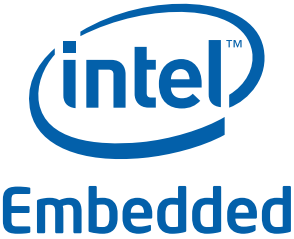


Intel® Platform Controller Hub EG20T

Direct Memory Access (DMA) Controller Driver for Windows* Programmer's Guide

March 2011

Order Number: 324256-002US

Contents

Figures

Tables

Revision History

Date	Revision	Description
March 2011	002	Updated Section 2.0, “Operating System (OS) Support” on page 7
September 2010	001	Initial release

1.0 Introduction

This document provides the programming details for the Intel® Platform Controller Hub EG20T Direct Memory Access (DMA) controller driver for Windows*. This includes information about the interfaces exposed by the driver and how to use these interfaces to drive the DMA hardware.

Direct memory access is a mechanism that allows function IPs to transfer data without giving overhead to the CPU. A DMA can be used to transfer large blocks of data from the peripherals to memory or vice versa. The role of the CPU in this mode of data transfer is to initiate the DMA. Once the transfer is over, the DMA hardware informs the CPU by generating an interrupt.

2.0 Operating System (OS) Support

The DMA 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 drivers delivered.

4.0 DMA Driver API Details

This section provides information about the interfaces exposed by the DMA driver. The current implementation of the driver exposes the interfaces through Input/Output Controls (IOCTLs) and exported functions, which can be called from another driver (kernel mode). No interfaces are exposed to the user mode applications. The following sections provide information about the IOCTLs and exported functions of the driver, and how to use them to drive the DMA hardware to perform the DMA operations successfully. The functionality of the driver can be used only by Intel® Platform Controller Hub EG20T function IPs, which have been wired to the DMA hardware.

Note: Currently, the function IPs that have been wired to the DMA hardware are: SPI and UART.

4.1 Features

The DMA driver supports:

- Up to 4, 8 or 12 channels per device
- Transfer from function IP in Intel® PCH EG20T to memory
- Transfer from memory to function IP in Intel® PCH EG20T
- One-shot mode and scatter/gather mode
- Access size setting: 8 bit, 16 bit and 32 bit
- Timeout setting
- Priority setting
- Transfer size at one-shot mode as follows:
  - Up to 2047 byte if access size is 8 bit
  - Up to 4094 byte if access size is 16 bit
  - Up to 4096 byte if access size is 32 bit
- Transfer size at scatter/gather mode as follows:
  - Up to 2047 byte per descriptor if access size is 8 bit.
  - Up to 4094 byte per descriptor if access size is 16 bit.
  - Up to 4096 byte per descriptor if access size is 32 bit.
- Fixed address mode and incremental address mode
- dreq handshake between DMA controller and function IP

4.2 Interface Details

The DMA driver is implemented as a kernel mode export driver. It also provides support for IOCTL interface. A programmer can choose either of the options depending upon the requirement. Use exported functions if another driver in the kernel mode needs to do the DMA from the ISR (Interrupt Service Routine) or from higher IRQLs. For other instances use the IOCTLs. Refer to the *Microsoft Driver Development Kit* for more details on ISR and different IRQL levels.

Note: No interfaces are exposed to the user mode applications.

Table 1 lists the functions exported by DMA driver.

Table 1. Functions Exported by DMA Driver

No	Function Name	Description
1	ioh_request_dma	Function to request (allocate) the DMA channel to perform DMA operation.
2	ioh_free_dma	Function to free the DMA channel allocated by calling the ioh_request_dma.
3	ioh_set_dma_channel	Function to configure the DMA channel for the DMA operation.
4	ioh_enable_dma_channel	Function to enable the DMA transfer for the specified channel.
5	ioh_disable_dma_channel	Function to disable the DMA transfer for the specified channel.
6	ioh_direct_start_dma	Function to forcefully start the DMA operation. This function is only used for testing purposes.
7	ioh_add_dma_desc	Function adds DMA descriptor information to the existing list of DMA descriptors. This function is only used in scatter-gather mode of DMA operation.

Table 2 lists the supported DMA driver IOCTLs.

