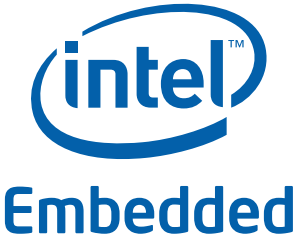

Intel® Platform Controller Hub EG20T

General Purpose Input Output (GPIO) Driver for Windows* Programmer's Guide

February 2011

Order Number: 324257-002US

Contents

Tables

Revision History

Date	Revision	Description
February 2010	002	Updated Section 2.0, “Operating System (OS) Support” on page 6 Added Section 5.1.1, “Using GUID Interface Exposed by the Driver” on page 15
September 2010	001	Initial release

1.0 Introduction

This document provides the programming details of the General Purpose Input Output (GPIO) driver for Windows*. This includes the information about the interfaces exposed by the driver and how to use those interfaces to drive the GPIO hardware.

2.0 Operating System (OS) Support

The GPIO 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 GPIO Driver API Details

This section provides information about the interfaces exposed by the GPIO driver. The current implementation of the driver exposes the interfaces through Input/Output Controls (IOCTLs), which 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 GPIO hardware to work properly.

4.1 Features

The GPIO Driver supports:

- Setting of different configurations for GPIO hardware
- Writing data to GPIO hardware
- Reading data from GPIO hardware
- Setting the direction of GPIO hardware
- Enabling/disabling the interrupts from GPIO hardware
- User-mode program notification by overlapping I/O when an interrupt occurs in the GPIO hardware

4.2 Interface Details

Table 1 lists IOCTLs supported by the driver.

Table 1. Supported IOCTLs

No	IOCTL	Remarks
1	IOCTL_GPIO_ENABLE_INT	Enable the interrupt to selected pins of given GPIO port; set the interrupt mask and mode
2	IOCTL_GPIO_DISABLE_INT	Disable the interruption to selected pins of given GPIO port
3	IOCTL_GPIO_READ	Read the data of selected pins of given GPIO port
4	IOCTL_GPIO_WRITE	Write the data to selected pins of given GPIO port
5	IOCTL_GPIO_DIRECTION	Set the direction of the selected pins of given GPIO port
6	IOCTL_GPIO_NOTIFY	Read the data of selected pins of given GPIO port, when the status of the pin is changed

4.3 IOCTL Usage Details

This section assumes a single client model where there is a single application-level program configuring the GPIO interface and initiating I/O operations. The following files contain the details of the IOCTLs and data structures used:

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

4.3.1 IOCTL_GPIO_ENABLE_INT

This IOCTL is called to enable the interrupt for the selected pins. The prerequisite for this is that the device must be installed and opened using the Win32 API CreateFile.

```
ioh_gpio_reqt gpio_reqt;
```

```
gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_IN;
gpio_reqt.enable = 1;

DeviceIoControl(hHandle,
                IOCTL_GPIO_ENABLE_INT,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);
```

4.3.2 IOCTL_GPIO_DISABLE_INT

This disables the interrupts of GPIO pins.

```
ioh_gpio_reqt gpio_reqt;

gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_IN;
gpio_reqt.enable = 1;

DeviceIoControl(hHandle,
                IOCTL_GPIO_DISABLE_INT,
                &&gpio_reqt,
                sizeof(gpio_reqt),N
                ULL,
                0,
                &dwSize,
                NULL);
```

4.3.3 IOCTL_GPIO_DIRECTION

This sets the direction of the selected pins of given GPIO port.

```
ioh_gpio_reqt gpio_reqt;
```

```
gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_IN;
gpio_reqt.enable = 1;
DeviceIoControl(hHandle,
                IOCTL_GPIO_DIRECTION,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);
```

4.3.4 IOCTL_GPIO_READ

Read the data of selected pins of given GPIO port.

```
ioh_gpio_reqt gpio_reqt,out_buffer;

gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_IN;
gpio_reqt.enable = 1;

DeviceIoControl(hHandle,
                IOCTL_GPIO_DIRECTION,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);

DeviceIoControl(hHandle,
                IOCTL_GPIO_READ,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                &out_buffer,
```

```
                sizeof(out_buffer),
                &dwSize,
                NULL);
```

4.3.5 IOCTL_GPIO_WRITE

The write operation writes to the selected pins of the GPIO.

```
ioh_gpio_reqt gpio_reqt,out_buffer;


gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_OUT;
gpio_reqt.enable = 1;


DeviceIoControl(hHandle,
                IOCTL_GPIO_DIRECTION,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);


DeviceIoControl(hHandle,
                IOCTL_GPIO_WRITE,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                &out_buffer,
                sizeof(out_buffer),
                &dwSize,NULL);
```

4.3.6 IOCTL_GPIO_NOTIFY

Read the data of selected pins of given GPIO port, when the status of the pin is changed.

```
ioh_gpio_reqt gpio_reqt,out_buffer;


gpio_reqt.pins = 0xff;
```

```
gpio_reqt.mode = GPIO_IN;

gpio_reqt.enable = 1;


DeviceIoControl(hHandle,
                IOCTL_GPIO_ENABLE_INT,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);


DeviceIoControl(hHandle,
                IOCTL_GPIO_DIRECTION,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);


DeviceIoControl(hHandle,
                IOCTL_GPIO_NOTIFY,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                &out_buffer,
                sizeof(out_buffer),
                &dwSize,
                NULL);
```

4.4 Structures, Enumerations and Macros

This section provides the details on the structures, enumerations and macros used by interfaces exposed by the GPIO driver. All the structures, enumerations and macros used by the interfaces are defined in ioh_gpio_common.h.

