# MB86290 Series Graphics Driver V02 User's Manual Rev. 3.0

FUJITSU LIMITED

## Revision History

| Date | Rev. | Changes of contents (Number in parentheses shows section number) |
|---|---|---|
| May. 30, 2003 | 0.1 | The provisional version |
| Apr. 13, 2004 | 0.2 | • The minimum size of display list buffer is changed from 16416 bytes to 16448 bytes<br>• Deletes the **GdcSetBurstMode** command<br>• Deletes the **GdcCapSetDisplaySize** command (magnify video capture function is not support)<br>• Updates the document name of the "Table 1 List of documents"<br>• Corrects the type of the return value of the **XGdcSetDLBuf** command from GDC_ULONG to GDC_BOOL<br>• Adds the error code **GDC_ERR_ILLEGAL_DL_BUF_NO** to description of the **XGdcFlushEx** command<br>• Corrects the type of the following arguments of the **XGdcBltCopyAltAlpha** command from GDC_ULONG to GDC_USHORT<br>Arguments: x0, y0, x1, y1, bx, by, w, h<br>• Corrects the error of the default value of the **GdcCapSetAttrMisc** command<br>• Corrects description of "Table 4.17.1 Drawing attributes to be restored" of the drawing attribute "For lines on object coordinate system " and "For surfaces on object coordinate system"<br>• Adds the "checks of the completion of drawing" procedure to the processing procedure of the DMA transfer and local display list transmission of the **GdcFlushDisplayList** command<br>• **GDC_TYPE_MB86295** is added to the macro showing the Graphics Controller kind specified by gdctype of an Initialize-Parameter Table (GDC_INITPARAM)<br>• Corrects the supported GDC of the **GdcSetDMAMode** command<br>• Adds **GDC_TYPE_MB86295** to explanation of the return value of the **GdcQueryGDCType** command<br>• Corrects the figure of the log format of the device coordinates in explanation of the **XGdcGeoSetLogOutBase** command<br>• Adds "Transfer of Local Display List by BCU" procedure to the processing procedure of display list transmission of the **GdcFlushDisplayList** command |
| Aug. 11, 2004 | 0.3 | • Adds "3 Application Program Development " chapter |
| Aug. 17, 2004 | 2.0 | • Adds "4 Driver Command List" and "9 Appendix" chapters |
| Apr. 14, 2005 | 2.1 | • Removes all words written by Japanese font |

## Revision History

| Date | Rev. | Changes of contents (Number in parentheses shows section number) |
|---|---|---|
| Mar. 28, 2006 | 3.0 | [Specifications changed]<br>• The minimum size of display list buffer is changed from 16448 bytes to 32800 bytes<br>• Each return value of the following Driver Command is changed from the void type to the GDC_BOOL type<br>[6.18.1],[6.18.3],[6.18.4],[6.18.5],[6.18.7],[6.18.8],[6.18.9],[6.18.10],[6.18.11],[6.18.12]<br>- **GdcCapSetVideoCaptureMode** command<br>- **GdcCapClearErrorStatus** command<br>- **GdcCapSetVideoCaptureBuffer** command<br>- **GdcCapSetImageArea** command<br>- **GdcCapSetWindowMode** command<br>- **GdcCapSetVideoCaptureScale** command<br>- **GdcCapSetAttrMisc** command<br>- **GdcCapSetInputDataCountNTSC** command<br>- **GdcCapSetInputDataCountPAL** command<br>- **GdcCapSetLPFMode** command<br>• The specifications of display list outputted from the **GdcTextureLoadExt16Fast** command is changed [9.1]<br>• Deletes following Driver Command.<br>- **XGdcTextureLoadInt16Fast** command<br>- **XGdcTextureLoadInt24** command<br>- **XGdcTextureLoadExt24** command<br>- **XGdcBltDraw24** command<br><br>[Addition/correction of description]<br>• Adds "3.7 Programming for Video Capture" and "3.8 Programming for Dual Display" section<br>• The description of MB86296S is added to the Driver Command List [4.1 to 4.20]<br>• The column of "Table 4.13 Texture Image Management Command List" is corrected for the internal texture memory uninstalled for MB86293 or later [4.13]<br>• Because MB86295S cannot use a fast bi-linear function, the column of "Table 4.13 textured image data management function list" is corrected [4.13]<br>• Because MB86293 cannot use the video capture function, the column of "Table 4.18 video capture control command list" is corrected [4.18]<br>• **GDC_TYPE_MB86296** is added to the macro showing the kind of the Graphics Controller specified by gdctype of an Initialize Parameter Table (GDC_INITPARAM) [5.2.9]<br>• **GDC_TYPE_MB86296** is added to the description of the return value of the **GdcQueryGDCType** command [6.1.18]<br>• Adds the description of the macro for standard display and the macro for extended display to the description of the following Driver Command<br>[6.4.5],[6.4.8],[6.4.9],[6.4.10],[6.4.11],[6.5.2],[6.5.3]<br>- **GdcDispDimension** command<br>- **GdcDispLayerOn** command<br>- **GdcDispLayerOff** command<br>- **GdcDispPos** command<br>- **GdcDispDoFlip** command<br>- **GdcColorTransparent** command<br>- **GdcColorZeroMode** command<br>• The description concerning 24-bit color mode is deleted from the description of the **XGdcDrawDimension** command because it can be used only with Coral-ES [6.8.1]<br>(Continue) |

## Revision History

| Date | Rev. | Changes of contents (Number in parentheses shows section number) |
|---|---|---|
| Mar. 28, 2006 | 3.0 | (Continued)<br>[Addition/correction of description]<br>• Adds the explanation and notes of 8bit Gouraud shading mode to the explanation of the **XGdcGeoPrimType** command [6.10.1]<br>• Adds **GDC_8BIT_SHADE_MODE** to the macro specified by the argument target of the **XGdcSetAttrSurf** command [6.11.5]<br>• Adds the error code **GDC_ERR_ILLEGAL_DIMENSION** to the explanation of the following driver commands [6.13.2],[6.13.3],[6.13.4]<br>- **XGdcTextureLoadInt[8/16]** command<br>- **XGdcTextureLoadExt[8/16]** command<br>- **XGdcTextureLoadExt16Fast** command<br>• Deletes the following explanation of the **GdcCapSetWindowMode** command [6.18.7]<br>"Sets YC mode when capturing video data."<br>"Color mode supports only 16-bit color mode."<br>• The description is added to the description of the **GdcCapSetAttrMisc** command [6.18.9]<br>• Each description of the following Driver Command is added<br>- **GdcMultiDisplayMode** command<br>- **GdcSingleDisplayMode** command<br>- **GdcMultiDispLayerOn** command<br>- **GdcMultiDispLayerOff** command<br>- **GdcMultiDispCursorOn** command<br>- **GdcMultiDispCursorOff** command<br>- **GdcDispSetYCMatrix** command<br>- **XGdcGeoSetupMode** command<br>- **GdcCapSetAttrVideo** command<br>- **GdcCapStartVideoCapture** command<br>- **GdcCapStopVideoCapture** command<br>- **GdcCapSetCaptureArea** command<br>- **GdcCapSetMaxHorizontalPixel** command<br>- **GdcCapSetMaxVerticalPixel** command<br>- **GdcCapSetRGBInputTiming** command<br>- **GdcCapSetRGBInputSync** command<br>- **GdcCapSetRGBMatrix** command<br>- **GdcCapStartClock** command<br>- **GdcCapStopClock** command<br>- **GdcWriteHostRegister** command<br>- **GdcReadHostRegister** command |

# Introduction

## Purposes and target readers

This document explains mechanism of MB86290 Series Graphics Driver V02 (hereafter referred as the "Graphics Driver") and the Application Program Interface.

This document is written for engineers developing application programs using the Graphics Driver. About the description of this document, premises the readers who understand the specifications of MB86290 Series Graphics Controller (hereafter referred as the "Graphics Controller") and technology about graphics. If needed, refer to the specification of Graphics Controller, or graphics-related books.

## Necessary knowledge about graphics for understanding this document

**Device coordinate system, Object coordinate system, Coordinate transformation, Conversion matrix, Clipping, Polygon, Shading, Z-buffer method, Hidden surface elimination, Texture mapping, Tilling, Anti-aliasing, Alpha blending, Chroma-key composition, Palette color, etc.**

## Specifications of the Graphics Controller

For hardware specifications of the Graphics Controller and programming, refer to the following documents.

- **Graphics Controller Specifications**
- **Application Note**

These documents are prepared for every Graphics Controller. Each Graphics Controller and document names are described in the Table 1 (on next page).

**Table 1 List of Documents**

| Graphics Controller | Document Title |
|---|---|
| MB86290A | MB86290A Graphics Controller Hardware Specifications |
| | MB86290A(Cremson) Application Note |
| MB86291/86291S | MB86291 <SCARLET> Graphics Controller Specifications |
| | MB86291 <SCARLET> Application Note |
| | MB86291S I²C Interface Specification |
| MB86291A/86291AS | MB86291A/291AS <SCARLET> Graphics Controller Specifications |
| | MB86291A <SCARLET2> Application Note |
| | MB86291AS I²C Interface Specification |
| MB86292/86292S | MB86292/292S <ORCHID> Graphics Controller Specifications |
| | MB86292 <ORCHID> Application Note |
| | MB86292S <Orchid> I²C Interface Specification |
| MB86293 | MB86293 <CORAL_LQ> Graphics Controller Specifications |
| | Coral Series Application Note Drawing Edit |
| MB86294/86294S | MB86294/294S <CORAL_LB> Graphics Controller Specifications |
| | MB86294S I²C Interface Specification |
| | Coral Series Application Note Drawing Edit |
| MB86295S | MB86295S <CORAL P> PCI Graphics Controller Specification |
| MB86296S | MB86296S <CORAL PA> PCI Graphics Controller Specification |

A specific Graphics Controller group may be shown as follows in this document.

| | |
|---|---|
| MB86291 or later: | MB86291/86291S/86291A/86291AS/86292/86292S/86293/86294/86294S/86295S/86296S Graphics Controller |
| MB86291: | MB86291/86291S/86291A/86291AS Graphics Controller |
| MB86292: | MB86292/86292S Graphics Controller |
| MB86293/86294: | MB86293/86294/86294S Graphics Controller |
| MB86293 or later: | MB86293/86294/86294S/86295S/86296S Graphics Controller |
| MB86294 or later: | MB86294/86294S/86295S/86296S Graphics Controller |
| MB86295S or later: | MB86295S/86296S Graphics Controller |

# Contents



# 1 MB86290 Series Graphics Driver Overview

This section describes the abstract of the MB86290 Series Graphics Driver.

## 1.1 Overview

MB86290 Series Graphics Driver (hereafter referred as the "Graphics Driver") is a set of commands to assist development of application programs utilizing the MB86290 Series Graphics Controller (hereafter referred as the "Graphics Controller"). Each command to use the Graphics Driver from application programs is called the "Driver Command". Overview of the Graphics Driver is shown by Figure 1.1.

By using this Graphics Driver, application programs can be made without the code concerned with hardware mechanism such as accessing to hardware registers and managing display list (refer to "1.2.4 Display List").

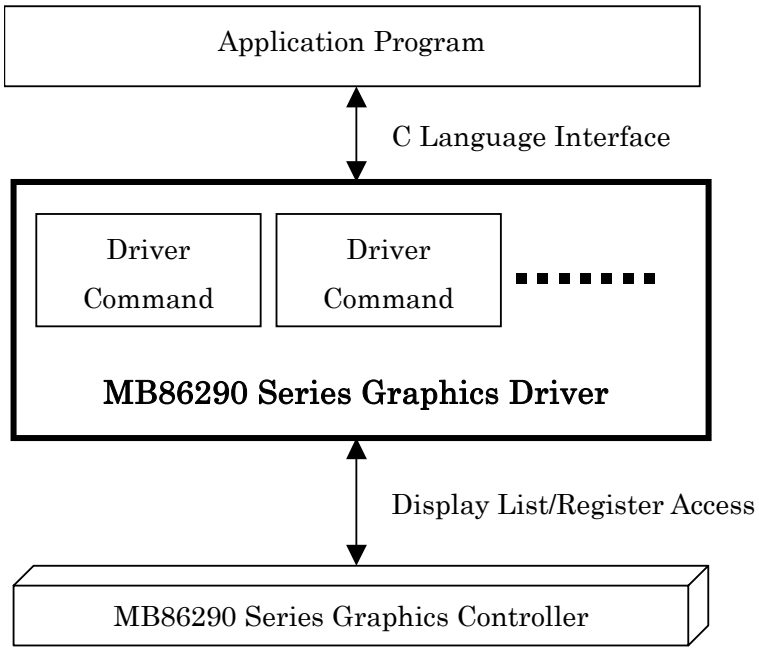



**Figure 1.1 Overview of Graphics Driver**

# 1.2 Component Parts of Graphics Driver

This section describes the component parts of the Graphics Driver.

## 1.2.1 Whole Structure

The Graphics Driver consists of the following parts. Figure 1.2.1 shows the relationship of each part. The description of each element is done at the following.

(1) Driver Command

(2) Context

(3) Display list

(4) System Dependent Command

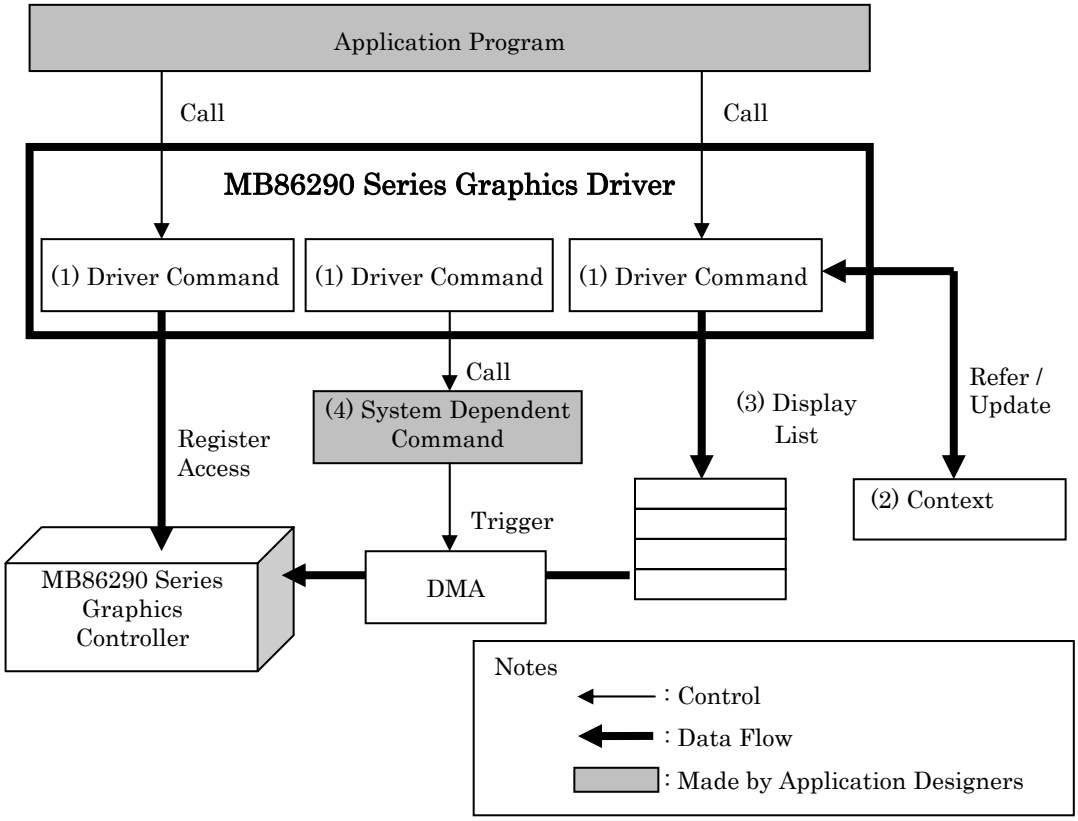



**Figure 1.2.1 Component Parts of Graphics Driver**

## 1.2.2 Driver Command

The Driver Command is an interface function used by application programs. The Driver Command consists of "drawing command", "display controlling command" and etc. By calling the Driver Command, application programs are able to use the function of the Graphics Controller. The types of the Driver Command are described in section "2.1 Types of Driver Command".

### 1.2.3 Context

The "context" is the data that stores the status and rendering attribute of the Graphics Driver.

Before the use of the Driver Command for the graphics drawing, the context needs to be generated by the **XGdcCreateContext** command first. (Refer to the section "6.2.1 **XGdcCreateContext** [Create Context]" for the detail of the **XGdcCreateContext** command.)

Application programs have to allocate the memory area for the context.

When creating two or more tasks which perform a drawing, by generating the context individually by each task, drawing processing of these tasks does not need exclusion control, but can be performed simultaneously.

### 1.2.4 Display List

The display list consists of the various drawing commands and the parameters. The display list is made by drawing command of the Driver Command and transferred to the Graphics Controller. Application programs have to manage to store the display list and transfer them to the Graphics Controller.

Regarding the management of the display list, please refer to "2.2 Management of Display List".

### 1.2.5 System Dependent Command

The System Dependent Command is a command to process procedure such as DMA transfer that depends on a target system and application programs in the Graphics Driver. The System Dependent Command must be designed by each application designer according to the command interface specified by the Graphics Driver.

The detail of this command is described in section "2.3 Types of System Dependent Command".

# 2 Details of Graphics Driver

This section describes the details for the Graphics Driver.

## 2.1 Types of Driver Command

This section describes the type of the Driver Command.

### 2.1.1 Components of Driver Command

The Driver Command can be categorized by the follows. Figure 2.1.1 shows the block diagram of each command.

(1) System control commands

(2) Display control commands

(3) Drawing control commands

(4) Display list control commands

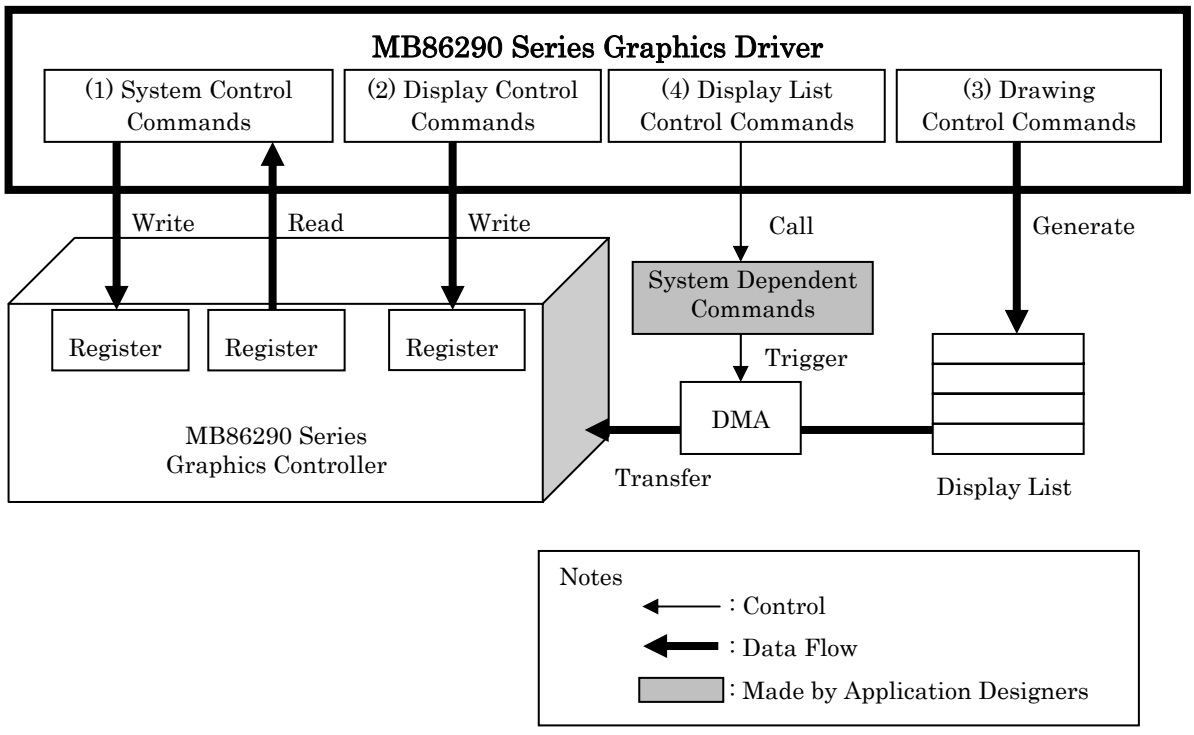



**Figure 2.1.1 Work of Driver Command**

### 2.1.2 System Control Commands

The system control commands are the commands to initialize the driver, set the mode and get the status of the Graphics Controller, generate the context, get the error code and etc.

The system control commands operate the area (error code etc.) that the Graphics Driver manages to operate registers of the Graphics Controller and to control the system.

The system control commands are categorized as follows.

- System commands (Refer to the section "6.1")
- Context control command (Refer to the section "6.2")
- Error control commands (Refer to the section"6.3")
- Register control commands (Refer to the section"6.20")

### 2.1.3 Display Control Commands

The display control commands are the commands for controlling display such as setting layers and displaying cursor. The display control commands operate registers of the Graphics Controller for above control.

The display control commands are categorized as follows.

- Display setting commands (Refer to the section "6.4")
- Color setting commands (Refer to the section "6.5")
- Cursor control commands (Refer to the section "6.6")
- Video capture control commands (Refer to the section "6.18")
- $I^2C$ control commands (Refer to the section "6.19")

### 2.1.4 Drawing Control Commands

The drawing control commands are the commands to generate the display list that draws the graphics primitives such as points, lines, triangles, polygons, etc. on a drawing frame. The drawing control commands are categorized as follows.