Table 2. DMA Driver IOCTLs

No	IOCTL	Description
1	IOCTL_DMA_REQUEST_DMA	IOCTL used to request (allocate) the DMA channel to do the DMA operation.
2	IOCTL_DMA_FREE_DMA	IOCTL used to free the DMA channel allocated by calling the IOCTL_DMA_FREE_DMA.
3	IOCTL_DMA_SET_DMA	IOCTL to configure the DMA channel for the DMA operation.
4	IOCTL_DMA_ADD_DESCRIPTOR	This IOCTL adds DMA descriptor information to the existing list of DMA descriptors. This IOCTL is only used in scatter-gather mode of DMA operation.
5	IOCTL_DMA_ENABLE_DMA	IOCTL to enable DMA transfer for the specified channel.
6	IOCTL_DMA_DISABLE_DMA	IOCTL to disable DMA transfer for the specified channel.
7	IOCTL_DMA_DIRECT_START	IOCTL to forcefully start the DMA operation. This IOCTL is used only for testing purposes.

4.3 Exported Function Usage Details

This section provides the details of the function exported by the DMA driver to the kernel mode and the usage of these functions to drive the DMA driver properly. The following file contains interface details for the DMA driver:

- ioh_dma_common.h

For the programming details of all the exported functions, refer to Chapter 5.0.

4.3.1 ioh_request_dma

This function requests the DMA driver to allocate/reserve the specified channel for DMA operation.

```
NTSTATUS ioh_request_dma(PIOH_REQUEST_DMA pRequestDMA,PULONG pChannel)
```

4.3.2 ioh_free_dma

This function frees the channel that was allocated using the ioh_request_dma.

```
NTSTATUS ioh_free_dma(ULONG uChannel)
```

4.3.3 ioh_set_dma_channel

This function configures the DMA channel for the DMA operation.

```
NTSTATUS ioh_set_dma_channel(PIOH_SET_DMA pSetDMA)
```

4.3.4 ioh_enable_dma_channel

This function enables the DMA transfer for the specified channel.

```
NTSTATUS ioh_enable_dma_channel(ULONG uChannel)
```

4.3.5 ioh_disable_dma_channel

This function disables the DMA transfer for the specified channel.

```
NTSTATUS ioh_disable_dma_channel(ULONG uChannel)
```

4.3.6 ioh_direct_start_dma

This function forces the start of the DMA operation. This function is only used for testing purposes.

```
NTSTATUS ioh_direct_start_dma(ULONG uChannel)
```

4.3.7 ioh_add_dma_desc

This function adds DMA descriptor information to the existing list of DMA descriptors. This function is only used in scatter gather mode of DMA operation.

```
NTSTATUS ioh_add_dma_desc(PIOH_SET_DMA pSetDMA)
```

For programming details, refer to Chapter 5.0, “Programming Guide”.

4.4 IOCTL Details

This section provides the details for configuring the DMA interface and initiating DMA operations. The following files contain the details of the IOCTLs and data structures used:

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

4.5 Structures, Enumerations and Macros

This section provides the structures, enumerations and macros used by interfaces exposed by the DMA driver. All the structures, enumerations and macros used by the interfaces are defined in ioh_dma_common.h.

4.5.1 Structures

4.5.1.1 IOH_REQUEST_DMA

The structure contains the details for requesting the DMA channel.

Table 3. IOH_REQUEST_DMA Structure

Name	Description
IOH_DMA_CHANNELS uChannel	Channel to be reserved
IOH_CHANNEL_REQUEST_ID uDreq	DMA request Rx/Tx
IOH_DMA_REQ_DEVICES uDevID	Requesting Device Information

4.5.1.2 IOH_ADDRESS_PAIR

This contains the physical address and virtual address pair details.

Table 4. IOH_ADDRESS_PAIR structure

Name	Description
ULONG uPhysicalAddress	Physical address
PVOID pVirtualAddress	Virtual address

4.5.1.3 IOH_DMA_ADDRESS_PAIR_DESC

This DMA contains the DMA descriptor details which are used in scatter gather mode.

