office
Marvell
2Fl., No. 1, Alley 20, Lane 407
Ti-Ding Blvd., Nei Fu District
Taipei, Taiwan 114, R. O. C
Tel: (886-2).7720.5700
FAX: (886-2).7720.5707

For more information, visit our website at: www.marvell.com

Copyright © 2003 Marvell. All rights reserved. Marvell, the Marvell logo, Moving Forward Faster, Alaska, and GalNet are registered trademarks of Marvell. Discovery, Fastwriter, GalTis, Horizon, Libertas, Link Street, NetGX, PHY Advantage, Prestera, Raise The Technology Bar, UniMAC, Virtual Cable Tester, and Yukon are trademarks of Marvell. All other trademarks are the property of their respective owners.

MARVELL CONFIDENTIAL - UNAUTHORIZED DISTRIBUTION OR USE STRICTLY PROHIBITED

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao * UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925

86mazc0oog79myzxn3ia3gk7ix65-jbddy6ob * Senao MARVELL CONFIDENTIAL, UNDER NDA# 12156925