- Drawing frame setting commands (Refer to the section "6.8")
- Primitive drawing commands for device coordinate system (Refer to the section "6.9")
- Primitive drawing commands for objects coordinate system (Refer to the section "6.10")
- Drawing attribute setting commands (Refer to the section "6.11")
- Attribute setting commands for object coordinate system (Refer to the section "6.12")
- Texture image management commands (Refer to the section "6.13")
- Binary pattern drawing commands (Refer to the section "6.14")
- BLT commands (Refer to the section "6.15")
- Execution control commands (Refer to the section "6.16")
- Drawing attribute restore command (Refer to the section "6.17")

### 2.1.5 Display List Control Commands

The display list control commands are the commands to flush the display list to the Graphics Controller and to cancel the display list and etc. Drawing by the Graphics Controller is done by transferring the display list to the Graphics Controller. For details, refer to the section "6.7 Display List Control Commands".

# 2.2 Management of Display List

This section describes the management of display list.

## 2.2.1 Store of Display List

The Graphics Driver stores the display list to the display list buffer (hereafter referred as the "DL buffer"). The DL buffer is the area that is in the main memory or the Graphics Controller's local memory (hereafter referred as the "graphics memory"). This area has to be allocated by application programs.

## 2.2.2 Construction of DL Buffer

The DL buffer consists of 4-byte alignment data area and is able to use plural numbers. Regarding the size of the DL buffer is described in "2.2.3 Condition of Size and Allocating Address of DL Buffer".

The DL buffer is related to the context by registering the DL buffer with the **XGdcCreateContext** command when the context is generated. (Refer to "6.2.1 **XGdcCreateContext** [Creates Context]".)

By preparing the multiple DL buffer, both transferring the created display list and creating a new display list in another DL buffer can be executed simultaneously. The display list can be saved to reuse them. (Refer to "3.4 Save and Reuse of Display List".)

The each DL buffer is managed by the serial number from "0". This DL buffer number is allocated to all of the contexts.

Figure 2.2.2 (on next page) shows the example of three DL buffer.

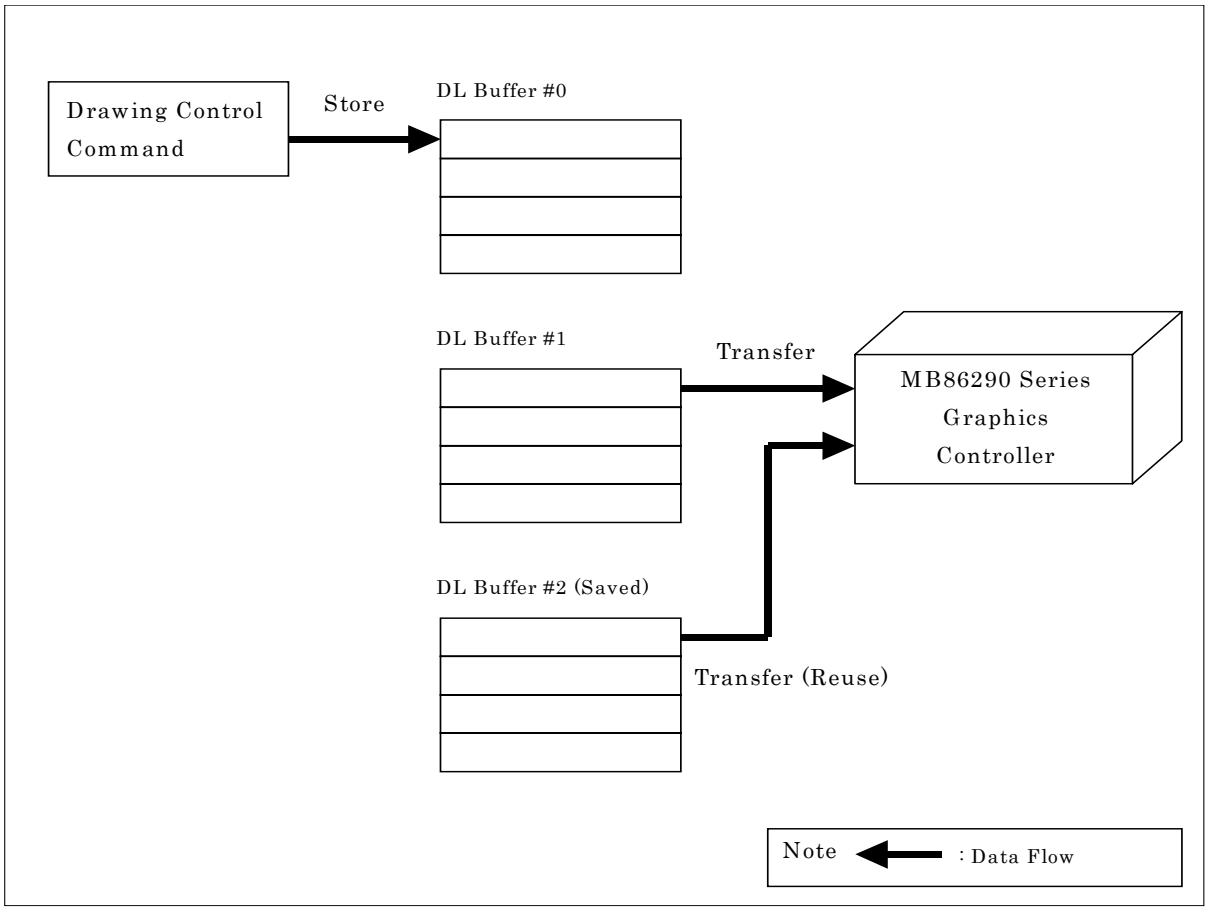



**Figure 2.2.2 Example of Using Three DL Buffers Prepared**

### 2.2.3 Condition of Size and Allocating Address of DL Buffer

The size of a DL buffer should be equal or larger than 32800 bytes and be in multiples of 32 bytes.

The size of a DL buffer is recommended the size of display list for drawing one frame.

The addressing of a DL buffer should be set to 4-byte boundary.

If the size and addressing do not obey the above condition in the **XGdcCreateContext** command, this command returns an error code. In case of it, if execute drawing, the system may hang. Deal with the problem of occurring error in application programs in which an error may be generated. (The details of the **XGdcCreateContext** command refer to the section "6.2.1 **XGdcCreateContext** [Creates Context]".)

### 2.2.4 Selection of DL Buffer

The display list generated by the drawing control commands is stored to "current DL buffer". Current DL buffer is selected by the **XGdcSetDLBuf** command. (The details of the **XGdcSetDLBuf** command refer to the section "6.7.1 **XGdcSetDLBuf** [Sets Current DL Buffer]".) The default number of the DL buffer is "0".

If the **XGdcSetDLBuf** command is called, the contents of selected DL buffer are canceled and the display list is stored from top of the DL buffer. The display list stored in the DL buffer is maintained until the same DL buffer is specified with the **XGdcSetDLBuf** command again.

Figure 2.2.4 shows the image of the selecting the DL buffer.

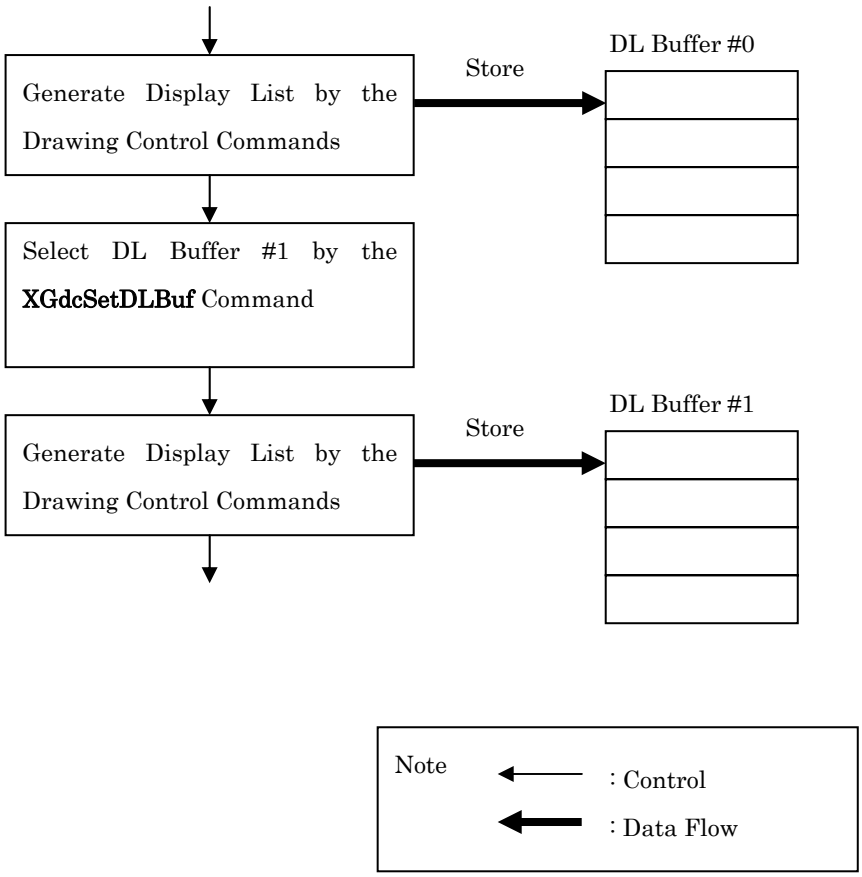



**Figure 2.2.4 Image of Selecting DL Buffer**

### 2.2.5 In Case of Inadequate DL Buffer Size

If the DL buffer is not allocated enough size, DL buffer may be insufficient in generating the display list. In this case, to continue generating the display list, switch the DL buffer from current buffer to other buffer by the **XGdcSwitchDLBuf** command, which is one of the System Dependent Commands. (Refer to the section "2.3 Types of System Dependent Command" and "8. System Dependent Command Interface")

Figure 2.2.5 shows the case of inadequate DL buffer size.

Also, when the **XGdcSwitchDLBuf** command is called, the generated display list is stored across in two or more DL buffers by switching a present DL buffer. For this reason, the Graphics Driver does not support the means though it is necessary to know in which DL buffer in which order the display list is stored to transmit the display list in application programs. Therefore, prepare the mean to acquire the switched DL buffer number when you implement the **XGdcSwitchDLBuf** command.

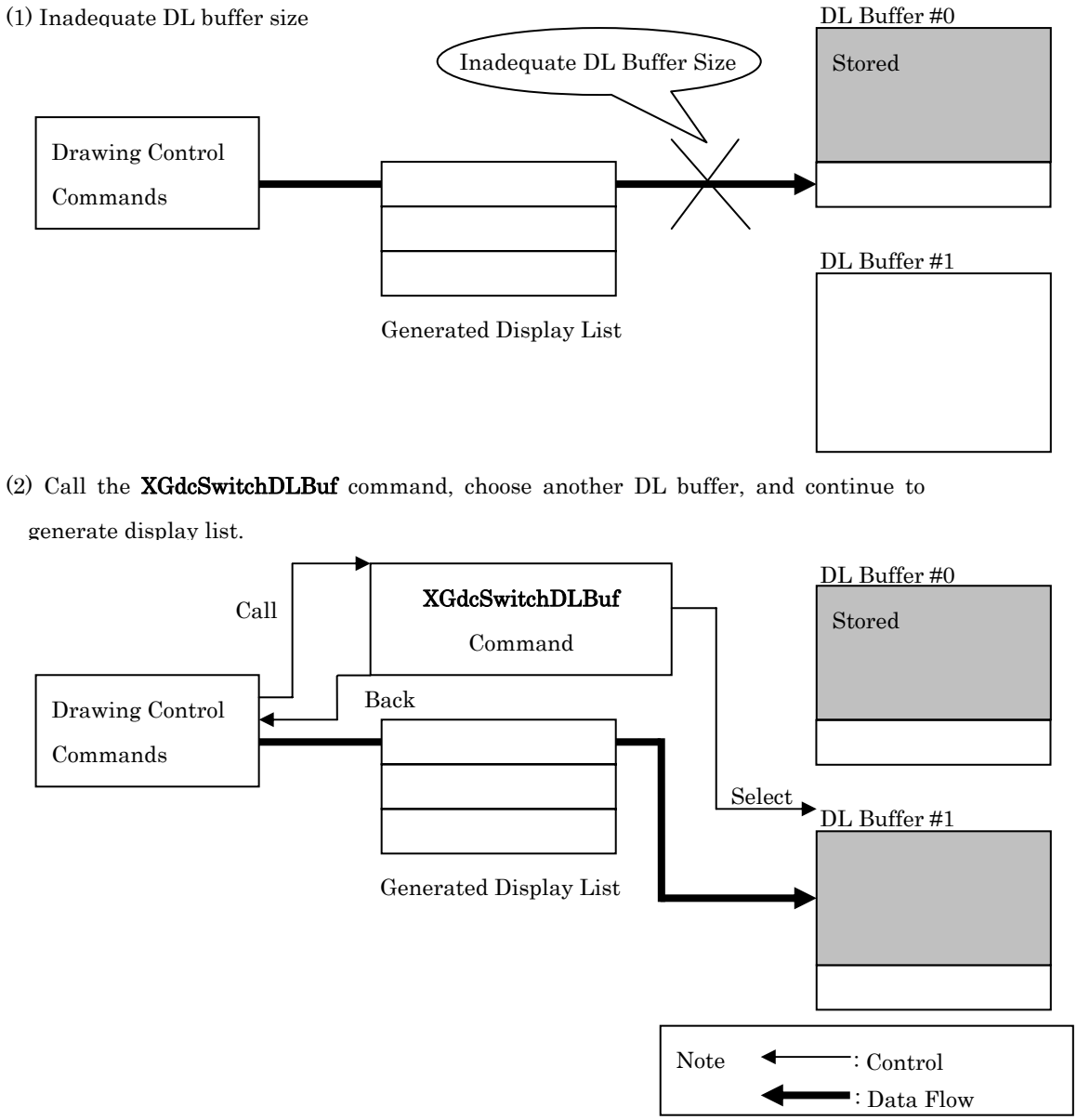



**Figure 2.2.5 Inadequate DL Buffer Size**

### 2.2.6 Method of Transferring Display List

For the display list transfer, the following four options are available. Depends on the target system configuration, each application designer should choose the most appropriate option.

The method of transfer is selected by the **GdcFlushDisplayList** command, which is one of the System Dependent Commands. For the details, refer to the section "2.3 Types of System Dependent Command" and "8. System Dependent Command Interface".

(1) DMA transfer

(2) Reading of display list by the Graphics Controller (Local display list transfer)

(3) Transfer of local display list by BCU

(4) Direct writing to FIFO register in the Graphics Controller by host CPU

## 2.3 Types of System Dependent Command

The System Dependent Commands are the commands to execute the process depending on the target system or application programs such as the procedure of DMA Transfer. Figure 2.3 shows the relationship between the System Dependent Command and the Graphics Driver.

The System Dependent Commands should be programmed by application designers according to the Graphics Driver interface. There are the following three commands in the System Dependent Commands. The details refer to the section "8. System Dependent Command Interface".

(1) **GdcWait** command (Wait process)

This command is called to wait for the time necessary to stabilize the operation after the Graphics Controller is initialized with Graphics Driver's initialization command (**GdcInitialize** command).

(2) **XGdcSwitchDLBuf** command (Switch the DL buffer)

This command is called when switching the DL buffer in case of inadequateness of the DL buffer in the drawing control command.

(3) **GdcFlushDisplayList** command (Transfer the display list)

This command is called when flushing the display list. By the method described in "2.2.6 Method of Transferring Display List", the display list is transferred to the Graphics Controller.

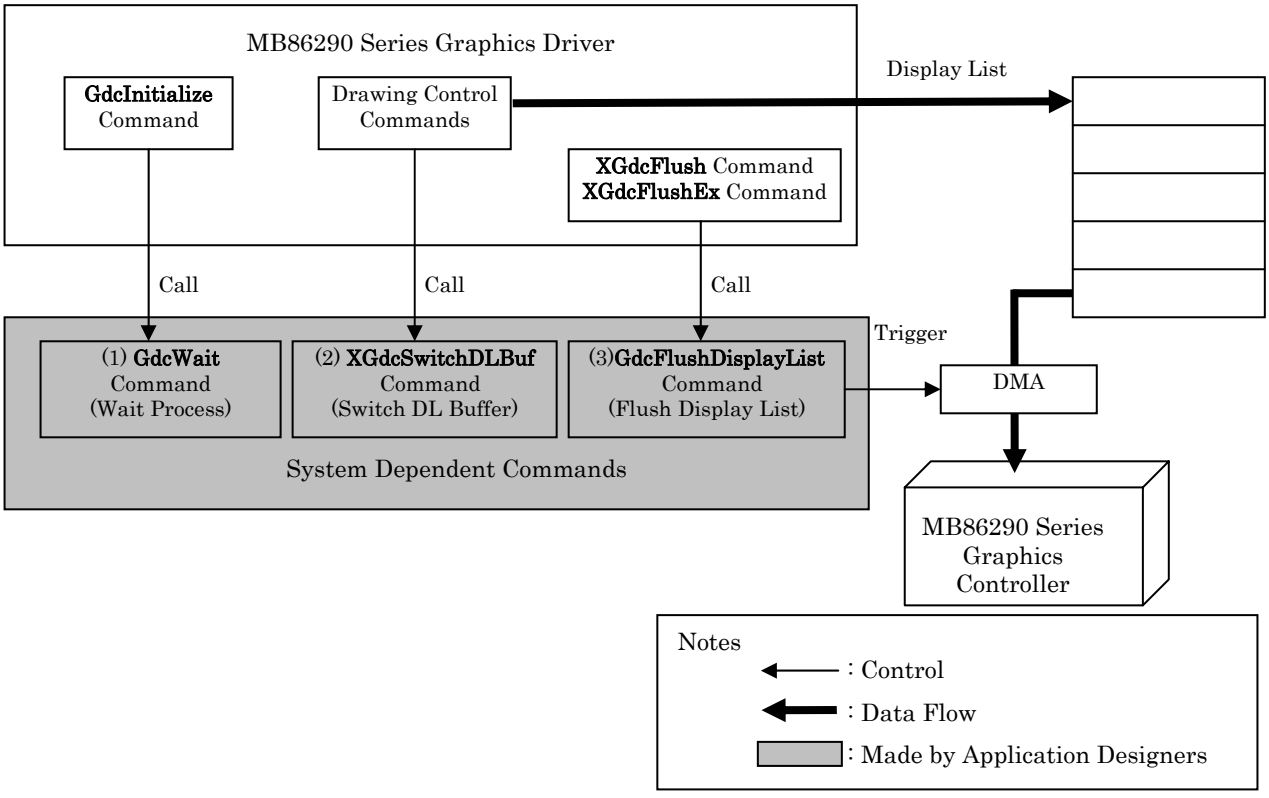



**Figure 2.3 Relationships between System Dependent Command and Graphics Driver**

## 2.4 Error Process

If a parameter value is out of range in the Driver Command that defines the range of parameter, the Driver Command set the error code. (In this case, it returns the **GDC_FALSE** as return value.)

The error code can be got by the **XGdcGetErrCode** command. (For details of the **XGdcGetErrCode** command, refer to the section "6.3.1 **XGdcGetErrCode** [Gets Error Code]".)

If the Driver Command is completed normally (in this case, the Driver Command returns the **GDC_TRUE**), the error code is not set and the last one is maintained as it is.

To clear the error code, set the **GDC_ERR_**NOERROR by the **XGdcSetErrCode** command. (For details of the **XGdcSetErrCode** command, refer to the section "6.3.2 **XGdcSetErrCode** [Sets Error Code]".)

# 3 Application Program Development

This chapter explains a necessary work and programming for developing application programs that use the Graphics Driver.

## 3.1 Necessary Matter in Programming

This section explains a necessary matter in programming.

### 3.1.1 Header Files

This Graphics Driver provides the following header files to application designers.

When this Driver Command is used, include "gdc.h" file.

Application designers must include only "gdc.h" because "gdctypes.h" file is included by "gdc.h".

- gdc.h Driver Command prototype declaration
- gdctypes.h Definition of data types used by this Graphics Driver

### 3.1.2 Allocation of DL Buffer

Allocate the DL buffer by application programs in the size that provided by "2.2.3 Condition of Size and Allocating Address of DL Buffer".

Take care of the source address of the DL buffer when you use DMA to transmit the display list.

(Set these after confirming whether to limit a transferring source address to DMA used.)

DMA transferring source address of transferring the display list is always the top address in each DL buffer.

### 3.1.3 Making System Dependent Commands

Make the System Dependent Commands according to each command interface described in "8. System Dependent Command Interface".

## 3.2 Processing Procedure

A general processing procedure to program using the Graphics Driver is shown in Figure 3.2a.

