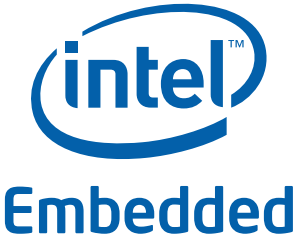

Intel® Platform Controller Hub EG20T

Packet HUB Driver for Windows* Programmer's Guide

February 2011

Order Number: 324265-002US

Contents

Tables

Revision History

Date	Revision	Description
February 2011	002	Updated Section 2.0, “Operating System (OS) Support” on page 6 Added Section 5.1.1, “Using Packet HUB Interface Exposed by the Driver” on page 13
September 2010	001	Initial release

1.0 Introduction

This document provides the programming details of the Packet HUB driver for Windows*. This includes the information about the interfaces exposed by the driver and how to use those interfaces to drive the Packet HUB hardware.

2.0 Operating System (OS) Support

The PACKET HUB driver is supported by the following operating systems:

No	OS	Notes
1	Microsoft Windows XP*	Service Pack 3
2	Windows Embedded Standard*	2009
3	Windows Embedded POSReady*	2009
4	Microsoft Windows 7*	
5	Windows Embedded Standard7	

3.0 Dependencies

This driver is only dependent upon appropriate OS driver installation. Also, this driver is not dependent upon any other software delivered.

4.0 Packet HUB Driver API Details

This section provides information about the interfaces exposed by the Packet HUB driver. In the current implementation of the driver, the interfaces are exposed through IOCTLs that can be called from the application (user mode) using the Win32 API DeviceIoControl (refer to the MSDN documentation for more details on this API). The following sections provide information about the IOCTLs and how to use them to configure the Packet HUB hardware to work properly.

4.1 Features

The Packet HUB driver supports:

- Reading and writing IOH Packet HUB hardware registers
- Reading and writing Serial EEPROM via Packet HUB Serial ROM interface

4.2 Interface Details

Table 1 lists IOCTLs supported by the driver.

Table 1. Supported IOCTLs

No	IOCTL	Remarks
1	IOCTL_PCIEQOS_READ_REG	Read IOH Packet HUB hardware registers
2	IOCTL_PCIEQOS_WRITE_REG	Write IOH Packet HUB hardware registers
3	IOCTL_PCIEQOS_READ_MODIFY_WRITE_REG	Read, modify and write IOH Packet HUB hardware registers
4	IOCTL_PCIEQOS_READ_OROM	Read Option ROM data via Packet HUB Serial ROM interface
5	IOCTL_PCIEQOS_WRITE_OROM	Write Option ROM data via Packet HUB Serial ROM interface
6	IOCTL_PCIEQOS_READ_MAC_ADDR	Read MAC address via Packet HUB Serial ROM interface
7	IOCTL_PCIEQOS_WRITE_ MAC_ADDR	Write MAC address via Packet HUB Serial ROM interface

4.3 IOCTL Usage Details

The following files contain the details of the IOCTLs and data structures used.

- ioh_pcieqos_ioctls.h – contains IOCTL definitions
- ioh_pcieqos_common.h – data structures and other variables used by the IOCTLs

Refer to the “Register Address Map | Memory-Mapped I/O Registers | Device Control Registers” section in the “Packet Hub” chapter of the *Intel® Platform Controller Hub EG20T IOH External Design Specification* for IOCTL_PCIEQOS_READ_REG and IOCTL_PCIEQOS_WRITE_REG offset address.

4.3.1 IOCTL_PCIEQOS_READ_REG

This IOCTL is called to read the Packet HUB register. The prerequisite for this is that the device must be installed and opened using the Win32 API CreateFile.

```
ioh_pcieqos_reqt pcieqos_reqt;

unsigned long dataout;            /* read data */
```

```
pcieqos_reqt.addr_offset = 0xXX;  /* target register address offset[1] */

pcieqos_reqt.data = 0;

pcieqos_reqt.mask = 0;

DeviceIoControl(hHandle,

                IOCTL_PCIEQOS_READ_REG,

                &pcieqos_reqt,

                sizeof(pcieqos_reqt),

                &dataout,

                sizeof(dataout),

                &ret_len,

                NULL);
```

4.3.2 IOCTL_PCIEQOS_WRITE_REG

This writes data into the Packet HUB register.

```
ioh_pcieqos_reqt pcieqos_reqt;

pcieqos_reqt.addr_offset = 0xXX;  /* target register address offset */

pcieqos_reqt.data = 0xXX;         /* write data */

pcieqos_reqt.mask = 0;

DeviceIoControl(hHandle,

                IOCTL_PCIEQOS_WRITE_REG,

                &pcieqos_reqt,

                sizeof(pcieqos_reqt),

                NULL,

                0,

                &ret_len,

                NULL);
```

4.3.3 IOCTL_PCIEQOS_READ_MODIFY_WRITE_REG

This reads data from the Packet HUB register, modifies it and writes it back to the Packet HUB register.

```
ioh_pcieqos_reqt pcieqos_reqt;

pcieqos_reqt.addr_offset = 0xXX;  /* target register address offset */

pcieqos_reqt.data = 0xXX;         /* write data */
```

1. Refer to the “Register Address Map | Memory-Mapped I/O Registers | Device Control Registers” section in the “Packet Hub” chapter of the *Intel® Platform Controller Hub EG20T External Design Specification*


```
pcieqos_reqt.mask = 0xXX;          /* mask value */
DeviceIoControl(hHandle,
                IOCTL_PCIEQOS_READ_MODIFY_WRITE_REG,
                &pcieqos_reqt,
                sizeof(pcieqos_reqt),
                NULL,
                0,
                &ret_len,
                NULL);
```