Table 5. IOH_DMA_ADDRESS_PAIR_DESC Structure

Name	Description
IOH_ADDRESS_PAIR iohInsideAddress	Function IP Address details.This contains the physical address and virtual address details.
IOH_ADDRESS_PAIR iohOutsideAddress	Memory Address information. This contains the details of physical and virtual address of the memory.
ULONG uSize	Data size.
IOH_ADDRESS_PAIR iohNextDescAddress	Next DMA descriptor details in the list of descriptors for DMA transfer.

4.5.1.4 IOH_DMA_DESC

This DMA contains the DMA descriptor details which are used in scatter gather mode.

Table 6. IOH_DMA_DESC structure

Name	Description
ULONG uInsideAddress	Inside address
ULONG uOutsideAddress	Outside address
ULONG uSize	Size
ULONG uNextDesc	Next Descriptor address

4.5.1.5 IOH_DMA_MODE_PARAM

This structure contains the transfer size, DMA direction, and DMA type details.

Table 7. IOH_DMA_MODE_PARAM Structure

Name	Description
UINT16 TransferDirection	Direction of Transfer (IN to OUT or OUT to IN).
UINT16 DMASizeType	Type of DMA Transfer size (8-bit, 16-bit or 32-bit).
UINT16 DMATransferMode	Mode of Transfer (ONE_SHOT_MODE or SCATTER_GATHER_MODE).

4.5.1.6 IOH_SCATTER_GATHER_MODE

This structure contains DMA scatter gather mode details.

Table 8. IOH_SCATTER_GATHER_MODE Structure

Name	Description
PIOH_DMA_ADDRESS_PAIR_DESC pStart	Details of the start DMA descriptor in the list of DMA descriptors
PIOH_DMA_ADDRESS_PAIR_DESC pEnd	Details of the last DMA descriptor in the list of DMA descriptors

4.5.1.7 IOH_ONE_SHOT_MODE

This structure contains DMA one shot mode details.

Table 9. IOH_ONE_SHOT_MODE Structure

Name	Description
IOH_ADDRESS_PAIR iohInsideAddress	Function IP Address
IOH_ADDRESS_PAIR iohOutsideAddress	Memory Details
int count	Data size

4.5.1.8 IOH_SET_DMA

To provide the configuration details for a DMA channel

Table 10. IOH_SET_DMA structure

Name	Description
ULONG uChannel	Handle to the opened channel
int priority	Channel priority
IOH_DMA_MODE_PARAM iohModeParam	Transfer mode details

union {
IOH_ONE_SHOT_MODE iohOneShotMode;
IOH_SCATTER_GATHER_MODE
IohScatterGatherMode;
}
	data for ONE SHOT MODE
	data for SCATTER-GATHER MODE

4.5.2 Enumerations

This section lists the enumerations exposed by the interface.

4.5.2.1 IOH_DMA_REQ_DEVICES

List of devices that can request DMA.

Table 11. IOH_DMA_REQ_DEVICES enumerator

Name	Description
IOH_UART0	UART 0 device
IOH_UART1	UART 1 device
IOH_UART2	UART 2 device
IOH_UART3	UART 3 device
IOH_SPI0	SPI0 device

4.5.2.2 IOH_DMA_CHANNELS

Specifies DMA in use/reserved.

Table 12. IOH_DMA_CHANNELS enumerator

Name	Description
IOH_DMA_CHANNEL0	To request channel 0
IOH_DMA_CHANNEL1	To request channel 1
IOH_DMA_CHANNEL2	To request channel 2
IOH_DMA_CHANNEL3	To request channel 3
IOH_DMA_CHANNEL4	To request channel 4
IOH_DMA_CHANNEL5	To request channel 5
IOH_DMA_CHANNEL6	To request channel 6
IOH_DMA_CHANNEL7	To request channel 7
IOH_DMA_CHANNEL_RESERVED	This channel is reserved

4.5.2.3 IOH_CHANNEL_REQUEST_ID

Transmission and Reception channel request ID.

Table 13. IOH_CHANNEL_REQUEST_ID enumerator

Name	Description
IOH_DMA_TX_DATA_REQ0	To identify the channel type to transmission type
IOH_DMA_RX_DATA_REQ0	To identify the channel type to reception type

4.5.3 Macros

These are the Macro definitions exposed by the DMA driver.