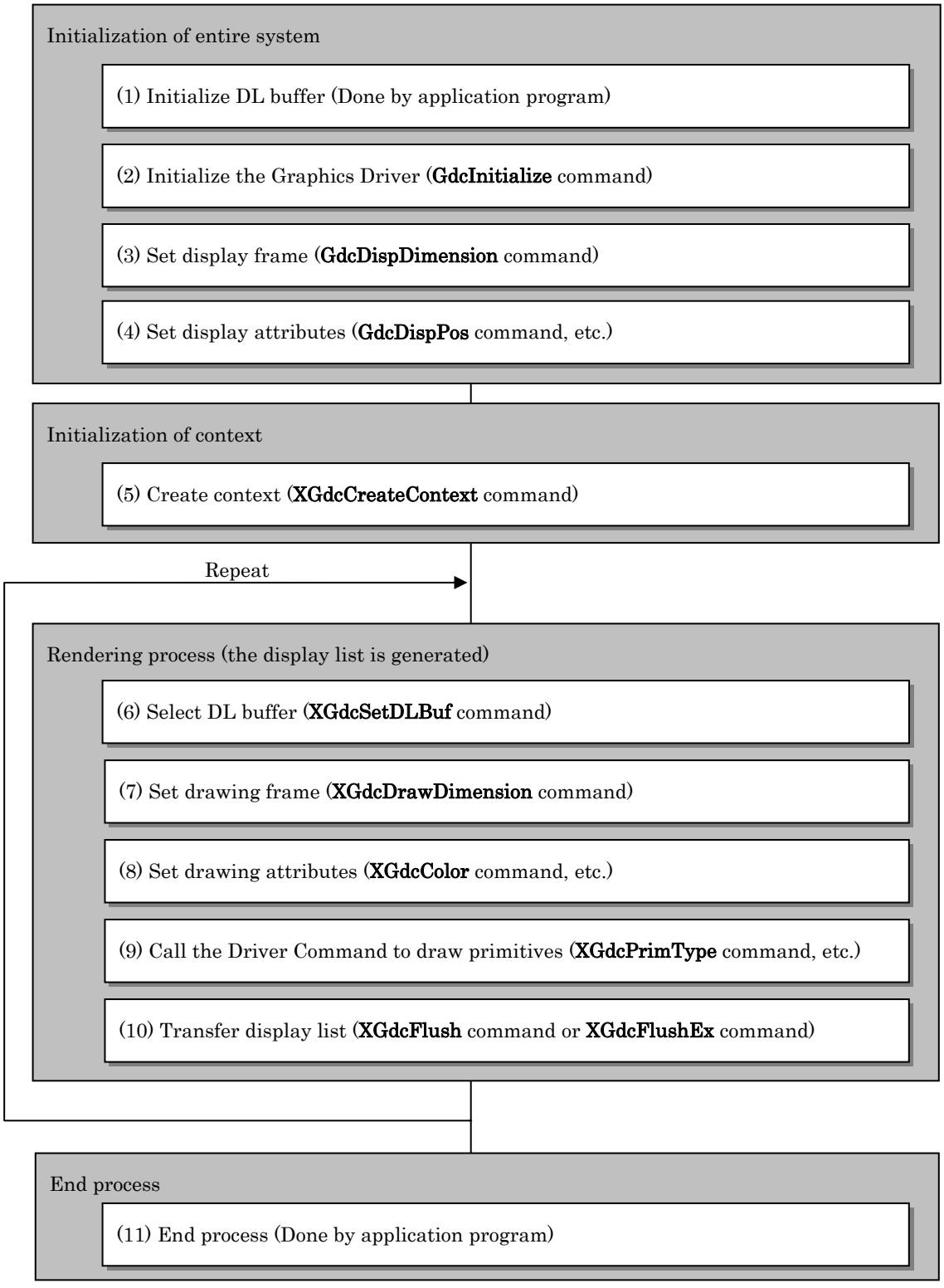



**Figure 3.2a Processing Procedure**

The processing procedures are as follows:

(1) Initialize DL buffer (Done by application programs)

Initialize the DL buffer by the following procedure.

1. Allocate DL buffer
Allocate DL buffer in the main memory or the graphics memory.

[Example of allocating DL buffer]

```
/* Allocate the three DL buffers whose size is DLBUF_SIZE*/
#define  DLBUF_SIZE   (4096 * 32)
GDC_ULONG             dlbuf[3][DLBUF_SIZE];
```

The allocating area of DL buffer depends on the transfer method of the display list. Use the main memory when you use the CPU or DMA display list transfer, and use the graphics memory when you use the local display list transfer.

2. Allocate DL buffer structure information area
Allocate DL buffer structure information area in the main memory.
The example is below.
Specify the total number of DL buffer in the array elements. The number in which the array element is specified is DL buffer number.
Refer to "5.2.10 GDC_DLBUF_STRUCT [DL Buffer Structure Information]".

[Example of allocating DL buffer structure information area]

```
/* Allocate DL buffer structure information's for three DL buffers */
GDC_DLBUF_STRUCT              dlbuf_info[3];
```

3. Set DL buffer structure information
Set the following information to DL buffer structure information. Set this information only once.

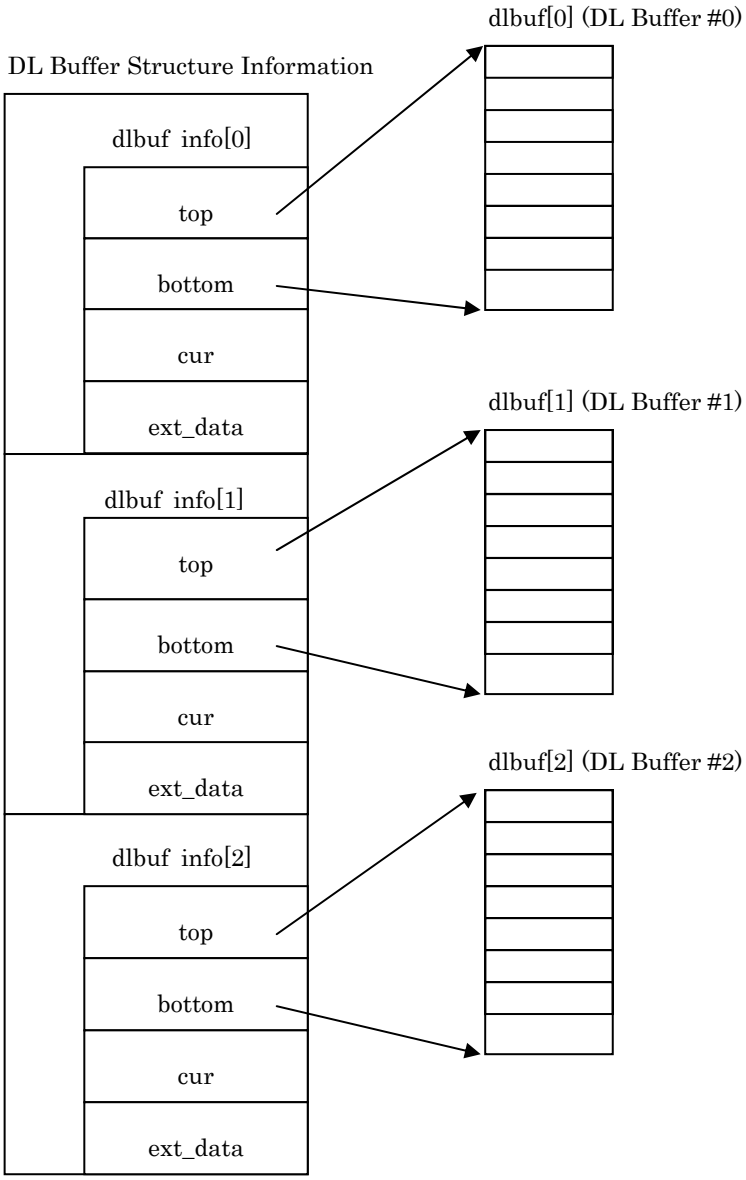
[DL buffer structure information]

- Top address of each DL buffer (top)
- Bottom address +1 of each DL buffer (bottom)

(Notes) Do not set the display list write address (cur) by application programs because this area is managed by the Graphics Driver.

Figure 3.2b shows the example for the relationship between DL buffer structure information and DL buffer when allocate the three DL buffer.




```
[Example for setting to DL Buffer information]
#include “gdc.h"
#define   DLBUF_SIZE          (4096 * 32)
/* Allocate three DL Buffer whose size is DLBUF_SIZE */
GDC_ULONG                     dlbuf[3][DLBUF_SIZE];

/* Allocate three DL Buffer structure information area */
GDC_DLBUF_STRUCT          dlbuf_info[3];

dlbuf_info[0].top             = &dlbuf[0][0];
dlbuf_info[0].bottom          = &dlbuf[0][DLBUF_SIZE];
dlbuf_info[1].top             = &dlbuf[1][0];
dlbuf_info[1].bottom          = &dlbuf[1][DLBUF_SIZE];
dlbuf_info[2].top             = &dlbuf[2][0];
dlbuf_info[2].bottom          = &dlbuf[2][DLBUF_SIZE];
```

**Figure 3.2b DL Buffer Structure Information, and Relation and Setting Example of DL Buffer**

(2) Initialize the Graphics Driver (**GdcInitialize** command)

It is necessary to initialize the Graphics Driver to call the Driver Command and to draw the primitive. Call the **GdcInitialize** command without fail before calling other Driver Command and initialize the Graphics Driver.

(3) Set display frame (**GdcDispDimension** command)

It is necessary to allocate display frame area in the graphics memory according to the size and the number of layers of screens used. Call the **GdcDispDimension** command so that the Graphics Controller can use the allocated display frame area. Call the **GdcDispDimension** command as many times as the number of using layer to set display frame.

(4) Set display attributes (**GdcDispPos** command, etc.)

Call the commands to set display attributes according to the following procedure.

| | |
|---|---|
| 1. **GdcDispPos** command | Set display starting position for each layers. |
| 2. **GdcDispDividePos** command | Set the position where the display is divided into left and right. |
| 3. **GdcColorTransparent** command | Set transparent color for each layers. |
| 4. **GdcColorZeroMode** command | Set whether the color code "0" treats as transparency or not for each layer. |
| 5. **GdcOverlayPriorityMode** command | Set whether to display layer C (layer L0) prioritized in the highest order or to display layer C in blended with the result of prioritizing layers excluding layer C. |
| 6. **GdcDispLayerOn** command | Set display layer. |
| 7. **GdcDispOn** command | Permit the output of the video signal so that the screen display may begin. |

(5) Create context (**XGdcCreateContext** command)

It is necessary to create context before calling the drawing command. Call the **XGdcCreateContext** command to create and initialize context.

(6) Select DL buffer (**XGdcSetDLBuf** command)

It is necessary to select DL buffer before call the drawing control commands to make the display list. Call the **XGdcSetDLBuf** command to select DL buffer number that stores the display list.

(7) Set drawing frame (**XGdcDrawDimension** command)

It is necessary to set drawing frame before draw a primitive. Set color mode, the origin address and the size of the drawing frame. Call the **XGdcDrawDimension** command to set the drawing frame. This command creates the display list.

(8) Set drawing attributes (**XGdcColor** command, etc.)

Set drawing attributes before calling the drawing commands. Set vertex color/foreground color and the drawing attribute of the line width, etc. The drawing attributes are set by calling the "Drawing Attribute Setting Commands" (See the section "6.11") and the "Attribute Setting Commands for Object Coordinate System" (See the section "6.12"). These commands create the display list.

Since the Graphics Driver doesn't have the initial values for the drawing attributes, set all of the necessary attributes.

(9) Call the Driver Command to draw the primitive (**XGdcPrimType** command, etc.)

Call the driver command to draw the primitive of the triangle or the line, etc. To draw the primitive, call the Driver Command such as the "Primitive Drawing Commands for Device Coordinate System" (See the section "6.9") or the "Primitive Drawing Commands for Object Coordinate System" (See the section "6.10"). These Driver Command create the display list.

(10) Transfer display list (**XGdcFlush** command or **XGdcFlushEx** command)

It is necessary to transfer the display list to the Graphics Controller to start drawing by the Graphics Controller. Call the **XGdcFlush** command or the **XGdcFlushEx** command, and transfer the display list to the Graphics Controller.

(11) End Process (Done by the application programs)

Release the memory area allocated by the application programs if necessary.

# 3.3 Programming for Multitask

This section explains programming for multitask.

## 3.3.1 System Configuration

The Graphics Driver is assuming the system composed of two or more tasks. The Graphics Driver is a mechanism that maintains the context and the DL buffer of each drawing task to make it draw separately with two or more drawing tasks (task that uses the function for the drawing control) in the primitive. Drawing is asynchronously enabled from each drawing task with this mechanism.

Because the Driver Command other than the drawing control commands doesn't correspond to a reentrant call, the task that uses this Driver Command can't be asynchronously executed. Do not execute it by another task when using only by one task or using it by two or more tasks until it controls exclusively, and processing ends completely.

Figure 3.3.1 (on next page) shows the example of the system configuration that the Graphics Driver assumes. The explanations of each task are as follows.

(1) Initialization task

The initialization task is a task of initializing the entire system. Do the initialization of the entire system that explains by "3.2 Processing Procedure".

(2) Display task

The display task is a task of controlling the screen display.

(3) Drawing task

The drawing task is a task of drawing a primitive or setting an attribute using the drawing commands.

Each drawing task can be executed asynchronously by keeping the context and the DL buffer individually.

(4) Transfer task

The transferring task is a task that controls transferring the display list stored in the DL buffer to the Graphics Controller.

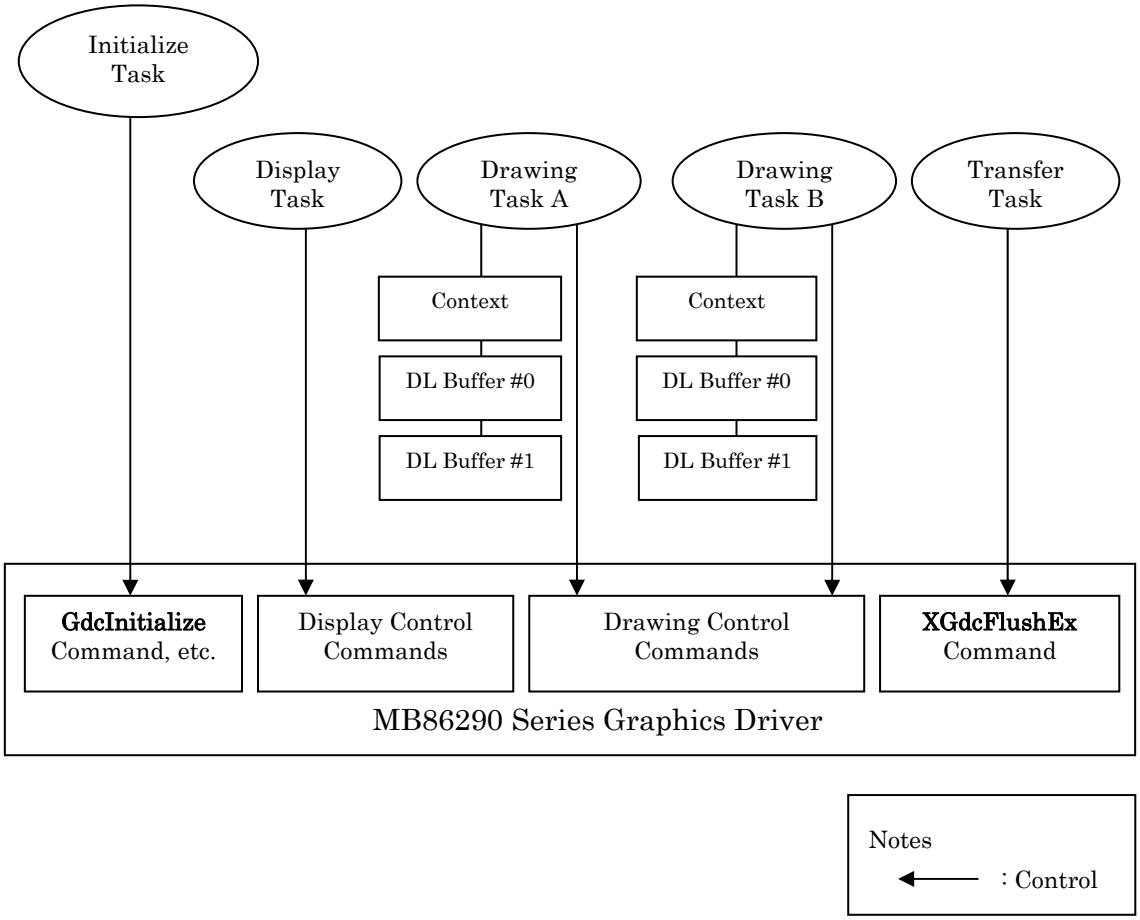



**Figure 3.3.1 Example of System Configuration that Graphics Driver Assume**

### 3.3.2 Transfer Display List

The display list made by each drawing task is transferred by the previous described task.

The display list has to be transferred exclusively or to be controlled transfer order by queuing, etc. because it is impossible to do the multiple transferring the display list to the Graphics Controller simultaneously. Figure 3.3.2 shows the example of queuing. (Because the Graphics Driver doesn't have the queuing mechanism, it is necessary to make it by using the function of OS in the application program side.) In case of controlling the execution order by queuing, it is used the **XGdcFlushEx** command to transfer the display list. And store the following information specified the **XGdcFlushEx** command to the queue.

- Pointer for the context (Address)
- DL buffer number

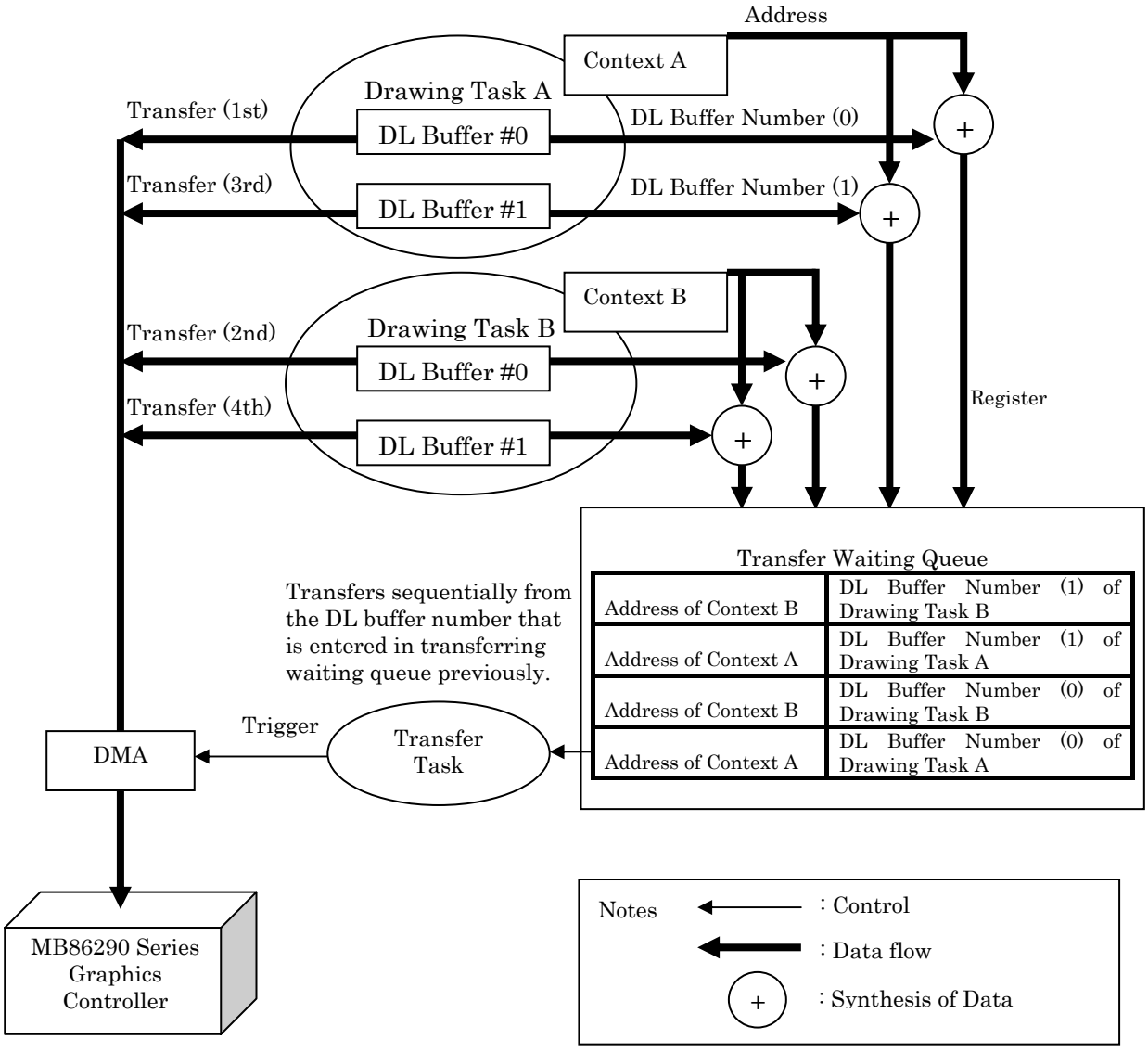



**Figure 3.3.2 Example of Queuing**

### 3.3.3 Restore Drawing Attributes

The drawing attributes after transferring the display list to the Graphics Controller in the same task are not necessarily the same as previous one. This is because there is a possibility that the other drawing task changes the drawing attribute after transmitting the display list to the Graphics Controller. Figure 3.3.3 shows the example of the changing of the drawing attributes by the other task.