4.3.4 IOCTL_PCIEQOS_READ_OROM

This reads Option ROM data via the Packet HUB Serial ROM interface.

```
ioh_pcieqos_reqt pcieqos_reqt;
unsigned long dataout;             /* read data */
pcieqos_reqt.addr_offset = 0xXX;  /* target ROM area address offset */
pcieqos_reqt.data = 0x00;
pcieqos_reqt.mask = 0;
DeviceIoControl(hHandle,
                IOCTL_PCIEQOS_READ_OROM,
                &pcieqos_reqt,
                sizeof(pcieqos_reqt),
                &dataout,
                sizeof(dataout),
                &ret_len,
                NULL);
```

4.3.5 IOCTL_PCIEQOS_WRITE_OROM

This writes Option ROM data via the Packet HUB Serial ROM interface.

```
ioh_pcieqos_reqt pcieqos_reqt;
pcieqos_reqt.addr_offset = 0xXX;  /* target ROM area address offset */
pcieqos_reqt.data = 0xXX;          /* write data */
pcieqos_reqt.mask = 0;
DeviceIoControl(hHandle,
                IOCTL_PCIEQOS_WRITE_OROM,
                &pcieqos_reqt,
```

```
                sizeof(pcieqos_reqt),

                NULL,0,

                &ret_len,

                NULL);
```

4.3.6 IOCTL_PCIEQOS_READ_MAC_ADDR

This reads the MAC address via the Packet HUB Serial ROM interface.

```
unsigned char arMacAddress[6] = { 0 } ;   /* read MAC address */

pcieqos_reqt.addr_offset = 0x0;           /*Offset set as 0. Reads the entire MAC
address*/

pcieqos_reqt.data = 0x00;

pcieqos_reqt.mask = 0;

DeviceIoControl(hHandle,

                IOCTL_PCIEQOS_READ_MAC_ADDR,

                &pcieqos_reqt,

                sizeof(pcieqos_reqt),

                &arMacAddress,

                sizeof(arMacAddress),

                &ret_len,

                NULL);
```

4.3.7 IOCTL_PCIEQOS_WRITE_MAC_ADDR

This writes the MAC address via the Packet HUB Serial ROM interface.

```
ioh_pcieqos_reqt pcieqos_reqt;

pcieqos_reqt.addr_offset = 0xXX;  /* target MAC address offset (0-5) */

pcieqos_reqt.data = 0xXX;         /* write MAC address */

pcieqos_reqt.mask = 0;

DeviceIoControl(hHandle,

                IOCTL_PCIEQOS_WRITE_ MAC_ADDR,

                &pcieqos_reqt,

                sizeof(pcieqos_reqt),

                NULL,

                0,

                &ret_len,

                NULL);
```

4.4 Structures

This section provides the details on the structures used by interfaces exposed by the Packet HUB driver. All the structures used by the interfaces are defined in ioh_pcieqos_common.h.

4.4.1 ioh_pcieqos_reqt

This structure is used for preserving information related to the Packet HUB request.

Table 2. ioh_pcieqos_reqt structure

Name	Description
ULONG addr_offset	Specifies the register address offset
ULONG data	Specifies the write data
ULONG mask	Specifies the mask

4.5 Error Handling

Since the IOCTL command is implemented using Windows* API, the return value of the call is dependent on and defined by the OS. On Windows*, the return value will be a non-zero value. If the error is detected within or outside the driver, an appropriate system defined value is returned by the driver.

4.6 Inter-IOCTL dependencies

There are no inter-IOCTL dependencies. Once the driver has been loaded successfully, the IOCTLs stated above can be used in any order.

5.0 Programming Guide

This section describes the basic procedure for using the Packet HUB driver from a user mode application. All operations are through the IOCTLs exposed by the Packet HUB driver. Refer to Section 4.3 for details on the IOCTLs. The steps involved in accessing the Packet HUB driver from the user mode application are described below:

1. Open the device.
2. Perform read/write operations.
3. Close the device.

5.1 Opening the Device

Packet HUB driver is opened using the Win32 CreateFile API. To get the device name, refer to Section 5.1.1.

5.1.1 Using Packet HUB Interface Exposed by the Driver

A device interface class is a way of exporting device and driver functionality to other system components, including other drivers, as well as user-mode applications. A driver can register a device interface class, and then enable an instance of the class for each device object to which user-mode I/O requests might be sent. The Intel® PCH EG20T Packet HUB driver registers the following interface.

No	Interface Name
1	GUID_DEVINTERFACE_IOHPACKETHUB

This is defined ioh_pcieqos_common.h.

Device interfaces are available to both kernel-mode components and user-mode applications. User-mode code can use SetupDiXxx functions to find out about registered, enabled device interfaces.

Please refer the following site to get the details about SetupDiXxx functions.

http://msdn.microsoft.com/en-us/library/dd406734.aspx

5.2 Read/Write Operation

Refer to Section 4.3.

5.3 Close the Device

Once all the operations related to the Packet HUB driver are complete, the device handle must be freed by calling the Win32 API CloseHandle.

```
CloseHandle(hHandle);
```

6.0 Default Settings

Table 3 lists the default settings that the Packet HUB driver sets on the hardware.

Refer to the “Register Address Map | Memory-Mapped I/O Registers | Device Control Registers” section in the “Packet Hub” chapter of the *Intel® Platform Controller Hub EG20T External Design Specification* for other default settings.

Table 3. Default Register Values Used in Packet HUB Driver for Initialization

Register	Value
Device Read Pre-Fetch Control Register (offset start 014h offset end 017h)	0x000ffffa
Interrupt Reduce Value Dev0 Func1 (offset start 044h offset end 047h)	0x25

Note: The performance of SATA/Gigabit Ethernet/USB/SDIO decreases when the above values are changed.