Table 14. Macros

Name	Description
IOH_DMA_DIR_IN_TO_OUT	Set the DMA transfer direction to IN to OUT
IOH_DMA_DIR_OUT_TO_IN	Set the DMA transfer direction to OUT to IN
IOH_DMA_SIZE_TYPE_32BIT	Set the transfer type to 32 bit
IOH_DMA_SIZE_TYPE_16BIT	Set the transfer type to 16 bit
IOH_DMA_SIZE_TYPE_8BIT	Set the transfer type to 8 bit
DMA_SCATTER_GATHER_MODE	Set the DMA transfer mode to scatter gather mode
DMA_ONE_SHOT_MODE	Set the DMA transfer mode to one shot mode

4.6 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.

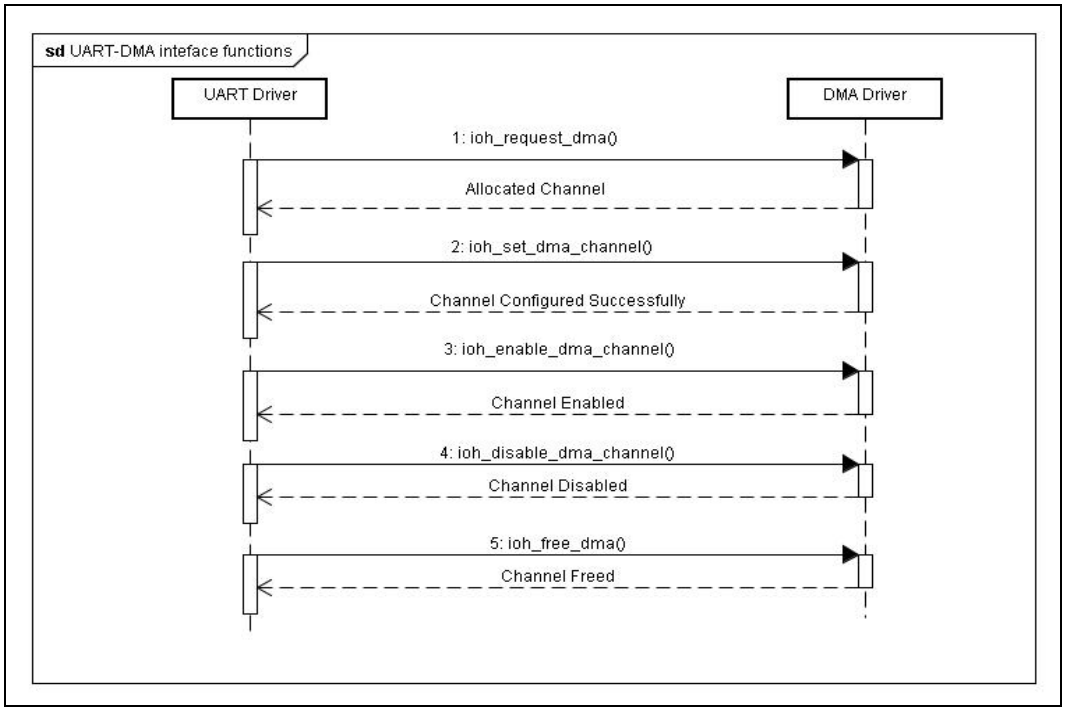
The exported functions which are used in this driver return NT_STATUS error values. More information about the NT_STATUS error value details can be found in the *Windows Driver Development Kit Reference*.

5.0 Programming Guide

This section provides basic programming details of the Intel® Platform Controller Hub EG20T DMA interface. It covers the use of exported functions that are used by other drivers (kernel mode) to perform the DMA operations.

The exported functions must be called in a particular sequence. The details of the sequence are explained in Figure 1.

Figure 1. Exported Function Sequence



The steps involved in configuring the DMA driver are explained below.

Note: Refer to Section 4.5 for details on structures and enumerations used by the exported interface functions.

5.1 Request DMA Channel

To allocate a DMA channel for the device, exported function “ioh_request_dma” is called. Table 15 shows the device and channel relationship.