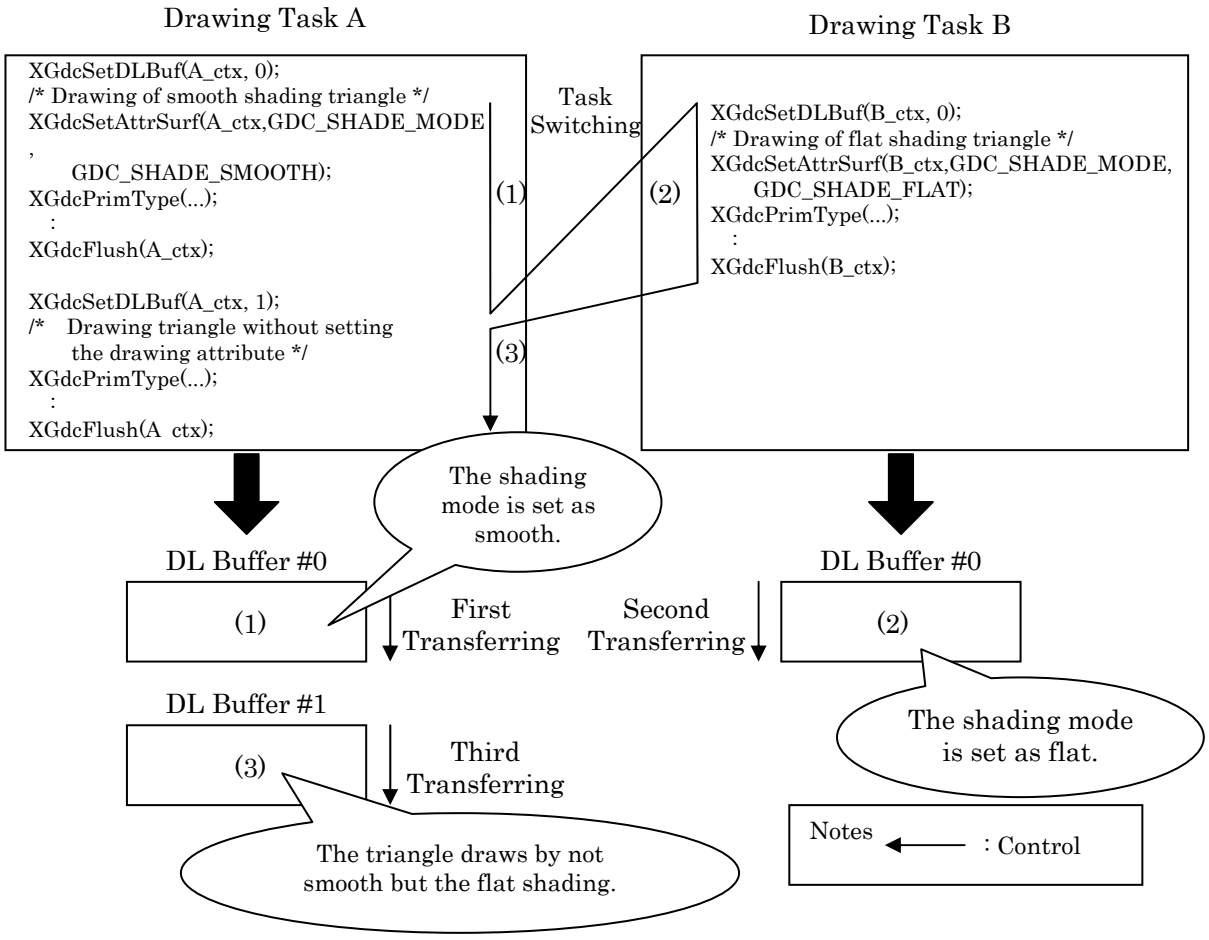



**Figure 3.3.3 Example of Changing Drawing Attribute**

Consequently, it might be necessary to set the drawing attributes again in multitask environment after transferring the display list.

The drawing attributes can be set by calling the drawing control commands individually. Moreover, it is also possible to set two or more drawing attributes again at a time by using the drawing attributes maintained in the context with the drawing control commands last. To restore the drawing attributes, call the **XGdcRestoreAttr** command. It is more efficient to call this command than to call the individual drawing control commands when restoring the state with a lot of drawing attributes to the previous state where the display list is not transferred.

# 3.4 Save and Reuse of Display List

This section explains about save and reuse of display list.

## 3.4.1 Overview of Save and Reuse of Display List

The display list is saved in the DL buffer and able to be reused by the **XGdcFlush** or the **XGdcFlushEx** command until call the **XGdcSetDLBuf** or the **XGdcCancelDisplayList** command. Using this, the drawing process such as the drawing same object can be efficiently done.

Moreover, saving only the drawing primitive display list, it is possible to draw the different scene by changing the drawing attributes. For example, save the display list for triangle and it is possible to draw every time different color triangle by changing the foreground color. Moreover changing the matrix parameter, it is possible to change the drawing position.

### 3.4.2 Procedure of Saving and Reusing Display List

Figure 3.4.2a shows the image of how to save and reuse the display list. In this case, save and reuse the two kinds of the drawing attributes display list and the drawing line display list. And Figure 3.4.2b shows the example of coding.

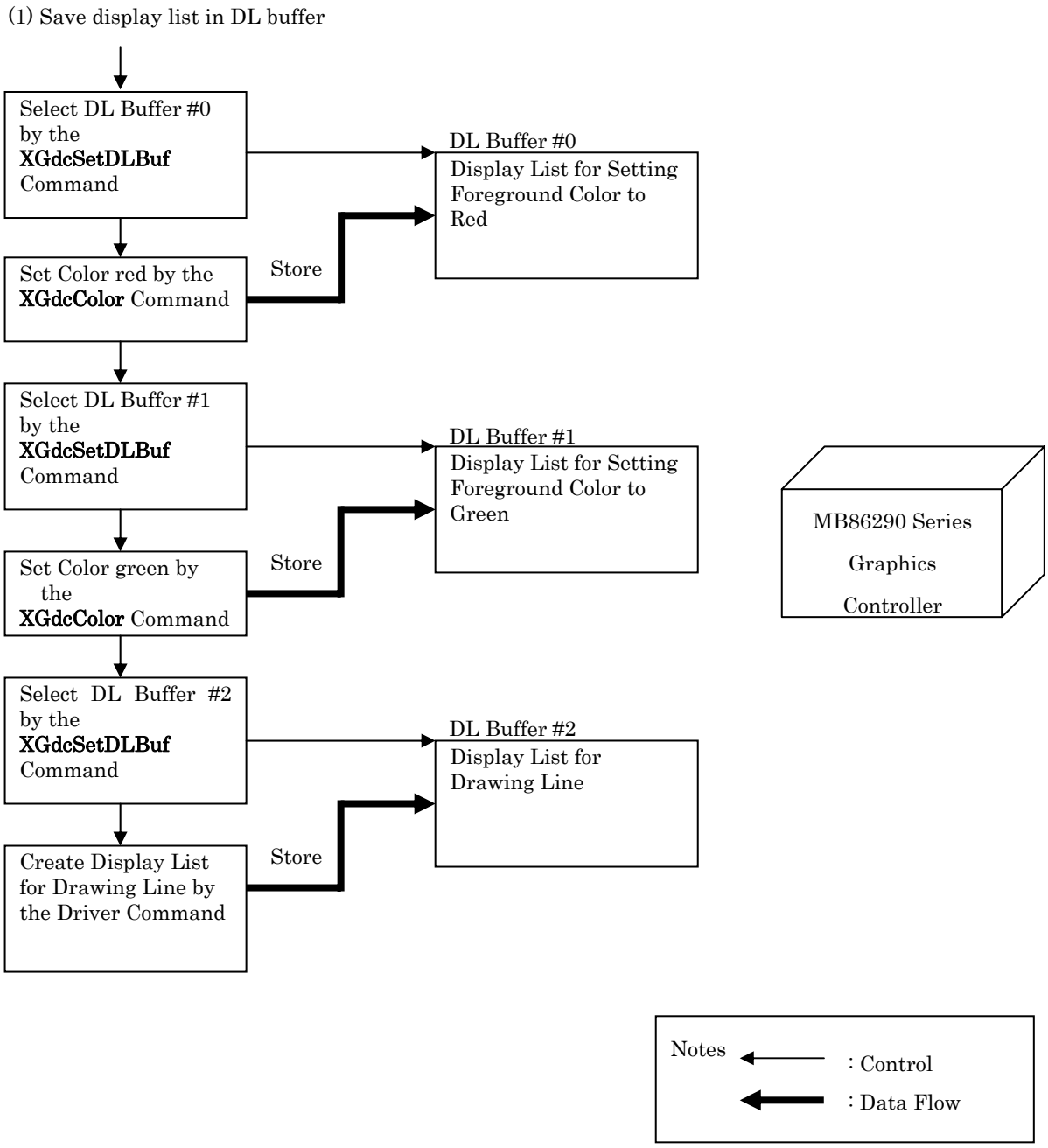



**Figure 3.4.2a Processing Procedure when Three Kinds of Display Lists are Saved, and Reused (Continue)**

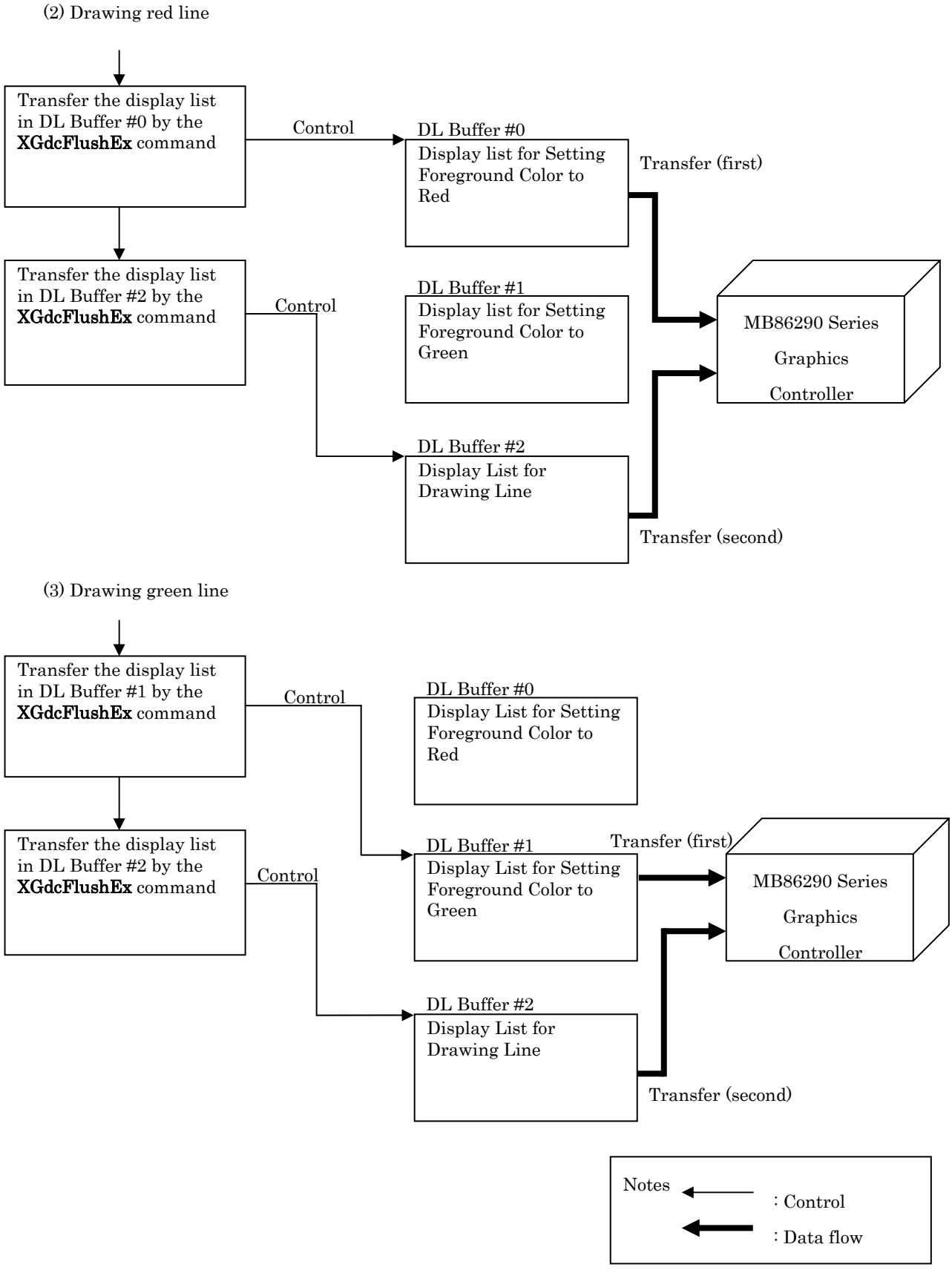



**Figure 3.4.2a Processing Procedure when Three Kinds of Display Lists are Saved, and Reused (Continued)**

<Description of Figure 3.4.2b>

- Condition: The initialization of the entire system has been executed. Pointer to context is "drvctx".

```
/* Select the DL Buffer #0 for saving the display list
   which sets the drawing attribute #1 */
  XGdcSetDLBuf(drvctx, 0)

/* Set the drawing attribute #1 */
  XGdcColor(drvctx, 0x7c00); /* Set foreground color to red */

/* Select the DL Buffer #1 for storing the display list
   which sets the drawing attribute #2 */
  XGdcSetDLBuf(drvctx, 1);

/* Set the drawing attribute #2 */
  XGdcColor(drvctx, 0x03e0); /* Set foreground color to green */

/* Select the DL Buffer #2 for storing the display list which draws the line */
  XGdcSetDLBuf(drvctx, 2);

/* Draw the line */
  XGdcPrimType(drvctx, GDC_LINES_FAST); /* Start of line drawing */
  XGdcDrawVertex2Di(drvctx, 0,0);       /* Vertex #0 */
  XGdcDrawVertex2Di(drvctx, 10,10);     /* Vertex #1 */
  XGdcPrimEnd(drvctx);                  /* End of line drawing */

/* Transfer the display list for setting the drawing attribute #1
   to the Graphics Controller */
  XGdcFlushEx(drvctx, 0);

/* Transfer the display list for drawing the line
   to the Graphics Controller */
  XGdcFlushEx(drvctx, 2);

/* Transfer the display list for setting the drawing attribute #2
   to the Graphics Controller */
  XGdcFlushEx(drvctx, 1);

/* Transfer the display list for drawing the line to the Graphics Controller */
  XGdcFlushEx(drvctx, 2);
```

**Figure 3.4.2b Example of Coding of Saving and Reusing Display List**

### 3.4.3 Notes for Saving and Reusing Display List

It is necessary to consider how to give the drawing attribute when reuse the display list. The display list can be safely used when the setting of the drawing attribute is included in the saved display list, and when the setting of the drawing attribute is not included, it is necessary to set an appropriate drawing attribute in advance.

Moreover, depending on display list to be reused, the setting of the Graphics Controller may fall into an unpredictable state. This is caused because some Driver Command temporarily change the drawing attribute not directly related to and generate the display list that returns the status back at the end.

The value of the display list returns the status back is a value that the context maintains when generating the display list, so the value is not guaranteed to correspond the Graphics Controller's present state. Therefore, it is necessary to set the drawing attribute again for this case by the **XGdcRestoreAttr** command etc.

Table 3.4.3 shows the Driver Command which change the drawing attribute temporarily and the drawing attributes.

**Table 3.4.3 Driver Command which Change Drawing Attributes Temporarily and Drawing Attributes**

| Driver Command | Drawing attributes |
|---|---|
| **XGdcBufferClearZ** command | • Color mode<br>• Clipping mode<br>• Enlarge/shrink mode for binary pattern<br>• Precision of z value<br>• Drawing frame base address<br>• BitBlt attribute<br>• Vertex color/foreground color |
| **XGdcTextureLoadExt8** command | • Color mode<br>• Clipping mode<br>• Enlarge/shrink mode for binary pattern<br>• Precision of z value<br>• Drawing frame base address<br>• BitBlt attribute<br>• Stride for drawing frame |
| **XGdcTextureLoadExt16** command | |
| **XGdcTextureLoadExt16Fast** command | |

## 3.5 Timing of Error Check

The Driver Command set the error code and return the **GDC_FALSE** when an error occurs. The error code is possible to get by the **XGdcGetErrCode** command. Application programs check out an error by this information. However, after an error occurs in the **GdcInitialize** command or the **XGdcCreateContext** command, the system might hang if continuing the processing after that. Therefore perform an error check after these Driver Command are called. It is enough for the other Driver Command to check an error before transfer the display list. Figure 3.5 shows the example of an error check.

```
/* (Notes)"..." means abbreviation.  Bold font means checking error  */
/* Omits the initialization of the DL Buffer and display */
int main(){
        int ret;
        int errcode;
        ret = GdcInitialize(...); /* Initialize the Graphics Driver */
        if(ret != GDC_TRUE){
                /* Error process */
                return GDC_FALSE;
        }
        ret = XGdcCreateContext(...); /* Create context */
        if(ret != GDC_TRUE){
                /* Error process */
                return GDC_FALSE;
        }
        XGdcSetDLBuf(...);/* Select the DL Buffer */
        XGdcDrawDimension(...);/* Set the drawing frame */

        /* Start the drawing triangle */
        XGdcSetAttrSurf(...);
        XGdcColor(...);
        XGdcPrimType(...);
          XGdcDrawVertex2Di(...);
          XGdcDrawVertex2Di(...);
          XGdcDrawVertex2Di(...);
        XGdcPrimEnd(...);
        /* End of drawing triangle */
        /* Error check of drawing triangle */
        errcode = XGdcGetErrCode(...);/* Get error code */
        if(errcode == GDC_ERR_INVALID_ATTRIBUTE){
                /* Processing corresponding to error code */
                :
        }
        XGdcFlushEx(...); /* Transfer the display list */
}
```

**Figure 3.5 Example of Error Check**

## 3.6 Combination of Drawing Control Commands

The drawing is done by the combination of the drawing control commands. The follows explains the notes for these commands combination.

(1) The combination of the vertex setting commands between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

Use a vertex setting command with the same data type.

<Correct example>

```
XGdcGeoPrimType
  XGdcGeoDrawVertex2D  /* GDC_FIXED32 type */
  XGdcGeoDrawVertex2D  /* GDC_FIXED32 type */
XGdcGeoPrimEnd
```

<Wrong example>

There is a vertex setting command with a different data type.

```
XGdcGeoPrimType
  XGdcGeoDrawVertex2Di /* GDC_LONG type */ /* NG */
  XGdcGeoDrawVertex2D  /* GDC_FIXED32 type */
XGdcGeoPrimEnd
```

(2) The combination of the texture coordinates setting commands between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

The data type of the texture coordinates setting commands has to be the same with the vertex setting commands. And if the data type of the vertex setting command is GDC_LONG type, use the GDC_FIXED32 type for texture coordinates setting command.

<Correct example>

```
XGdcGeoPrimType
  XGdcGeoTexCoord2DNf /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcGeoTexCoord2DNf /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcGeoTexCoord2DNf /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
XGdcGeoPrimEnd
```

<Wrong example>

The data type of the texture coordinates setting command is different from one of vertex setting command.

```
XGdcGeoPrimType
  XGdcGeoTexCoord2DN /* GDC_FIXED32 type */ /* NG */
  XGdcGeoDrawVertex3Df
  XGdcGeoTexCoord2DNf /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcGeoTexCoord2DNf /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
XGdcGeoPrimEnd
```

(3) The combination of the color setting commands between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

The data type of the color setting commands has to be the same with the vertex setting commands.

<Correct example>

```
XGdcGeoPrimType
  XGdcVertexColor3f /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcVertexColor3f /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcVertexColor3f /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
XGdcGeoPrimEnd
```

<Wrong example>

The data type of the color setting command is different from one of the vertex setting commands.

```
XGdcGeoPrimType
  XGdcVertexColor32 /* GDC_COLOR32 type */ /* NG */
  XGdcGeoDrawVertex3Df
  XGdcVertexColor3f /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
  XGdcVertexColor3f /* GDC_SFLOAT type */
  XGdcGeoDrawVertex3Df
XGdcGeoPrimEnd
```

(4) The combination between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

Use the **XGdcGeoPrimType** command a pair with the **XGdcGeoPrimEnd** command.

<Correct example>

```
XGdcGeoPrimType
  XGdcGeoDrawVertex3D
  XGdcGeoDrawVertex3D
XGdcGeoPrimEnd
```

<Wrong example 1>

The **XGdcGeoPrimEnd** command is not called.

```
XGdcGeoPrimType
  XGdcGeoDrawVertex3D
  XGdcGeoDrawVertex3D
XGdcGeoPrimType
  XGdcGeoDrawVertex3D
  XGdcGeoDrawVertex3D
```

<Wrong example 2>

The **XGdcGeoPrimType** command is not called.

```
  XGdcGeoDrawVertex3D
  XGdcGeoDrawVertex3D
XGdcGeoPrimEnd
```

(5) The combination between the **XGdcPrimType** command and the **XGdcPrimEnd** command.

Use the **XGdcPrimType** command a pair with the **XGdcPrimEnd** command.

<Correct example>

```
XGdcPrimType
  XGdcDrawVertex3D
  XGdcDrawVertex3D
XGdcPrimEnd
```

<Wrong example 1>

The **XGdcPrimEnd** command is not called.

```
XGdcPrimType
  XGdcDrawVertex3D
  XGdcDrawVertex3D
XGdcPrimType
  XGdcDrawVertex3D
  XGdcDrawVertex3D
```

<Wrong example 2>

The **XGdcPrimType** command is not called.

```
  XGdcDrawVertex3D
  XGdcDrawVertex3D
XGdcPrimEnd
```

(6) The combination between the **XGdcBufferClearZ** command and the **XGdcDrawDimension** command.

Call the **XGdcDrawDimension** command before calling the **XGdcBufferClearZ** command.

<Correct example>

```
XGdcDrawDimension
XGdcBufferClearZ
```

<Wrong example>

The **XGdcDrawDimension** command has never been called before calling the **XGdcBufferClearZ** command.

```
XGdcBufferClearZ /* NG */
XGdcDrawDimension
```

(7) The combination between the **XGdcBufferClearC** command and the **XGdcDrawDimension** command.

Call the **XGdcDrawDimension** command before calling the **XGdcBufferClearC** command.

<Correct example>

```
XGdcDrawDimension
XGdcBufferClearC
```

<Wrong example>

The **XGdcDrawDimension** command has never been called before calling the **XGdcBufferClearC** command.

```
XGdcBufferClearC /* NG */
XGdcDrawDimension
```

(8) The combination of the drawing control commands between the **XGdcPrimType** command and the **XGdcPrimEnd** command.

The following drawing control commands can be called between the **XGdcPrimType** command and the **XGdcPrimEnd** command. Don't call the other than the following commands between the **XGdcPrimType** command and the **XGdcPrimEnd** command.