4.4.1 Structures

4.4.1.1 ioh_gpio_reqt

This structure is a used for preserving information related to the GPIO request.

Table 2. ioh_gpio_reqt structure

Name	Description
ULONG port	Set the port number
ULONG pins	Data in the case of read and write and in other cases it indicates applicable pins
UINT64 mode	Set the interrupt mode and direction mode
ULONG enable	Set 1 for enabling the interrupt and 0 for disabling the interrupt

4.4.2 Enumerations

4.4.2.1 GPIO_SET_MODE

This enum is used for preserving information related to the imode.

Table 3. GPIO_SET_MODE enumeration

Name	Description
DISABLE_SEL	Disable selection
ENABLE_SEL	Enable selection
NEITHER	Select neither

4.4.2.2 GPIO_SET_DIR

This enum is used for preserving information related to the direction.

Table 4. GPIO_SET_DIR enumeration

Name	Description
INPUT_SEL	Set direction as input
OUTPUT_SEL	Set direction as output
NEITHER	Set neither direction

4.4.2.3 GPIO_SET_VALUE

This enum is used for setting a GPIO pin to active low or active high.

Table 5. GPIO_SET_ VALUE enumeration

Name	Description
GPIO_LOW	Set GPIO active low
GPIO_HIGH	Set GPIO active high

4.4.3 Macros

These are the Macro definitions exposed by the GPIO driver.

Table 6. Macros

Name	Description
IOH_GPIO_ALL_PINS	This is used to select all pins

4.5 Error Handling

Since the IOCTL command is implemented using the Windows* API, the return value of the call is dependent on and defined by the OS. On Windows*, the return value is 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 GPIO driver from a user mode application. All operations are through the IOCTLs exposed by the GPIO driver. Refer to Section 4.3 for details on the IOCTLs. The steps involved in accessing the GPIO driver from the user mode application are described below:

- Open the device.
- Initialize and configure the driver with desired settings through the interfaces exposed.
- Perform read/write operations.
- Close the device.

5.1 Opening the Device

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

5.1.1 Using GUID 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 GPIO driver registers the following interface.

No	Interface Name
1	GUID_DEVINTERFACE_IOHGPIO

This is defined ioh_gpio_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/ff549791.aspx

5.2 Driver Configuration

The following IOCTLS are used to initialize and configure the settings for the GPIO driver:

- IOCTL_GPIO_DIRECTION
- IOCTL_GPIO_ENABLE_INT
- IOCTL_GPIO_DISABLE_INT

DeviceIoControl Win32 API is used for sending information to the GPIO driver.

```
ioh_gpio_reqt gpio_reqt;


gpio_reqt.pins = 0xff;
```

```
gpio_reqt.mode = GPIO_IN;

gpio_reqt.enable = 1;


DeviceIoControl(hHandle,

                IOCTL_GPIO_ENABLE_INT,

                &&gpio_reqt,

                sizeof(gpio_reqt),

                NULL,

                0,

                &dwSize,

                NULL);
```

5.3 Read / Write Operation

IOCTL_GPIO_READ and IOCTL_GPIO_WRITE are used for read and write operations respectively.

- Read Operation:

```
ioh_gpio_reqt gpio_reqt,out_buffer;


gpio_reqt.pins = 0xff;

gpio_reqt.mode = GPIO_IN;

gpio_reqt.enable = 1;


DeviceIoControl(hHandle,

                IOCTL_GPIO_DIRECTION,

                &&gpio_reqt,

                sizeof(gpio_reqt),

                NULL,

                0,

                &dwSize,

                NULL);


DeviceIoControl(hHandle,

                IOCTL_GPIO_READ,

                &&gpio_reqt,

                sizeof(gpio_reqt),
```

```
                &out_buffer,
                sizeof(out_buffer),
                &dwSize,
                NULL);
```

- Write Operation:

```
ioh_gpio_reqt gpio_reqt,out_buffer;


gpio_reqt.pins = 0xff;
gpio_reqt.mode = GPIO_OUT;
gpio_reqt.enable = 1;
DeviceIoControl(hHandle,
                IOCTL_GPIO_DIRECTION,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                NULL,
                0,
                &dwSize,
                NULL);


DeviceIoControl(hHandle,
                IOCTL_GPIO_WRITE,
                &&gpio_reqt,
                sizeof(gpio_reqt),
                &out_buffer,
                sizeof(out_buffer),
                &dwSize,
                NULL);
```

5.4 Close the Device

Once all the operations related to the GPIO driver are finished, the device handle must free the application by calling the Win32 API CloseHandle.

```
CloseHandle(hHandle);
```