Table 15. Device and Channel Relationship (Sheet 1 of 2)

DMA No.	Channel	Requests
D10:F0	7	UART3 (D10:F4) RX DATA Request
	6	UART3 (D10:F4) TX DATA Request
	5	UART2 (D10:F3) RX DATA Request
	4	UART2 (D10:F3) TX DATA Request
	3	UART1 (D10:F2) RX DATA Request

Table 15. Device and Channel Relationship (Sheet 2 of 2)

DMA No.	Channel	Requests
	2	UART1 (D10:F2) TX DATA Request
	1	UART0 (D10:F1) RX DATA Request
	0	UART0 (D10:F1) TX DATA Request
D12:F0	3	-
	2	-
	1	SPI (D12:F1) RX DATA Request
	0	SPI (D12:F1) TX DATA Request

The code snippet below explains the use of ioh_request_dma for channel 0 and channel 1 for UART0.

```
IOH_REQUEST_DMA iohRequestDMA ={0};

NTSTATUS ntStatus = STATUS_SUCCESS;

ULONG uChannel0,uChannel1;

/*Request Channel 0*/

iohRequestDMA.uChannel = 0;

iohRequestDMA.uDevID = IOH_UART0;

iohRequestDMA.uDreq =IOH_DMA_TX_DATA_REQ0;


ntStatus = ioh_request_dma(&iohRequestDMA,&uChannel0);

/*Request Channel 1*/

iohRequestDMA.uChannel = 1;

iohRequestDMA.uDevID = IOH_UART0;

iohRequestDMA.uDreq =IOH_DMA_RX_DATA_REQ0;


ntStatus = ioh_request_dma(&iohRequestDMA,&uChannel1);
```

5.2 Configure DMA Channel

The allocated DMA channel can be configured with different settings. The DMA can be done in two modes:

- ONE-SHOT mode
- SCATTER-GATHER mode

The function ioh_set_dma_channel is used to configure the DMA channel.

```
IOH_SET_DMA iohSetDMA = {0};

ULONG uAmoutToWrite = 512;// Size of transfer

PVOID pVirtualAddress = 0xffeeff00;// UART Based address mapped
```

```
ULONG uPhysicalAddress = 0xfeaf9c00;//UART Base address

UCHAR RxBuffer[1024] ={0};

PHYSICAL_ADDRESS tempPhyAddr = MmGetPhysicalAddress(RxBuffer);


/*Configure DMA for ONE SHOT MODE */

iohSetDMA.uChannel = 0;

iohSetDMA.priority = 1;

iohSetDMA.iohModeParam.TransferDirection = IOH_DMA_DIR_OUT_TO_IN;

iohSetDMA.iohModeParam.DMASizeType = IOH_DMA_SIZE_TYPE_8BIT;

iohSetDMA.iohModeParam.DMATransferMode = DMA_ONE_SHOT_MODE;

iohSetDMA.iohOneShotMode.count = uAmountToWrite;

iohSetDMA.iohOneShotMode.iohInsideAddress.pVirtualAddress = pVirtualAddress;

iohSetDMA.iohOneShotMode.iohInsideAddress.uPhysicalAddress = uPhysicalAddress;

iohSetDMA.iohOneShotMode.iohOutsideAddress.pVirtualAddress = RxBuffer;

iohSetDMA.iohOneShotMode.iohOutsideAddress.uPhysicalAddress =tempPhyAddr.LowPart =
tempPhyAddr;

status = ioh_set_dma_channel(&iohSetDMA);
```

5.3 Enable DMA Channel

The function ioh_enable_dma_channel is used to enable DMA transfer on a configured DMA channel.

```
ULONG uDMAChannel = 0;//Allocated Channel

status = ioh_enable_dma_channel(uDMAChannel);
```

5.4 Disable DMA Channel

Once the DMA transfer is over, the DMA channel can be disabled by calling the exported function ioh_disable_dma_channel.

```
ULONG uDMAChannel = 0;//Allocated Channel

status = ioh_disable_dma_channel(uDMAChannel);
```

5.5 Free DMA Channel

The allocated DMA channel is freed using the function ioh_free_dma.

```
ULONG uDMAChannel = 0;//Allocated Channel

status = ioh_free_dma(uDMAChannel);
```