- **XGdcDrawVertex2D**
- **XGdcDrawVertex2Di**
- **XGdcDrawVertex3D**
- **XGdcDrawVertex3Df**
- **XGdcTexCoord2D**
- **XGdcTexCoord2Df**
- **XGdcTexCoord2DNf**
- **XGdcTexCoord3D**
- **XGdcTexCoord3Df**
- **XGdcTexCoord3DNf**
- **XGdcColor**
- **XGdcBackColor**

It is possible to call the commands below outside between the **XGdcPrimType** command and the **XGdcPrimEnd** command. (Also possible to call those before the **XGdcPrimType** command.)

- **XGdcColor**
- **XGdcBackColor**

<Correct example>

```
XGdcSetAttrSurf /* Set drawing attributes for surface */
XGdcColor       /* Set vertex color */
/* Start drawing */
XGdcPrimType
     XGdcDrawVertex2D
     XGdcDrawVertex2D
     XGdcDrawVertex2D
XGdcPrimEnd
```

<Wrong example>

There is a command other than above between the **XGdcPrimType** command and the **XGdcPrimEnd** command.

```
XGdcColor       /* Set vertex color */
/* Start drawing */
XGdcPrimType
     XGdcSetAttrSurf    /* Set drawing attributes for surface */ /* NG */
     XGdcDrawVertex2D
     XGdcDrawVertex2D
     XGdcDrawVertex2D
XGdcPrimEnd
```

(9) The combination of the drawing control commands between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

The following drawing control commands can be called between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command. Don't call the other than the following commands between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

- **XGdcGeoDrawVertex2D**
- **XGdcGeoDrawVertex2Df**
- **XGdcGeoDrawVertex2Di**
- **XGdcGeoDrawVertex3D**
- **XGdcGeoDrawVertex3Df**
- **XGdcGeoDrawVertex3Di**
- **XGdcGeoTexCoord2DN**
- **XGdcGeoTexCoord2DNf**
- **XGdcVertexColor32**
- **XGdcVertexColor3f**
- **XGdcColor**
- **XGdcBackColor**
- **XGdcGeoShadowColor**
- **XGdcGeoShadowBackColor**
- **XGdcGeoBorderColor**
- **XGdcGeoBorderBackColor**
- **XGdcGeoShadowXY**
- **XGdcGeoOverlapZ**

It is possible to call the commands below outside between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

- **XGdcColor**
- **XGdcBackColor**
- **XGdcGeoShadowColor**
- **XGdcGeoShadowBackColor**
- **XGdcGeoBorderColor**
- **XGdcGeoBorderBackColor**
- **XGdcGeoShadowXY**
- **XGdcGeoOverlapZ**

<Correct example>

```
XGdcSetAttrLine /* Set drawing attributes for line */
XGdcColor       /* Set vertex color */
/* Start drawing */
XGdcGeoPrimType
     XGdcGeoDrawVertex2D
     XGdcGeoDrawVertex2D
XGdcGeoPrimEnd
```

<Wrong example>

There is a command other than those above between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

```
XGdcColor       /* Set vertex color */
/* Start drawing */
XGdcGeoPrimType
     XGdcSetAttrLine    /* Set drawing attributes for line */ /* NG */
     XGdcGeoDrawVertex2D
     XGdcGeoDrawVertex2D
XGdcGeoPrimEnd
```

# 3.7 Programming for Video Capture

This section explains about changing in the video capture mode setting specification and procedures for video capture.

When using a digital video decoder connected via an I²C interface, initialize the digital video decoder by using the I²C control commands (Driver Command name that start with GdcI2C) before executing video capture.

## 3.7.1 About Changes in Video Capture Mode Setting Specifications

Though two or more video capture modes are used by the **GdcCapSetVideoCaptureMode** command in up to Graphics Driver V02L02, the specification has changed to that video capture modes are individually used by new Driver Command for V02L03.

In the Table 3.7.1 the relations between the **GdcCapSetVideoCaptureMode** command and new Driver Command are shown. The **GdcCapSetVideoCaptureMode** command is scheduled to be abolished in the Graphics Driver of the next version though the **GdcCapSetVideoCaptureMode** command can also be used to keep compatibility for now.

**Table 3.7.1 Relations between GdcCapSetVideoCaptureMode Command and New Driver Command**

| Video Capture Mode | **GdcCapSetVideoCaptureMode** Command | New Driver Command |
|---|---|---|
| Start of Video Capture | Specify with **GDC_CAP_START** Macro | **GdcCapStartVideoCapture**() |
| Stop of Video Capture | Specify with **GDC_CAP_STOP** Macro | **GdcCapStopVideoCapture**() |
| Use Interpolation Processing in the Vertical Direction | Specify with **GDC_CAP_ENABLE_V_INTERPOLATION** Macro | **GdcCapSetAttrVideo**( **GDC_CAP_V_INTERPOLATION_MODE**, **GDC_ENABLE**) |
| Do Not Use Interpolation Processing in the Vertical Direction | Specify with **GDC_CAP_DISABLE_V_INTERPOLATION** Macro | **GdcCapSetAttrVideo**( **GDC_CAP_V_INTERPOLATION_MODE**, **GDC_DISABLE**) |
| Video Format = NTSC | Specify with **GDC_CAP_NTSC** Macro | **GdcCapSetAttrVideo**( **GDC_CAP_VIDEO_SELECT**, **GDC_CAP_NTSC**) |
| Video Format = PAL | Specify with **GDC_CAP_PAL** Macro | **GdcCapSetAttrVideo**( **GDC_CAP_VIDEO_SELECT**, **GDC_CAP_PAL**) |

### 3.7.2 Procedure for Video Capture

The Graphics Controller can capture video data which is in the ITU-R BT656 format or the RGB666 format. After video data is stored in the graphics memory area allocated by users (video capture buffer), it is displayed.

The procedure for capturing video data is shown in Figure 3.7.2.

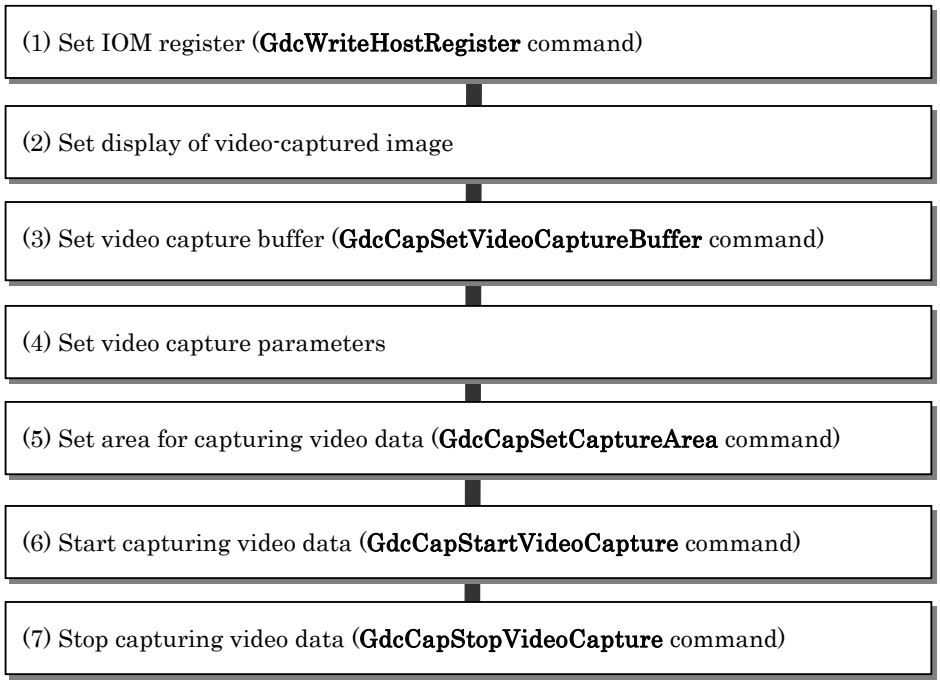



**Figure 3.7.2 Procedures for Capturing Video data**

The processing procedure is as follows:

(1) Set IOM register (**GdcWriteHostRegister** command)

When capturing video data in the RGB666 format, it is necessary to control the IOM register to set RGB input enable. When capturing video data in the ITU-R BT656 format, this procedure is omissible.

RGB input function can be exclusively used with the following functions;

- GPIO function
- Serial interface function

The IOM register is controlled by the **GdcWriteHostRegister** command and the **GdcReadHostRegister** command. Refer to "6.20 Register Control Commands" for details.

(2) Set display of video-captured image

Set the following to display video-captured image on the layer L1 (layer W) using the following corresponding the Driver Command.

- Set display frame attribute of layer L1 (**GdcDispDimension** command)
- Set position and size of layer L1 (**GdcDispSetLayerWindow** command)
- Set display mode of layer L1 (**GdcCapSetWindowMode** command)

(3) Set video capture buffer (**GdcCapSetVideoCaptureBuffer** command)

Set video capture buffer using the **GdcCapSetVideoCaptureBuffer** command. Video capture buffer must be of the following size, depending on the size of the image to be captured.

size = width * height * frame

| | |
|---|---|
| size: | Required buffer size |
| width: | Width of image to be captured |
| height: | Height of image to be captured |
| frame: | Number of frames |

The top (saddr) and bottom (eaddr) address of video capture buffer must be on a 16-byte boundary, and the width of video capture buffer must be allocated in 64-byte unit. In addition, set a value that is smaller for two rasters to the bottom address specified to the **GdcCapSetVideoCaptureBuffer** command. Therefore, set the following value for the parameter eaddr of the **GdcCapSetVideoCaptureBuffer** command.

eaddr = saddr + 64 * stride * (height * frame – 2)

| | |
|---|---|
| saddr: | Top address (16-byte boundary) |
| stride: | Buffer width (by 64-byte unit) |
| height: | Height of graphics to be captured |
| frame: | Number of frames |

As a result, the area from saddr to (eaddr + 2 rasters) is used as video capture buffer.

(4) Set video capture parameters

Set video capture parameters, using the **GdcCapSetAttrVideo** command and the **GdcCapSetAttrMisc** command.

(5) Set area for capturing video data (**GdcCapSetCaptureArea** command)

Set area of the image of the video capture target and after enlargement or reduction of the image size using the **GdcCapSetCaptureArea** command.

(6) Start capturing video data (**GdcCapStartVideoCapture** command)

Start capturing video data using the **GdcCapStartVideoCapture** command.

(7) Stop capturing video data (**GdcCapStopVideoCapture** command)

Stop capturing video data using the **GdcCapStopVideoCapture** command after capturing image is completed.

### 3.7.3 RGB Input

When using RGB input, the settings with the Driver Command are as follows.

- Set input format to RGB (**GDC_CAP_CAPTURE_FORMAT** is specified for the first argument of the **GdcCapSetAttrMisc** command)
  - One of the following is specified for the second argument of the **GdcCapSetAttrMisc** command at RGB input.

    **GDC_CAP_RGBIN_YCOUT**

    **GDC_CAP_RGBIN_RGB555OUT**

    **GDC_CAP_RGBIN_RGB888OUT**
  - One of the following is specified for the second argument of the **GdcCapSetAttrMisc** command at native RGB input (*1).

    **GDC_CAP_NATIVE_RGBIN_RGB555OUT**

    **GDC_CAP_NATIVE_RGBIN_RGB888OUT**
- Set RGB input area (**GdcCapSetRGBInputTiming** command)
- Set RGB input synchronization signal (**GdcCapSetRGBInputSync** command)
- Set transformation matrix from RGB format to YCbCr format (**GdcCapSetRGBMatrix** command)

(*1) Native RGB input is a function that stores video data in RGB form as is without converting into the YCbCr form. It is possible to use this function only with MB86296S.

### 3.7.4 Scaling

The Graphics Controller can scale the video-captured image to display. The procedure for scaling the video-captured image is shown in Figure 3.7.4.


**Figure 3.7.4 Scaling Procedure**

The processing procedures is as follows:

(1) Set area of capturing video data (**GdcCapSetCaptureArea** command)

Set range of captured-image to be targeted and the size of image after enlargement or reduction by the **GdcCapSetCaptureArea** command.

(2) Set display layer (**GdcDisplaySetLayerWindow** command)

Set display position and size of the layer L1 using the **GdcDispSetLayerWindow** command. In case of enlargement with the **GdcCapSetCaptureArea** command, the position of the layer L1 is not set though the size of the layer L1 is automatically set, set it by calling this command.

### 3.7.5 Application to Texture Mapping

Video-captured image can be used for texture mapping by setting the following. Use the Driver Command for this setting.

- Set the buffer mode of the video capture buffer to single buffer (**GdcCapSetAttrMisc** command)
- Set the format for writing to the video capture buffer to RGB mode (**GdcCapSetAttrMisc** command)
- Set the width of the memory of the video capture buffer to the value of the power of "2" (**GdcCapSetVideoCaptureBuffer** command)

Video-captured images are stored in RGB format in the video capture buffer. Therefore, when setting texture image data using the **XGdcTextureDimension** command, set the starting address of the video capture buffer as the origin address of the texture data.

When using video-captured image only as a texture image, it is not necessary to specify the video capture buffer area for the layer L1 display frame.

# 3.8 Programming for Dual Display

This section explains the overview and the procedure of the dual display.

## 3.8.1 Overview of Dual Display

The dual display function is a function to display both any layers and any cursors on two displays. This function can be used only with MB86296S. It is necessary to set the following display modes to use this function.

## 3.8.2 Display Mode

It is necessary to set the mode of the display (hereafter referred as the "display mode") according to the number of displays connected by the Graphics Controller.

- The number of display is two: Sets to the dual display mode.
- The number of display is one: Sets to the single display mode. (Default is the single display mode.)

The display condition of each layer and each cursor are maintained in each display mode.

When one of displays is made the status of not-display because the display background color is always displayed on two displays, it is necessary to set black (0x0) to the display background color. When the Graphics Driver is initialized, the display background color sets black (0x0). Please note the display background color when it changes by application programs besides the black, and one of displays are made the status of not-display.

### 3.8.3 Shift Procedure to Dual Display Mode

The procedure when shifting to the dual display mode is shown in Figure 3.8.3.

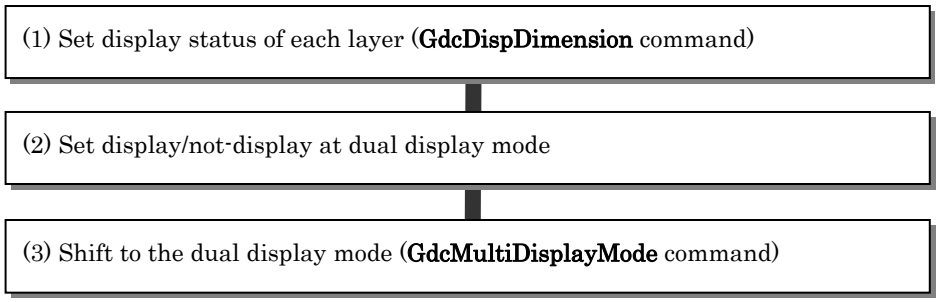



**Figure 3.8.3 Procedure for shifting to Dual Display Mode**

The shifting procedure is as follows:

(1) Set display status of each layer (**GdcDispDimension** command)

The display status of each layer is set by the **GdcDispDimension** command.

(2) Set display/not-display at dual display mode

Display/not-display of each layer and each cursor are done by the following settings at the dual display mode. Moreover, the Driver Command used for the setting is shown at the same time.

- Setting of layer display (**GdcMultiDispLayerOn** command)
- Setting of layer not-display (**GdcMultiDispLayerOff** command)
- Setting of cursor display (**GdcMultiDispCursorOn** command)
- Setting of cursor not-display (**GdcMultiDispCursorOff** command)

(3) Shift to the dual display mode (**GdcMultiDisplayMode** command)

Display status shifts to the dual display mode by the **GdcMultiDisplayMode** command. Each layer becomes the display status set by the **GdcDispDimension** command if procedure (2) is omitted, and the cursor is not-display.

After the display status shifts to the dual display mode, display/not-display of each layer and each cursor can be arbitrarily set by the Driver Command shown according to procedure (2).

### 3.8.4 Shift Procedure to Single Display Mode

The procedure when shifting to the single display mode is shown in Figure 3.8.4.

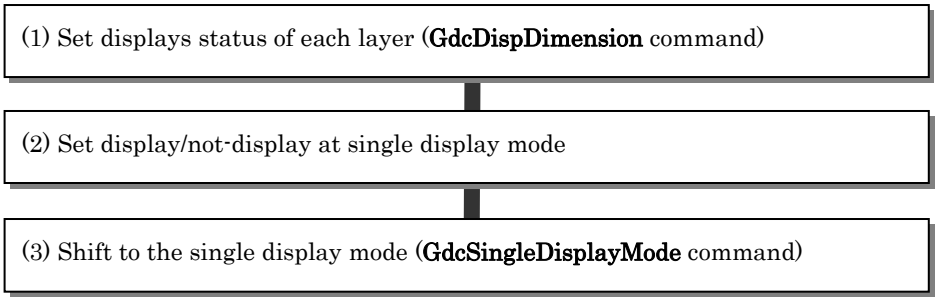



**Figure 3.8.4 Procedure for shifting to Single Display Mode**

The shifting procedure is as follows:

(1) Set display status of each layer (**GdcDispDimension** command)

The display status of each layer is set by the **GdcDispDimension** command.

(2) Set display/not-display at single display mode

Display/not-display of each layer and each cursor is done by the following settings at the single display mode. Moreover, the Driver Command used for the setting is shown at the same time.

- Setting of layer display (**GdcDispLayerOn** command)
- Setting of layer not-display (**GdcDispLayerOff** command)
- Setting of cursor display (**GdcCursorDisplay** command)
- Setting of cursor not-display (**GdcCursorDisplay** command)

(3) Shift to the single display mode (**GdcSingleDisplayMode** command)

Display status shifts to the single display mode by the **GdcSingleDisplayMode** command. Each layer becomes the display status set by the **GdcDispDimension** command if procedure (2) is omitted, and the cursor is not-display. If the display mode has never shifted from single display mode to dual display mode after the initialization of the Graphics Driver, this procedure is omissible.

After the display status shifts to the single display mode, display/not-display of each layer and each cursor can be arbitrarily set by the Driver Command shown according to procedure (2).

# 4 Driver Command Lists

This section describes each Graphics Controller can use any Driver Command.

Driver Command is shown in the following.

- System commands
- Context control command
- Error control commands
- Display setting commands
- Color setting commands
- Cursor control commands
- Display list commands
- Drawing frame setting commands
- Primitive drawing commands for device coordinate system
- Primitive drawing commands for object coordinate system
- Drawing attribute setting commands
- Attribute setting commands for object coordinate system
- Texture pattern management commands
- Binary pattern drawing commands
- BLT commands
- Execution control commands
- Drawing attributes restore command
- Video capture control commands
- $I^2C$ control commands
- Register control commands

# 4.1 System Command List

System commands are shown in the Table 4.1.

**Table 4.1 System Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcInitialize** | Initialization of the Graphics Driver | Y | Y | Y | Y | Y |
| 2 | **GdcSetDMAMode** | Sets DMA Mode | Y | Y | Y | N | N |
| 3 | **GdcSetInterruptMask** | Sets interrupt mask for MB86290A | Y | N | N | N | N |
| 4 | **GdcGeoSetInterruptMask** | Sets interrupt mask for MB86291 or later | N | Y | Y | Y | Y |
| 5 | **GdcGetInterruptStatus** | Gets interrupt status for MB86290A | Y | N | N | N | N |
| 6 | **GdcGeoGetInterruptStatus** | Gets interrupt status for MB86291 or later | N | Y | Y | Y | Y |
| 7 | **GdcClearInterruptStatus** | Clears interrupt request for MB86290A | Y | N | N | N | N |
| 8 | **GdcGeoClearInterruptStatus** | Clears interrupt request for MB86291 or later | N | Y | Y | Y | Y |
| 9 | **GdcGetFIFOStatus** | Gets status of display list FIFO | Y | Y | Y | Y | Y |
| 10 | **GdcGetFIFORemain** | Gets remains of display list FIFO | Y | Y | Y | Y | Y |
| 11 | **GdcGetFIFOErrorStatus** | Gets error status of display list FIFO | Y | Y | Y | Y | Y |
| 12 | **GdcGeoGetFIFOStatus** | Gets status of geometry display list FIFO | N | Y | Y | Y | Y |
| 13 | **GdcGeoGetFIFORemain** | Gets remains of geometry display list FIFO | N | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

(Continue)

**Table 4.1 System Command List** (Continued)

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 14 | **GdcGetPixelEngineStatus** | Gets status of pixel engine | Y | N | N | N | N |
| 15 | **GdcGeoGetPixelEngineStatus** | Gets status of geometry pixel engine | N | Y | Y | Y | Y |
| 16 | **GdcGetLocalDisplayListTransferStatus** | Gets status of local display list transfer | Y | Y | Y | Y | Y |
| 17 | **GdcQueryChipID** | Queries chip ID | N | N | Y | Y | Y |
| 18 | **GdcQueryGDCType** | Queries the Graphics Controller type | Y | Y | Y | Y | Y |
| 19 | **GdcQueryVersion** | Queries version number | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.2 Context Control Command List

Context control command is shown in the Table 4.2.

**Table 4.2 Context Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcCreateContext** | Creates context | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.3 Error Control Command List

Error control commands are shown in the Table 4.3.

**Table 4.3 Error Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291 /292 | 293 /294 | 295 | 296 |
| 1 | **XGdcGetErrCode** | Gets an error code | Y | Y | Y | Y | Y |
| 2 | **XGdcSetErrCode** | Sets an error code | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.4 Display Setting Command List

Display setting commands are shown in the Table 4.4.

**Table 4.4 Display Setting Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcDispClock** | Sets display clock mode | Y | Y | Y | Y | Y |
| 2 | **GdcDispTiming** | Sets display timing parameters | Y | Y | Y | Y | Y |
| 3 | **GdcDispTimingWindow** | Sets display position of layer W | Y | Y | Y | Y | Y |
| 4 | **GdcDispDividePos** | Sets border position of screen partition | Y | Y | Y | Y | Y |
| 5 | **GdcDispDimension** | Sets attributes of display frame | Y | Y | Y | Y | Y |
| 6 | **GdcDispOn** | Asserts video signal output | Y | Y | Y | Y | Y |
| 7 | **GdcDispOff** | Negates video signal output | Y | Y | Y | Y | Y |
| 8 | **GdcDispLayerOn** | Asserts screen display | Y | Y | Y | Y | Y |
| 9 | **GdcDispLayerOff** | Negates screen display | Y | Y | Y | Y | Y |
| 10 | **GdcDispPos** | Sets display start position | Y | Y | Y | Y | Y |
| 11 | **GdcDispDoFlip** | Flips display bank | Y | Y | Y | Y | Y |
| 12 | **GdcOverlayPriorityMode** | Sets overlay display mode | Y | Y | Y | Y | Y |
| 13 | **GdcOverlayBlend** | Sets blend parameter for overlay blend | Y | Y | Y | Y | Y |
| 14 | **GdcDispDisplayMode** | Sets display mode | N | N | Y | Y | Y |
| 15 | **GdcDispDisplayLayerMode** | Sets layer display mode | N | N | Y | Y | Y |
| 16 | **GdcDispSetBackColor** | Sets background color | N | N | Y | Y | Y |
| 17 | **GdcDispSetLayerWindow** | Sets position and size of the window mode layer | N | N | Y | Y | Y |
| 18 | **GdcLayerOverlayPriorityMode** | Sets overlay display mode in every layer | N | N | Y | Y | Y |
| 19 | **GdcLayerOverlayBlend** | Sets blend mode in every layer | N | N | Y | Y | Y |
| 20 | **GdcDispLayerOrder** | Sets layer display order | N | N | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

(Continue)

**Table 4.4 Display Setting Command List** (Continued)

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 21 | **GdcMultiDisplayMode** | Shifts to dual display mode | N | N | N | N | Y |
| 22 | **GdcSingleDisplayMode** | Shifts to single display mode | N | N | N | N | Y |
| 23 | **GdcMultiDispLayerOn** | Sets layer display for dual display mode | N | N | N | N | Y |
| 24 | **GdcMultiDispLayerOff** | Sets layer not-display for dual display mode | N | N | N | N | Y |
| 25 | **GdcMultiDispCursorOn** | Sets cursor display for dual display mode | N | N | N | N | Y |
| 26 | **GdcMultiDispCursorOff** | Sets cursor not-display for dual display mode | N | N | N | N | Y |
| 27 | **GdcDispSetYCMatrix** | Sets YCbCr to RGB transformation matrix | N | N | N | N | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.5 Color Setting Command List

Color setting commands are shown in the Table 4.5.

**Table 4.5 Color Setting Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291 /292 | 293/ 294 | 295 | 296 |
| 1 | **GdcColorPalette** | Sets palette colors | Y | Y | Y | Y | Y |
| 2 | **GdcColorTransparent** | Sets transparent color | Y | Y | Y | Y | Y |
| 3 | **GdcColorZeroMode** | Sets color code "0" as transparent mode | Y | Y | Y | Y | Y |
| 4 | **GdcChromaKeyMode** | Sets chroma-key mode | Y | Y | Y | Y | Y |
| 5 | **GdcColorKey** | Sets key color for chroma-key | Y | Y | Y | Y | Y |
| 6 | **GdcColorPaletteOffset** | Sets of the color palette offset | N | N | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.6 Cursor Control Command List

Cursor control commands are shown in the Table 4.6.

**Table 4.6 Cursor Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcCursorAddress** | Sets cursor pattern memory address | Y | Y | Y | Y | Y |
| 2 | **GdcCursorPattern** | Sets cursor pattern | Y | Y | Y | Y | Y |
| 3 | **GdcCursorDisplay** | Controls cursor display | Y | Y | Y | Y | Y |
| 4 | **GdcCursorPos** | Sets cursor display position | Y | Y | Y | Y | Y |
| 5 | **GdcCursorPriority** | Sets cursor display priority | Y | Y | Y | Y | Y |
| 6 | **GdcCursorColorTransparent** | Sets cursor transparent color | Y | Y | Y | Y | Y |
| 7 | **GdcCursorColorZeroMode** | Sets cursor color code "0" as transparent mode | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.7 Display List Control Command List

Display list control commands are shown in the Table 4.7.

**Table 4.7 Display List Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcSetDLBuf** | Sets current DL buffer | Y | Y | Y | Y | Y |
| 2 | **XGdcQueryCurrentDLBuf** | Queries current DL buffer | Y | Y | Y | Y | Y |
| 3 | **XGdcGetDLBufNum** | Gets the number of DL buffers | Y | Y | Y | Y | Y |
| 4 | **XGdcGetDLBufInfo** | Gets DL buffer structure information | Y | Y | Y | Y | Y |
| 5 | **XGdcFlush** | Transfers display list in current DL buffer | Y | Y | Y | Y | Y |
| 6 | **XGdcFlushEx** | Transfers display list in specified DL buffer | Y | Y | Y | Y | Y |
| 7 | **XGdcCancelDisplayList** | Cancels display list | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.8 Drawing Frame Setting Command List

Drawing frame setting commands are shown in the Table 4.8.

**Table 4.8 Drawing Frame Setting Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcDrawDimension** | Sets drawing frame | Y | Y | Y | Y | Y |
| 2 | **XGdcSetZPrecision** | Sets precision of z value | N | N | Y | Y | Y |
| 3 | **XGdcBufferCreateZ** | Sets Z-buffer base address | Y | Y | Y | Y | Y |
| 4 | **XGdcBufferCreateC** | Sets base address of polygon drawing control buffer | Y | Y | Y | Y | Y |
| 5 | **XGdcBufferClearZ** | Clears Z-buffer | Y | Y | Y | Y | Y |
| 6 | **XGdcBufferClearC** | Clears polygon drawing control buffer | Y | Y | Y | Y | Y |
| 7 | **XGdcDrawClipFrame** | Sets drawing clip border | Y | Y | Y | Y | Y |
| 8 | **XGdcSetAlphaMapBase** | Sets base address of alpha map area | N | N | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

## 4.9 Primitive Drawing Command List for Device Coordinate System

Primitive drawing commands for device coordinate system are shown in the Table 4.9.

**Table 4.9 Primitive Drawing Command List for Device Coordinate System**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcPrimType** | Starts drawing procedure | Y | Y | Y | Y | Y |
| 2 | **XGdcPrimEnd** | Completes drawing procedure | Y | Y | Y | Y | Y |
| 3 | **XGdcTexCoord2D** | Sets coordinates of 2D texture (GDC_FIXED32 type) | Y | Y | Y | Y | Y |
| 4 | **XGdcTexCoord2Df** | Sets coordinates of 2D texture (GDC_SFLOAT type) | Y | Y | Y | Y | Y |
| 5 | **XGdcTexCoord2DNf** | Sets normalized coordinates of 2D texture (GDC_SFLOAT type) | Y | Y | Y | Y | Y |
| 6 | **XGdcTexCoord3D** | Sets coordinates of 3D texture (GDC_FIXED32 type) | Y | Y | Y | Y | Y |
| 7 | **XGdcTexCoord3Df** | Sets coordinates of 3D texture (GDC_SFLOAT type) | Y | Y | Y | Y | Y |
| 8 | **XGdcTexCoord3DNf** | Sets normalized coordinates of 3D texture (GDC_SFLOAT type) | Y | Y | Y | Y | Y |
| 9 | **XGdcDrawVertex2D** | Sets coordinates of 2D vertex (GDC_FIXED32 type) | Y | Y | Y | Y | Y |
| 10 | **XGdcDrawVertex2Di** | Sets coordinates of 2D vertex (GDC_LONG type) | Y | Y | Y | Y | Y |
| 11 | **XGdcDrawVertex3D** | Sets coordinates of 3D vertex (GDC_FIXED32 type) | Y | Y | Y | Y | Y |
| 12 | **XGdcDrawVertex3Df** | Sets coordinates of 3D vertex (GDC_SFLOAT type) | Y | Y | Y | Y | Y |
| 13 | **XGdcDrawPrimitive** | Draws multiple 3D triangles | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.10 Primitive Drawing Command List for Object Coordinate System

Primitive drawing commands for object coordinate system are shown in the Table 4.10.

**Table 4.10 Primitive Drawing Command List for Object Coordinate System**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcGeoPrimType** | Starts drawing procedure | N | Y | Y | Y | Y |
| 2 | **XGdcGeoPrimEnd** | Completes drawing procedure | N | Y | Y | Y | Y |
| 3 | **XGdcGeoDrawVertex2D** | Sets XY coordinates of vertex (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 4 | **XGdcGeoDrawVertex2Df** | Sets XY coordinates of vertex (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 5 | **XGdcGeoDrawVertex2Di** | Sets XY coordinates of vertex (GDC_LONG type) | N | Y | Y | Y | Y |
| 6 | **XGdcGeoDrawVertex3D** | Sets XYZ coordinates of vertex (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 7 | **XGdcGeoDrawVertex3Df** | Sets XYZ coordinates of vertex (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 8 | **XGdcGeoDrawVertex3Di** | Sets XYZ coordinates of vertex (GDC_LONG type) | N | Y | Y | Y | Y |
| 9 | **XGdcGeoTexCoord2DN** | Sets texture coordinates (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 10 | **XGdcGeoTexCoord2DNf** | Sets texture coordinates (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 11 | **XGdcVertexColor32** | Sets color of vertex (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 12 | **XGdcVertexColor3f** | Sets color of vertex (GDC_COLOR32 type) | N | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.11 Drawing Attribute Setting Command List

Drawing attribute setting commands are shown in the Table4.11.

**Table 4.11 Drawing Attribute Setting Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcColor** | Sets vertex color/foreground color | Y | Y | Y | Y | Y |
| 2 | **XGdcBackColor** | Sets background color | Y | Y | Y | Y | Y |
| 3 | **XGdcClipMode** | Sets clipping mode | Y | Y | Y | Y | Y |
| 4 | **XGdcSetAttrLine** | Sets line drawing attribute | Y | Y | Y | Y | Y |
| 5 | **XGdcSetAttrSurf** | Sets surface drawing attribute | Y | Y | Y | Y | Y |
| 6 | **XGdcSetAttrTexture** | Sets texture mapping attribute | Y | Y | Y | Y | Y |
| 7 | **XGdcSetAttrBlt** | Sets BitBlt attribute | Y | Y | Y | Y | Y |
| 8 | **XGdcSetAlpha** | Sets alpha blending coefficient | Y | Y | Y | Y | Y |
| 9 | **XGdcSetLinePattern** | Sets broken line pattern | Y | Y | Y | Y | Y |
| 10 | **XGdcSetTextureBorder** | Sets texture border color | Y | Y | Y | Y | Y |
| 11 | **XGdcSetRop** | Sets logical operation mode | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.12 Attribute Setting Command List for Object Coordinate System

Attribute setting commands for object coordinate system are shown in the Table4.12.

**Table 4.12 Attribute Setting Command List for Object Coordinate System**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcGeoSetAttrLine** | Sets line drawing attribute for object coordinate system | N | N | Y | Y | Y |
| 2 | **XGdcGeoSetAttrSurf** | Sets surface drawing attribute for object coordinate system | N | Y | Y | Y | Y |
| 3 | **XGdcGeoLoadMatrix** | Sets matrix (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 4 | **XGdcGeoLoadMatrixf** | Sets matrix (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 5 | **XGdcGeoNdcDcViewportCoef** | Sets coefficients of NdcDc transformation for xy (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 6 | **XGdcGeoNdcDcViewportCoeff** | Sets coefficients of NdcDc transformation for xy (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 7 | **XGdcGeoNdcDcDepthCoef** | Sets coefficients of NdcDc transformation for z (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 8 | **XGdcGeoNdcDcDepthCoeff** | Sets coefficients of NdcDc transformation for z (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 9 | **XGdcGeoViewVolumeXYClip** | Sets view volume boundary for xy (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 10 | **XGdcGeoViewVolumeXYClipf** | Sets view volume boundary for xy (GDC_SFLOAT type) | N | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

(Continue)

**Table 4.12 Attribute Setting Command List for Object Coordinate System** (Continued)

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 11 | **XGdcGeoViewVolumeZClip** | Sets view volume boundary for z (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 12 | **XGdcGeoViewVolumeZClipf** | Sets view volume boundary for z (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 13 | **XGdcGeoViewVolumeWminClip** | Sets view volume boundary for w (GDC_FIXED32 type) | N | Y | Y | Y | Y |
| 14 | **XGdcGeoViewVolumeWminClipf** | Sets view volume boundary for w (GDC_SFLOAT type) | N | Y | Y | Y | Y |
| 15 | **XGdcGeoSetLogOutBase** | Sets base address for log output of device coordinates | N | N | Y | Y | Y |
| 16 | **XGdcGeoSetLogOutMode** | Sets log output mode of the device coordinates | N | N | Y | Y | Y |
| 17 | **XGdcGeoShadowXY** | Sets xy offset of shadow | N | N | Y | Y | Y |
| 18 | **XGdcGeoOverlapZ** | Sets z value of primitives (body / shadow / border / correction in top-left rule non-applied mode) | N | N | Y | Y | Y |
| 19 | **XGdcGeoShadowColor** | Sets color of shadow | N | N | Y | Y | Y |
| 20 | **XGdcGeoShadowBackColor** | Sets background color of shadow | N | N | Y | Y | Y |
| 21 | **XGdcGeoBorderColor** | Sets color of border | N | N | Y | Y | Y |
| 22 | **XGdcGeoBorderBackColor** | Sets background color of border | N | N | Y | Y | Y |
| 23 | **XGdcGeoSetupMode** | Sets setup mode | N | N | N | N | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.13 Texture Image Management Command List

Texture image management commands are shown in the Table 4.13.

**Table 4.13 Texture Image Management Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcTextureMemoryMode** | Sets texture memory mode | Y | Y | N | N | N |
| 2 | **XGdcTextureLoadInt8** | Loads image data to the internal texture memory (8bpp) | Y | Y | N | N | N |
| 3 | **XGdcTextureLoadInt16** | Loads image data to the internal texture memory (16bpp) | Y | Y | N | N | N |
| 4 | **XGdcTextureLoadExt8** | Loads image data to the graphics memory (8bpp) | N | N | Y | Y | Y |
| 5 | **XGdcTextureLoadExt16** | Loads image data to the graphics memory (16bpp) | Y | Y | Y | Y | Y |
| 6 | **XGdcTextureLoadExt16Fast** | Loads image data to the graphics memory for bi-linear fast mode | N | N | Y | N | Y |
| 7 | **XGdcTextureDimension** | Sets texture/tile information | Y | Y | Y | Y | Y |
| 8 | **XGdcBltTexture** | Loads BitBlt texture from the graphics memory to the internal texture memory | Y | Y | N | N | N |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.14 Binary Pattern Drawing Command List

Binary pattern drawing commands are shown in the Table 4.14.

**Table 4.14 Binary Pattern Drawing Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcBitPatternDraw** | Draws binary pattern (no clipping) | Y | Y | Y | Y | Y |
| 2 | **XGdcBitPatternDrawByte** | Draws binary pattern (clipping) | Y | Y | Y | Y | Y |
| 3 | **XGdcBitPatternMode** | Sets enlargement/reduction mode | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.15 BLT Command List

BLT commands are shown in the Table 4.15.

**Table 4.15 BLT Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcBltCopy** | Copies BitBlt area in current drawing frame | Y | Y | Y | Y | Y |
| 2 | **XGdcBltCopyAlt** | Copies BitBlt area between arbitrary drawing frames (Async) | Y | Y | Y | Y | Y |
| 3 | **XGdcBltCopyAltSync** | Copies BitBlt area between arbitrary drawing frames (Sync) | Y | Y | Y | Y | Y |
| 4 | **XGdcBltDraw8** | Copies BitBlt area from main memory to current drawing frame (8bpp) | Y | Y | Y | Y | Y |
| 5 | **XGdcBltDraw16** | Copies BitBlt area from main memory to current drawing frame (16bpp) | Y | Y | Y | Y | Y |
| 6 | **XGdcBltFill** | Fills BitBlt area | Y | Y | Y | Y | Y |
| 7 | **XGdcBltColorTransparent** | Sets transparent color at copying BitBlt area | N | Y | Y | Y | Y |
| 8 | **XGdcBltCopyAltAlpha** | Copies BitBlt area with alpha blending | N | N | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.16 Execution Control Command List

Execution control commands are shown in the Table 4.16.

**Table 4.16 Execution Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcVerticalSync** | Generates Sync command | Y | Y | Y | Y | Y |
| 2 | **XGdcInterrupt** | Generates Interrupt command | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.17 Drawing Attribute Restore Command List

Drawing attribute restore command is shown in the Table 4.17.

**Table 4.17 Drawing Attribute Restore Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **XGdcRestoreAttr** | Restores drawing attributes | Y | Y | Y | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 4.18 Video Capture Control Command List

Video capture control commands are shown in the Table 4.18.

**Table 4.18 Video Capture Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcCapSetVideoCaptureMode** | Sets mode of video capture | N | Y | E | Y | Y |
| 2 | **GdcCapGetErrorStatus** | Gets error status of video capture | N | Y | E | Y | Y |
| 3 | **GdcCapClearErrorStatus** | Clears error status of video capture | N | Y | E | Y | Y |
| 4 | **GdcCapSetVideoCaptureBuffer** | Sets video capture buffer | N | Y | E | Y | Y |
| 5 | **GdcCapSetImageArea** | Sets range of image | N | Y | E | Y | Y |
| 6 | **GdcCapGetImageAddress** | Gets address of captured image | N | N | E | Y | Y |
| 7 | **GdcCapSetWindowMode** | Sets layer L1 (W) mode | N | Y | E | Y | Y |
| 8 | **GdcCapSetVideoCaptureScale** | Sets scale of video capture | N | Y | E | Y | Y |
| 9 | **GdcCapSetAttrMisc** | Sets attribute of video capture | N | Y | E | Y | Y |
| 10 | **GdcCapSetInputDataCountNTSC** | Sets number of video capture data for NTSC | N | Y | E | Y | Y |
| 11 | **GdcCapSetInputDataCountPAL** | Sets number of video capture data for PAL | N | Y | E | Y | Y |
| 12 | **GdcCapSetLPFMode** | Sets low pass filter mode | N | Y | E | Y | Y |
| 13 | **GdcCapSetAttrVideo** | Sets various modes of video capture | N | Y | E | Y | Y |
| 14 | **GdcCapStartVideoCapture** | Starts capturing video data | N | Y | E | Y | Y |
| 15 | **GdcCapStopVideoCapture** | Stops capturing video data | N | Y | E | Y | Y |
| 16 | **GdcCapSetCaptureArea** | Sets area of capturing video data | N | Y | E | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

E: exception, only the MB86294/86294S can use it

(Continue)

**Table 4.18 Video Capture Control Command List** (Continued)

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 17 | **GdcCapSetMaxHorizontalPixel** | Sets maximum, horizontal pixels of input images | N | N | E | Y | Y |
| 18 | **GdcCapSetMaxVerticalPixel** | Sets maximum, vertical pixels of input images | N | N | N | N | Y |
| 19 | **GdcCapSetRGBInputTiming** | Sets range of RGB input | N | N | N | Y | Y |
| 20 | **GdcCapSetRGBInputSync** | Sets RGB input synchronous signal | N | N | N | Y | Y |
| 21 | **GdcCapSetRGBMatrix** | Sets RGB to YCbCr transformation matrix | N | N | N | Y | Y |
| 22 | **GdcCapStartClock** | Starts video capture clock supply | N | N | N | N | Y |
| 23 | **GdcCapStopClock** | Stops video capture clock supply | N | N | N | N | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

E: exception, only the MB86294/86294S can use it

# 4.19 I²C Control Command List

I²C control commands are shown in the Table 4.19.

**Table 4.19 I²C Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcI2CGetBusStatus** | Gets I²C bus status | N | E | E | E | E |
| 2 | **GdcI2CSetBusControl** | Controls I²C bus | N | E | E | E | E |
| 3 | **GdcI2CGetBusControlStatus** | Gets I²C bus control status | N | E | E | E | E |
| 4 | **GdcI2CSetClock** | Sets I²C clock | N | E | E | E | E |
| 5 | **GdcI2CGetClock** | Gets I²C clock control status | N | E | E | E | E |
| 6 | **GdcI2CSetData** | Sets transfer data | N | E | E | E | E |
| 7 | **GdcI2CGetData** | Gets transfer data | N | E | E | E | E |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

E: exception, only the MB86291S, MB86291AS, MB86292S, MB86294S, MB86295S and MB86296S can use it

# 4.20 Register Control Command List

Register control commands are shown in the Table 4.20.

**Table 4.20 Register Control Command List**

| No. | Command Name | Function | Graphics Controller (*1) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 290A | 291/ 292 | 293/ 294 | 295 | 296 |
| 1 | **GdcWriteHostRegister** | Writes host interface register | N | N | N | Y | Y |
| 2 | **GdcReadHostRegister** | Reads host interface register | N | N | N | Y | Y |

(*1) 290A:MB86290A, 291/292:MB86291/86292, 293/294:MB86293/86294, 295:MB86295S, 296:MB86296S

Y: can be used

N: can not be used

# 5 Data Types

This section describes the data types and data structures specified by the Graphics Driver.

## 5.1 Data Type List

Data types defined in the Graphics Driver are shown in the Table 5.1.

**Table 5.1 Data Type List**

| Format | Description |
|---|---|
| GDC_UCHAR | 8 bits signed integer |
| GDC_SHORT | 16 bits signed integer |
| GDC_USHORT | 16 bits unsigned integer |
| GDC_LONG | 32 bits signed integer |
| GDC_ULONG | 32 bits unsigned integer |
| GDC_SFLOAT | 32 bits single precision float (IEEE754 compliant) |
| GDC_BOOL | True/false (value of **GDC_TRUE/GDC_FALSE**) |
| GDC_FIXED32 | 32 bits signed fixed point<br>(1 bit sign, 15 bits integer and 16 bits fraction) |
| GDC_FIXED_SCALE | 16 bits unsigned fixed point for Capture Scale<br>(5 bits integer and 11 bits fraction) |
| GDC_COLOR32 | 32 bits unsigned integer (32-bit color format) |
| GDC_COL32 | 32 bits unsigned integer (palette color format) |
| GDC_COL24 | 32 bits unsigned integer (24-bit color format) |
| GDC_COL16 | 16 bits unsigned integer (16-bit color format) |
| GDC_COL8 | 8 bits unsigned integer (8-bit color format) |
| GDC_BINIMAGE | 32 bits unsigned integer data (binary pattern data) |
| GDC_VERTEX | Vertex data structure |
| GDC_INITPARAM | Initialize parameter table |
| GDC_DLBUF_STRUCT | Information of DL buffer structure |
| GDC_CTX | Context |

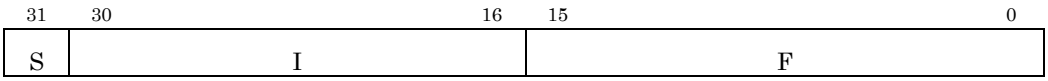
 

# 5.2 Data Formats

Data formats defined in the Graphics Driver are shown in the followings.

## 5.2.1 GDC_FIXED32 [32 Bits Fixed Point]

The 32 bits fixed point format is a format expressed by 1 bit in the sign, 15 bits in the integer, and 16 bits in the fraction.




S: Sign (1 bit)

0:Positive number or zero

1:Negative number

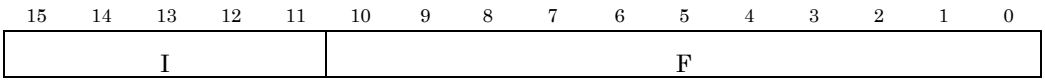
I: Integer (15 bits)

F: Fraction (16 bits)

**Figure 5.2.1 GDC_FIXED32 Format**

## 5.2.2 GDC_FIXED_SCALE [Capture Scale]

The capture scale format is a format expressed by 5 bits in the integer and 11 bits in the fraction. It used by the **GdcCapSetVideoCaptureScale** command.




I: Integer (5 bits)

F: Fraction (11 bits)

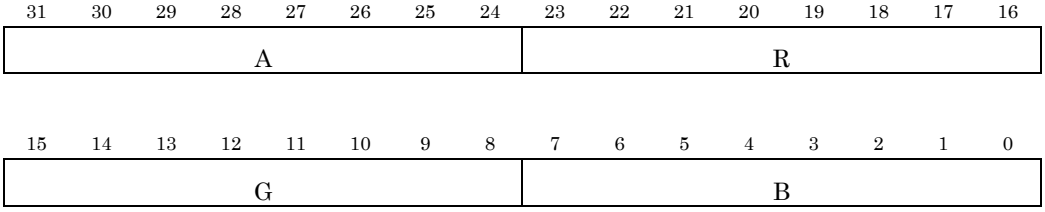
**Figure 5.2.2 GDC_FIXED_SCALE Format**

### 5.2.3 GDC_COLOR32 [32-bit Color]

The 32-bit color format is a format that expresses the color value by 32 bits. The used bit is different according to the Driver Command and the color mode. Refer to the description of each Driver Command for details. Explains the example when the color value is specified by the **XGdcVertexColor32** command as follows.

[Specify it by the **XGdcVertexColor32** command at 16-bit color mode]

The color data is expressed by the ARGB format.




A: Alpha bit (*1) (8 bits)

Sets blending coefficient of vertex. (0 to 255)

(*1) An alpha bit here is the blend coefficient referred to when using the alpha shading function at triangle drawing.

R, G, B:

Color bit (8 bits)

Sets vertex color, each value range is from "0" to "255".

**Figure 5.2.3a GDC_COLOR32 Format when specifying it by the XGdcVertexColor32 Command at 16-bit Color Mode**

[Specify it by the **XGdcVertexColor32** command at 8-bit color mode]

The color data is expressed by 8 bits of 16-23 bits.


color: 8 bits color data (8 bits)

Sets 8 bits vertex color. (0 to 255)

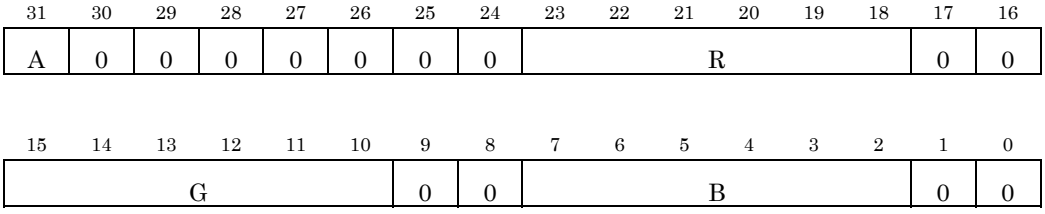
All other bits: 0

**Figure 5.2.3b GDC_COLOR32 Format when specifying it by the XGdcVertexColor32 Command at 8-bit Color Mode**

### 5.2.4 GDC_COL32 [Color for Color Palette]

The color for color palette format is a format that expresses the color value in RGB 6 bits per respectively. For layer C palette, bit 31 is an alpha bit.




A: Alpha bit (1 bit)

When blend mode is available, sets mode of blend

0: Not blending

1: Blending

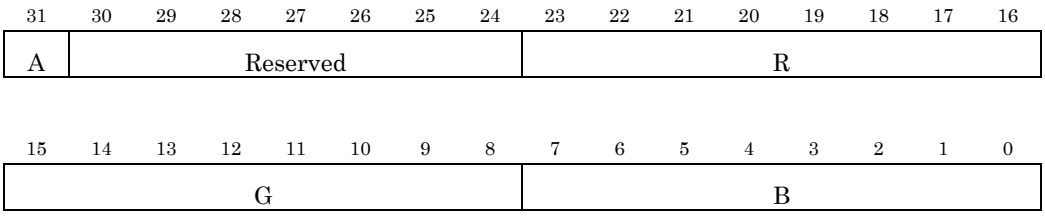
R, G, B:

Color bit (6 bits)

**Figure 5.2.4 GDC_COL32 Format**

### 5.2.5 GDC_COL24 [24-bit Color]

The 24-bit color format is a format that expresses the color value in RGB 8 bits per respectively. When this color data format is applied to texture, bit 31 is used as an alpha bit.




A: Alpha bit (1 bit)

When blend mode is available, sets mode of blend or stencil processing

0: Not blending or stencil processing

1: Blending or stencil processing

Reserved: Not in use

R, G, B:

Color bit (8 bits)

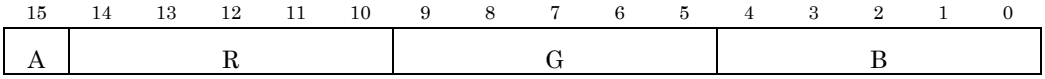
Sets image data's color, each value range is from "0" to "255".

**Figure 5.2.5 GDC_COL24 Format**

### 5.2.6 GDC_COL16 [16-bit Color]

The 16-bit color format is a format that expresses the color value in RGB 5 bits per respectively. When this color data format is applied to texture, bit 15 is used as an alpha bit.




A: Alpha bit (1 bit)

When blend mode is available, sets mode of blend or stencil processing

0: Neither blending nor stencil processing

1: Blending or stencil processing

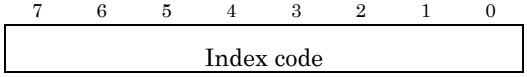
R, G, B:

Color bit (5 bits)

**Figure 5.2.6 GDC_COL16 Format**

### 5.2.7 GDC_COL8 [8-bit Color]

The 8-bit color format is a format that the color value is expressed by the index code in 8 bits.




Index code: Index code (8 bits)

Specify the index code in range from "0" to "255" in the case of referring to a color palette.

**Figure 5.2.7 GDC_COL8 Format**

### 5.2.8 GDC_VERTEX [Vertex Data Structure]

The vertex data structure is packed with vertex coordinates, texture coordinates and RGB values. It used by the **XGdcDrawPrimitive** command.

GDC_VERTEX structure is shown in the following.

[GDC_VERTEX structure]

```
typedef struct {
    GDC_SFLOAT      x, y, z;    /* x,y,z coordinates of vertex for device coordinates (unit: pixel) */
    GDC_SFLOAT      r, g, b;    /* r,g,b values of vertex color (normalized to [0,255]) */
    GDC_SFLOAT      u,v;        /* u,v texture coordinates of vertex */
    GDC_SFLOAT      rw;         /* Reciprocal w texture coordinates of vertex */
    long            work;       /* Reserved */
} GDC_VERTEX;
```

### 5.2.9 GDC_INITPARAM [Initialize Parameter Table]

An initialize parameter table is the table for specifying the various parameters used by initialization processing of the Graphics Driver, and is used with the **GdcInitialize** command.

The contents of a setting of the data structure of the initialize parameter table and each parameter are shown in the following.

[GDC_INITPARAM structure]

```
typedef struct {
    GDC_ULONG       cputype;        /* CPU type                          */
    GDC_ULONG       gdctype;        /* Graphics Controller type          */
    GDC_ULONG       gdcbase;        /* Graphics Controller base address  */
    GDC_ULONG       geoclock;       /* Geometry engine clock             */
    GDC_ULONG       otherclock;     /* Other clock                       */
    GDC_ULONG       locate;         /* Register location                 */
    GDC_ULONG       memorymode;     /* Memory interface mode             */
    struct {
        GDC_ULONG   mode;           /* Display clock mode                */
        GDC_ULONG   htp;            /* Horizontal pixel                  */
        GDC_ULONG   hsp;            /* Horizontal pulse position         */
        GDC_ULONG   hsw;            /* Horizontal sync width             */
        GDC_ULONG   hdp;            /* Horizontal display pixel          */
        GDC_ULONG   vtr;            /* Vertical total raster             */
        GDC_ULONG   vsp;            /* Vertical sync pulse position      */
        GDC_ULONG   vsw;            /* Vertical sync pulse width         */
        GDC_ULONG   vdp;            /* Vertical display raster           */
        GDC_ULONG   clockmode;      /* Digital display output mode       */
    } disp;
} GDC_INITPARAM;
```

[Content of each parameter setting]

cputype

CPU to be used is specified on one of the following macros.

| [Macro] | [Meaning] |
|---|---|
| **GDC_CPU_SH** | SH-3 or SH-4 |
| **GDC_CPU_WINDOWS** | PC |
| **GDC_CPU_V832** | V832 |
| **GDC_CPU_OTHER** | Other |

gdctype

The Graphics Controller to be used is specified on one of the following macros.

| [Macro] | [Meaning] |
|---|---|
| **GDC_TYPE_MB86290A** | MB86290A |
| **GDC_TYPE_MB86291** | MB86291 or MB86291S |
| **GDC_TYPE_MB86291A** | MB86291A or MB86291AS |
| **GDC_TYPE_MB86292** | MB86292 or MB86292S |
| **GDC_TYPE_MB86293** | MB86293 |
| **GDC_TYPE_MB86294** | MB86294 or MB86294S |
| **GDC_TYPE_MB86295** | MB86295S |
| **GDC_TYPE_MB86296** | MB86296S |

gdcbase

The address in which the Graphics Controller is mapped is specified.

Specify the return value of the **Get_FrameAddress** command of the map driver in PC.

geoclock (*1)

Specify the clock of geometry engine by the one of the followings.

| [Macro] | [Meaning] |
|---|---|
| **GDC_CLOCK_166MHZ** | 166MHz |
| **GDC_CLOCK_133MHZ** | 133MHz |
| **GDC_CLOCK_100MHZ** | 100MHz |

The above parameter is used only when one of the followings is specified as gdctype, and other cases are disregarded.

**GDC_TYPE_MB86293**

**GDC_TYPE_MB86294**

**GDC_TYPE_MB86295**

**GDC_TYPE_MB86296**

(*1) Combine geoclock and otherclock according to Table 5.2.9.

**Table 5.2.9 Internal Operation Frequency Combination**

| | | geoclock | | |
|---|---|---|---|---|
| | | 166MHz | 133MHz | 100MHz |
| otherclock | 133MHz | Y | Y | N |
| | 100MHz | Y | Y | Y |

Y: Can be combined

N: Cannot be combined (Not supported)

otherclock (*1)

Specify clock of operation other than geometry engine, such as pixel engine, by following either.

| [Macro] | [Meaning] |
|---|---|
| **GDC_CLOCK_133MHZ** | 133MHz |
| **GDC_CLOCK_100MHZ** | 100MHz |

The above parameter is used only when one of the followings is specified as gdctype, and other cases are disregarded.

**GDC_TYPE_MB86293**

**GDC_TYPE_MB86294**

**GDC_TYPE_MB86295**

**GDC_TYPE_MB86296**

locate

Specify the mapping location of the register area of the Graphics Controller by following either.

| [Macro] | [Meaning] |
|---|---|
| **GDC_REG_LOCATE_CENTER** | Locate from H'01FC0000 |
| **GDC_REG_LOCATE_BOTTOM** | Locate from H'03FC0000 |

The above parameter is used only when one of the followings is specified as gdctype. Other cases are disregarded.

**GDC_TYPE_MB86293**

**GDC_TYPE_MB86294**

**GDC_TYPE_MB86295**

**GDC_TYPE_MB86296**

memorymode

A memory interface mode value (setting value to MMR register) is specified.

Refer to the hardware specifications of the Graphics Controller of use about MMR register.

disp

The parameter about the display is specified as follows.

| [Member variable] | [Meaning] |
|---|---|
| mode | Display clock mode (Setting value to DCM or DCEM register) (*2) |
| htp | Horizontal total pixel |
| hsp | Horizontal sync pulse position |
| hsw | Horizontal sync pulse width |
| hdp | Horizontal display pixel |
| vtr | Vertical total raster |
| vsp | Vertical sync pulse position |
| vsw | Vertical sync pulse width |
| vdp | Vertical display raster |
| clockmode | Digital display output mode |

clockmode is used only when **GDC_TYPE_MB86296** is specified for gdctype, and other cases are disregarded.

Please set the value (registers image) set to DCM3 register (*2) to clockmode. Please specify 0x0 for clockmode when outputting it only to one display. Either of the following macros can be specified for clockmode.

| [Macro] | [Meaning] |
|---|---|
| **GDC_MULTIPLEX_OUTPUT_MODE** | Multiplex output mode |
| **GDC_PARALLEL_OUTPUT_MODE** | Parallel output mode |

(*2) Refer to the hardware specifications of the Graphics Controller of use about DCM, DCEM and DCM3 register.

### 5.2.10 GDC_DLBUF_STRUCT [DL Buffer Structure Information]

DL buffer structure information is used with the **XGdcCreateContext** command in order to register the structure of DL buffer into context.

The contents of a setting of the data structure of DL buffer structure information and each parameter are shown below.

[GDC_DLBUF_STRUCT structure]

```
typedef struct{
    GDC_ULONG       *top;           /* Top address of DL buffer              */
    GDC_ULONG       *bottom;        /* Bottom address of DL buffer + 1       */
    GDC_ULONG       *cur;           /* Display list writing address */
    GDC_ULONG       ext_data;       /* Extend data area                      */
} GDC_DLBUF_STRUCT;
```

[Content of each parameter setting]

top

The top address of DL buffer is specified.

bottom

The address of the last address +1 of DL buffer is specified.

cur

The address in which a display list is written next is set up by the Driver Command.

For this reason, do not change the value of this area by application programs.

ext_data

Set up the value of this area by application programs if needed. Since the Graphics Driver does not perform reference or setup of this area, user can use this area freely.

### 5.2.11 GDC_CTX [Context]

The context is the area where the execution state and drawing attribute of the Graphics Driver are stored, and allocates an area by the application program side. The **XGdcCreateContext** command generates a context. Since the command for system control, the command for drawing control, and the command for display list control use a context, do not change the value of this area by application programs (since the Graphics Driver manages this area, explanation of the data structure is omitted).

# 6 Driver Command Reference

This chapter explains the call interface of the Driver Command
Each item of the following Driver Command references is as follows.

**Interface**

Command interface

**Arguments**

Description of arguments
The "default" in explanation expresses the value set up at the initialization of the Graphics Driver.

**Return value**

The return values and the description of them.
There is a function that returns the return value only **GDC_TRUE**.

**Error code**

Error code and error name when the command terminates abnormally (When the return value returns by **GDC_FALSE**).
This item is omitted if the command has no return value.
There is a command in which the error code is not defined even when the return value is defined in the command in preparation for the function enhancing in the future.

**Target GDC**

Target Graphics Controllers of the command

**Description**

Description of the command

**Notes**

Notes
This item omitted if there are no notes.

# 6.1 System Commands

## 6.1.1 GdcInitialize [Initialization of Graphics Driver]

**Interface**

GDC_BOOL **GdcInitialize** (const GDC_INITPARAM **initparam*, int *flag*)

**Arguments**

| | | |
|---|---|---|
| *initparam* | Pointer to Initialize Parameter Table | |
| *flag* | Specify any of the following | |
| | **GDC_INIT_START** | Executes initialization |
| | **GDC_INIT_RESET** | Executes software reset |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_CPU_TYPE** | Invalid CPU type is specified |
| **GDC_ERR_INVALID_GDC_TYPE** | Invalid Graphics Controller is specified |
| **GDC_ERR_INVALID_FLAG** | Value of *flag* is invalid |

**Target GDC**

All

**Description**

Initializes the Graphics Controller.

After the system starts, call this command only once first before a calling other Driver Command. And at this time, specify **GDC_INIT_START** for *flag*.

Also specify address of the Initialize Parameter Table for *initparam*. It is necessary to allocate memory space for the Initialize Parameter Table and to set each parameter before the calling this command. Refer to 5.2.9 GDC_INITPARAM [Initialize Parameter Table] for specification of the initialize parameter table.

To work software reset, specify **GDC_INIT_RESET** for *flag*.

### 6.1.2 GdcSetDMAMode [Sets DMA Mode]

**Interface**

GDC_BOOL **GdcSetDMAMode** (int *tran_unit*, int *dma_request*,
int *address_mode*, int *ack_mode*)

**Arguments**

*tran_unit* Unit of DMA transfer.

**GDC_DMA_TRANUNIT_4** 4-byte

**GDC_DMA_TRANUNIT_32** 32-byte

*dma_request* DMA request. Specify any of the followings.

**GDC_DMA_REQUEST_NEGATE**

During transferring, when the Graphics Controller cannot receive data, DMA request is invalid (negate), and if it will be in the state where data is receivable, it will be valid (assert)

**GDC_DMA_REQUEST_NO_NEGATE**

Not negate while DMA is transferring display list

*address_mode* Address mode of external DMA request. Specify any of the followings.

**GDC_DMA_ADDRMODE_DUAL** Dual address mode

**GDC_DMA_ADDRMODE_SINGLE** Single address mode

*ack_mode* ACK mode. Specify any of the followings.

**GDC_DMA_ACKMODE**

Uses ACK (detect DMA request at a low level signal)

**GDC_DMA_NO_ACKMODE**

Not use ACK (detect DMA request at an edge)

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_INVALID_PARAMETER** Invalid parameter has been specified

**Target GDC**

MB86290A/86291/86291A/86292/86293/86294

### Description

Sets DMA transfer mode of DSU (DMA Set Up) register.

*ack_mode* is available only for the MB86293/86294. In MB86290A/291/292, Specifies **GDC_DMA_ACKMODE** (uses ACK) for *ack_mode*.

### 6.1.3 GdcSetInterruptMask [Sets Interrupt Mask for MB86290A]

#### Interface

void **GdcSetInterruptMask** (GDC_UCHAR *mask*)

#### Arguments

*mask* Mask pattern (shown below)

| bit 7 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 0 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Mask=1, Enable=0
bit 1: Command complete interrupt Mask=1, Enable=0
bit 2: VSYNC interrupt Mask=1, Enable=0
bit 3: Frame sync interrupt Mask=1, Enable=0
bit 4: External sync error interrupt Mask=1, Enable=0
All other bits: 0

**Figure 6.1.3 Mask Pattern Format**

#### Return value

None

#### Target GDC

MB86290A

#### Description

Sets interrupt mask pattern of IMASK (Interrupt MASK) register to disable interrupt requests generated by the respective events.

#### Notes

In MB86291 or later, use the **GdcGeoSetInterruptMask** command.

### 6.1.4 GdcGeoSetInterruptMask [Sets Interrupt Mask for MB86291 or Later]

#### Interface

void **GdcGeoSetInterruptMask** (GDC_ULONG *mask*)

#### Arguments

*mask* Mask pattern (shown below)

| bit 31 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 0 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Mask=0, Enable=1
bit 1: Command complete interrupt Mask=0, Enable=1
bit 2: VSYNC interrupt Mask=0, Enable=1
bit 3: Frame sync interrupt Mask=0, Enable=1
bit 4: External sync error interrupt Mask=0, Enable=1
All other bits: 0

**Figure 6.1.4 Mask Pattern Format**

#### Return value

None

#### Target GDC

MB86291 or later

#### Description

Sets interrupt mask pattern of IMASK (Interrupt MASK) to disable interrupt requests generated by the respective events.

#### Notes

In MB86290A, use the **GdcSetInterruptMask** command.

### 6.1.5 GdcGetInterruptStatus [Gets Interrupt Status for MB86290A]

#### Interface

GDC_UCHAR **GdcGetInterruptStatus** (void)

#### Arguments

None

#### Return value

Interrupts status (IST register value) in the following format:

| bit 7 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 0 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Occur=1, None=0
bit 1: Command complete interrupt Occur=1, None=0
bit 2: VSYNC interrupt Occur=1, None=0
bit 3: Frame sync interrupt Occur=1, None=0
bit 4: External sync error interrupt Occur=1, None=0
All other bits: 0

**Figure 6.1.5 Interrupt Status**

#### Target GDC

MB86290A

#### Description

Reads IST (Interrupt Status) register and returns interrupt status.

#### Notes

In MB86291 or later, use the **GdcGeoGetInterruptStatus** command.

### 6.1.6 GdcGeoGetInterruptStatus [Gets Interrupt Status for MB86291 or Later]

#### Interface

GDC_ULONG **GdcGeoGetInterruptStatus** (void)

#### Arguments

None

#### Return value

Interrupts status (IST register value) in the following format:

| bit 31 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 0 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Occur=1, None=0
bit 1: Command complete interrupt Occur=1, None=0
bit 2: VSYNC interrupt Occur=1, None=0
bit 3: Frame sync interrupt Occur=1, None=0
bit 4: External sync error interrupt Occur=1, None=0
All other bits: 0

**Figure 6.1.6 Interrupt Status**

#### Target GDC

MB86291 or later

#### Description

Reads IST (Interrupt Status) register and returns interrupt status.

#### Notes

In MB86290A, use the **GdcGetInterruptStatus** command.

### 6.1.7 GdcClearInterruptStatus [Clears Interrupt Request for MB86290A]

#### Interface

void **GdcClearInterruptStatus** (GDC_UCHAR *clear*)

#### Arguments

*clear* Clear pattern (shown below)

| bit 7 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 1 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Clear=0, Hold=1
bit 1: Command complete interrupt Clear=0, Hold=1
bit 2: VSYNC interrupt Clear=0, Hold=1
bit 3: Frame sync interrupt Clear=0, Hold=1
bit 4: External sync error interrupt Clear=0, Hold=1
All other bits: 1

**Figure 6.1.7 Clear Pattern Format**

#### Return value

None

#### Target GDC

MB86290A

#### Description

Clears the interrupt request indicated by bit0-4 in IST (Interrupt Status) register by the clear pattern specified as above. For *clear*, set "0" to the bit position to clear the interrupt respectively and set "1" to all other bits.

#### Notes

In MB86291 or later, use the **GdcGeoClearInterruptStatus** command.

### 6.1.8 GdcGeoClearInterruptStatus [Clears Interrupt Request for MB86291 or Later]

#### Interface

void **GdcGeoClearInterruptStatus** (GDC_ULONG *clear*)

#### Arguments

*clear* Clear pattern (shown below)

| bit 31 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 1 | . . . | x | x | x | x | x |

'x': used bit

bit 0: Command error interrupt Clear=0, Hold=1

bit 1: Command complete interrupt Clear=0, Hold=1

bit 2: VSYNC interrupt Clear=0, Hold=1

bit 3: Frame sync interrupt Clear=0, Hold=1

bit 4: External sync error interrupt Clear=0, Hold=1

All other bits: 1

**Figure 6.1.8 Clear Pattern Format**

#### Return value

None

#### Target GDC

MB86291 or later

#### Description

Clears the interrupt request indicated by bit0-4 in IST (Interrupt Status) register by the clear pattern specified as above. For *clear*, set "0" to the bit position to clear the interrupt respectively and set "1" to all other bits.

#### Notes

In MB86290A, use the **GdcClearInterruptStatus** command.

### 6.1.9 GdcGetFIFOStatus [Gets Status of Display List FIFO]

#### Interface

GDC_ULONG **GdcGetFIFOStatus** (void)

#### Arguments

None

#### Return value

Display list FIFO status in the following format:

| bit 31 | . . . | 2 | 1 | 0 |
|---|---|---|---|---|
| 0 | . . . | x | x | x |

'x': used bit

| | | |
|---|---|---|
| bit 0: | Valid data exists in display list FIFO | =0 |
| | Valid data does not exist in display list FIFO | =1 |
| bit 1: | Display list FIFO is full | =0 |
| | Display list FIFO is not full | =1 |
| bit 2: | Half or more entries of display list FIFO are empty | =0 |
| | Less than half entries of display list FIFO are empty | =1 |

All other bits: 0

**Figure 6.1.9 Display List FIFO Status**

#### Target GDC

All

#### Description

Reads IFSR (Input FIFO Status Register) registers and returns display list FIFO status.

### 6.1.10 GdcGetFIFORemain [Gets Remains of Display List FIFO]

#### Interface

GDC_ULONG **GdcGetFIFORemain** (void)

#### Arguments

None

#### Return value

Number of open entries in the display list FIFO (0 to 32)

#### Target GDC

All

#### Description

Reads IFCNT (Input FIFO CouNTer) register and returns the number of open entries in the display list FIFO.

### 6.1.11 GdcGetFIFOErrorStatus [Gets Error Status of Display List FIFO]

#### Interface

GDC_ULONG **GdcGetFIFOErrorStatus** (void)

#### Arguments

None

#### Return value

Display list FIFO error status in the following format:

| bit 31 | . . . | 1 | 0 |
|---|---|---|---|
| 0 | . . . | x | x |

'x': used bit

bit 0: Command error (type code of display list is invalid) Occur=1, None=0

bit 1: Packet error (command code of display list is invalid) Occur=1, None=0

All other bits: 0

**Figure 6.1.11 Display List FIFO Error Status**

#### Target GDC

All

#### Description

Reads EST (Error Status Register) and returns display list FIFO error status.

Command error or packet error occurs if invalid display list is sent to display list FIFO. To clear error status, execute software reset or hardware reset.

### 6.1.12 GdcGeoGetFIFOStatus [Gets Status of Geometry Display List FIFO]

#### Interface

GDC_ULONG **GdcGeoGetFIFOStatus** (void)

#### Arguments

None

#### Return value

Geometry display list FIFO status in the following format:

| bit 31 | . . . | 2 | 1 | 0 |
|---|---|---|---|---|
| 0 | . . . | x | x | x |

'x': used bit

| | | |
|---|---|---|
| bit 0: | Valid data exists in geometry display list FIFO | =0 |
| | Valid data does not exist in geometry display list FIFO | =1 |
| bit 1: | Geometry display list FIFO is full | =0 |
| | Geometry display list FIFO is not full | =1 |
| bit 2: | Half or more entries of geometry display list FIFO are empty | =0 |
| | Less than half entries of geometry display list FIFO are not empty | =1 |

All other bits: 0

**Figure 6.1.12 Display List FIFO Status**

#### Target GDC

MB86291 or later

#### Description

Returns current status of geometry display list FIFO.

### 6.1.13 GdcGeoGetFIFORemain [Gets Remains of Geometry Display List FIFO]

#### Interface

GDC_ULONG **GdcGeoGetFIFORemain** (void)

#### Arguments

None

#### Return value

Number of empty entries in the geometry display list FIFO, range from "0" to "32"

#### Target GDC

MB86291 or later

#### Description

Returns the number of empty entries in the geometry display list FIFO.

### 6.1.14 GdcGetPixelEngineStatus [Gets Status of Pixel Engine]

#### Interface

GDC_ULONG **GdcGetPixelEngineStatus** (void)

#### Arguments

None

#### Return value

Pixel engine status in the following format:

| bit 31 | . . . | 1 | 0 |
|---|---|---|---|
| Don't care | . . . | x | x |

'x': used bit

bit 1-0: Rendering is complete =00

Rendering is executing =01

All other bits: Don't care

**Figure 6.1.14 Pixel Engine Status**

#### Target GDC

MB86290A

#### Description

Returns Pixel Engine status.

#### Notes

In MB86291 or later, use the **GdcGeoGetPixelEngineStatus** command.

### 6.1.15 GdcGeoGetPixelEngineStatus [Gets Status of Geometry Pixel Engine]

#### Interface

GDC_ULONG **GdcGeoGetPixelEngineStatus** (void)

#### Arguments

None

#### Return value

Pixel engine status in the following format:

| bit 31 | . . . | 1 | 0 |
|---|---|---|---|
| Don't care | . . . | x | x |

'x': used bit

bit 1-0: Rendering is complete =00
Rendering is executing =01

All other bits: Don't care

**Figure 6.1.15 Geometry Pixel Engine Status**

#### Target GDC

MB86291 or later

#### Description

Returns Geometry Pixel Engine status.

#### Notes

In MB86290A, use the **GdcGetPixelEngineStatus** command.

### 6.1.16 GdcGetLocalDisplayListTransferStatus [Gets Status of Local Display List Transfer]

#### Interface

GDC_ULONG **GdcGetLocalDisplayListTransferStatus** (void)

#### Arguments

None

#### Return value

Transfer of local display list status in the following format:

| bit 31 | . . . | 1 | 0 |
|---|---|---|---|
| 0 | . . . | 0 | x |

'x': used bit

bit 0: Transfer is complete =0

Transfer is executing =1

All other bits: 0

**Figure 6.1.16 Local Display List Transfer Status**

#### Target GDC

All

#### Description

Returns status of transfer of local display list.

### 6.1.17 GdcQueryChipID [Queries Chip ID]

**Interface**

void **GdcQueryChipID** (int **chip_no*, int **version*)

**Arguments**

| | |
|---|---|
| *chip_no* | Pointer to the area which stores chip number |
| *version* | Pointer to the area which stores chip version number |

**Return value**

None

**Target GDC**

MB86293 or later

**Description**

Returns chip number and chip version number.

Chip number and chip version number are numerical value.

For details about each number, refer to the hardware specification of the Graphics Controller of use.

### 6.1.18 GdcQueryGDCType [Queries Graphics Controller Type]

#### Interface

GDC_ULONG **GdcQueryGDCType** (void)

#### Arguments

None

#### Return value

Type of the Graphics Controller, specify any of the following

| | |
|---|---|
| **GDC_TYPE_MB86290A** | MB86290A |
| **GDC_TYPE_MB86291** | MB86291 or MB86291S |
| **GDC_TYPE_MB86291A** | MB86291A or MB86291AS |
| **GDC_TYPE_MB86292** | MB86292 or MB86292S |
| **GDC_TYPE_MB86293** | MB86293 |
| **GDC_TYPE_MB86294** | MB86294 or MB86294S |
| **GDC_TYPE_MB86295** | MB86295S |
| **GDC_TYPE_MB86296** | MB86296S |

#### Target GDC

All

#### Description

Returns type of Graphics Controller.

#### Notes

The type of the Graphics Controller which this command returns is a type specified when the **GdcInitialize** command is called. Therefore, if a different Graphics Controller type from what is actually used is specified at the **GdcInitialize** command, this command also returns the different one.

### 6.1.19 GdcQueryVersion [Queries Version Number]

**Interface**

void **GdcQueryVersion** (int **version*, int **level*)

**Arguments**

| | |
|---|---|
| *version* | Pointer to store version number |
| *level* | Pointer to store level number |

**Return value**

None

**Target GDC**

All

**Description**

Returns current version and level number of the Graphics Driver.

*Version* number and *level* number are numerical value. For example, when the version number is "2", and the level number is "1", the following numbers are stored in each parameter.

**version* = 2

**level* = 1

## 6.2 Context Control Command

### 6.2.1 XGdcCreateContext [Creates Context]

**Interface**

GDC_BOOL **XGdcCreateContext** (GDC_CTX **drvctx*,
GDC_ULONG *dlbufnum*,
GDC_DLBUF_STRUCT **dlbufstr*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *dlbufnum* | Number of DL buffer to be used, 1 or more |
| *dlbufstr* | Pointer to DL buffer structure information table |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_ILLEGAL_NUMBER_OF_DL_BUF**

Illegal number of DL buffer has been specified

**GDC_ERR_ADDRESS_MISS_ALIGN**

Address of Context, DL buffer structure information or DL buffer is not on 4-byte boundary

**GDC_ERR_DL_BUF_TOO_SMALL**

Size of DL buffer is smaller than the minimum size

**GDC_ERR_DL_SIZE**

DL buffer size is not a multiple of 32 bytes

**Target GDC**

All

**Description**

Creates context.

Allocate memory space for the context, and specify that address for *drvctx*.

Specify number of DL buffers to be used for *dlbufnum*.

Specify address of array of DL buffer structure information for *dlbufstr*. The number of elements of that array is required for the number of DL buffers to be used. It is necessary to allocate memory space for DL buffer structure information

and to set each parameter before the calling this command. Refer to "5.2.10 GDC_DLBUF_STRUCT [DL Buffer Structure Information]" for specification of DL buffer structure information.

Address of context area, DL buffer structure information and each DL buffer must be on 4-byte boundary. Also, size of each DL buffer must be a multiple of 32 bytes.

**Notes**

- Since DL buffer structure information is referred by each Driver Command after this command returns, do not discard it.
- Adjust the allocation address of DL buffer according to DMA that you use when transferring display list by DMA. Generally, DMA requires that allocation address shall be on boundary for transferring the unit.

## 6.3 Error Control Commands

### 6.3.1 XGdcGetErrCode [Gets Error Code]

**Interface**

int **XGdcGetErrCode** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

Error code

**Target GDC**

All

**Description**

Returns an error code of the Driver Command. After the Driver Command returns, the cause of abnormal termination can be taken by the calling of this command.

Specify **NULL** for *drvctx* when to get error code of the Driver Command which does not use context such as the display control commands.

**Notes**

- Error code can be hold only one. Returns last error code when this command is called after the calling of two or more the Driver Command. However, holds separately the error code of commands use the Context and the error code of other commands.

Error code is not cleared automatically. To clear the error code, call the **XGdcSetErrCode** command with **GDC_ERR_NOERROR**.

| [Error code] | [Description] |
|---|---|
| GDC_ERR_NOERROR | No error has occurred |
| GDC_ERR_DL_SIZE | DL buffer size is not a multiple of 32 bytes |
| GDC_ERR_DATA_TOO_BIG | Too large data |
| GDC_ERR_INVALID_LAYER | Invalid layer has been specified |
| GDC_ERR_INVALID_BANK | Invalid bank has been specified |
| GDC_ERR_INVALID_COLOR_MODE | Invalid color mode has been specified |
| GDC_ERR_INVALID_CURSOR_NUMBER | Invalid cursor number has been specified |
| GDC_ERR_ILLEGAL_DIMENSION | Illegal vertical/horizontal size |
| GDC_ERR_INVALID_ATTRIBUTE | Invalid attribute has been specified |

(Continue)

(Continued)

| [Error code] | [Description] |
|---|---|
| GDC_ERR_INVALID_PRIMITIVE | Invalid primitive has been specified |
| GDC_ERR_ILLEGAL_VERTEX_COUNT | Illegal number of vertex |
| GDC_ERR_ILLEGAL_LINE_WIDTH | Illegal width of line |
| GDC_ERR_NOT_READY | Graphics Driver is not initialized |
| GDC_ERR_INVALID_CPU_TYPE | Invalid CPU type has been specified |
| GDC_ERR_INVALID_GDC_TYPE | Invalid Graphics Controller type has been specified |
| GDC_ERR_INVALID_FLAG | Invalid flag has been specified |
| GDC_ERR_ADDRESS_MISS_ALIGN | Address of Context area, DL buffer Info or DL buffer is not 4byte boundary |
| GDC_ERR_DL_BUF_TOO_SMALL | DL buffer size is less than the minimum |
| GDC_ERR_INVALID_PARAMETER | Invalid parameter has been specified |
| GDC_ERR_SWITCH_DL_BUF_FAILED | The **XGdcSwitchDLBuf** command has finished abnormally |
| GDC_ERR_ILLEGAL_DL_BUF_NO | Illegal DL buffer number has been specified |
| GDC_ERR_ILLEGAL_NUMBER_OF_DL_BUF | Illegal number of DL buffer has been specified |
| GDC_ERR_NOT_START_GEO_PRIM_TYPE | The **XGdcGeoPrimEnd** command has called before the call of the **XGdcGeoPrimType** command and the **XGdcGeoDrawVertex** command |
| GDC_ERR_NOT_START_GEO_DRAW_VERTEX | The **XGdcGeoPrimEnd** command has called before the call of the **XGdcGeoDrawVertex** command |
| GDC_ERR_ILLEGAL_CAPTURE_SCALE | An illegal video capture scale has specified |

### 6.3.2 XGdcSetErrCode [Sets Error Code]

**Interface**

void **XGdcSetErrCode** (GDC_CTX **drvctx*, int *errcode*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *errcode* | Error code |

**Return value**

None

**Target GDC**

All

**Description**

Sets an error code. Use this command for the following purposes.

- To clear error code
- To set error code of the error occurred inside of the System Dependent Commands

Specify **NULL** for *drvctx* when to set error code of the Driver Command which does not use context such as the display control commands.

**Notes**

When defining a new error code in application programs, use a value of "1000" or over.

# 6.4 Display Setting Commands

## 6.4.1 GdcDispClock [Sets Display Clock Mode]

### Interface

void **GdcDispClock** (GDC_ULONG *mode*)

### Arguments

| | |
|---|---|
| *mode* | Sets display clock mode. This parameter is directly set to DCM (MB86290A/86291/86292) or DCEM (MB86293 or later) register of the Graphics Controller. For details of the DCM and DCEM register description, refer to the Graphics Controller hardware specification of use. |

### Return value

None

### Target GDC

All

### Description

Controls display clock and sync mode and etc. by setting following parameters to display control mode register.

- Set display sync mode
- Set external sync mode
- Select signal type
- Select dot clock frequency
- Select dot clock source

### 6.4.2 GdcDispTiming [Sets Display Timing Parameters]

**Interface**

void **GdcDispTiming** (GDC_USHORT *htp*, GDC_USHORT *hsp*,
GDC_USHORT *hsw*, GDC_USHORT *hdp*,
GDC_USHORT *vtr*, GDC_USHORT *vsp*,
GDC_USHORT *vsw*, GDC_USHORT *vdp*)

**Arguments**

| | |
|---|---|
| *htp* | Total horizontal pixel count, range from "1" to "4096" (*1) |
| *hsp* | Hsync pulse position, range from "1" to "4096" (*1) |
| *hsw* | Hsync pulse width, range from "1" to "256" (*1) |
| *hdp* | Horizontal display pixel count, range from "1" to "4096" (*1) |
| *vtr* | Total vertical raster count, range from "1" to "4096" (*1) |
| *vsp* | Vsync pulse position, range from "1" to "4096" (*1) |
| *vsw* | Vsync pulse width, range from "1" to "64" (*1) |
| *vdp* | Vertical display raster count, range from "1" to "4096" (*1) |

**Return value**

None

**Target GDC**

All

**Description**

Sets display window size and display timing parameters.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.3 GdcDispTimingWindow [Sets Display Position of Layer W]

**Interface**

void **GdcDispTimingWindow** (GDC_USHORT *x*, GDC_USHORT *y*,
GDC_USHORT *w*, GDC_USHORT *h*)

**Arguments**

| | |
|---|---|
| *x* | x coordinate in the device coordinates, range from "0" to "4095" (*1) |
| *y* | y coordinate in the device coordinates, range from "0" to "4095" (*1) |
| *w* | Window width, in pixels, range from "1" to "4096" (*1) |
| *h* | Window height, in pixels, range from "1" to "4096" (*1) |

**Return value**

None

**Target GDC**

All

**Description**

Sets display position of layer W.

Specify the position to display the upper left of a window frame as *x* and *y*.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.4 GdcDispDividePos [Sets Border Position of Display Partition]

**Interface**

void **GdcDispDividePos** (GDC_USHORT *hdb*)

**Arguments**

| | |
|---|---|
| *hdb* | Horizontal pixel count of left window, range from "1" to "4096" (*1) |

**Return value**

None

**Target GDC**

All

**Description**

Sets the position where the display is divided into left and right.

Value "0" cannot be specified for *hdb*.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.5 GdcDispDimension [Sets Attributes of Display Frame]

**Interface**

GDC_BOOL **GdcDispDimension** (GDC_UCHAR *layer*, GDC_UCHAR *enable*,
GDC_UCHAR *cmode*, GDC_UCHAR *fmode*,
GDC_ULONG *loa0*, GDC_ULONG *loa1*,
GDC_USHORT *lw*, GDC_USHORT *lh*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_W** | Layer W |
| **GDC_DISP_LAYER_ML** | Layer ML |
| **GDC_DISP_LAYER_MR** | Layer MR |
| **GDC_DISP_LAYER_BL** | Layer BL |
| **GDC_DISP_LAYER_BR** | Layer BR |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*enable* Layer display enable/disable

| | |
|---|---|
| **GDC_ENABLE** | Layer display enable |
| **GDC_DISABLE** | Layer display disable |

*cmode* Color mode selection

| | |
|---|---|
| **GDC_24BPP_FORMAT** | 24-bit color mode |
| **GDC_16BPP_FORMAT** | 16-bit color mode |
| **GDC_8BPP_FORMAT** | 8-bit color mode |

*fmode* Flipping mode selection

| | |
|---|---|
| **GDC_FLIPMODE_0** | Display bank 0 |
| **GDC_FLIPMODE_1** | Display bank 1 |
| **GDC_FLIPMODE_AUTO** | Display both banks alternately |

*loa0* Top address of logical frame of bank 0, specify offset from top of the graphics memory

| | |
|---|---|
| *loa1* | Top address of logical frame of bank 1, specify offset from top of the graphics memory |
| *lw* | Logical frame width, in pixels (*1)<br>Range from "16" to "4096", in 16 pixel unit, in 24-bit color mode<br>Range from "32" to "4096", in 32 pixel unit, in 16-bit color mode<br>Range from "64" to "4096", in 64 pixel unit, in 8-bit color mode |
| *lh* | Logical frame height, in pixels, range from "1" to "4096" (*1) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid layer has been specified |

**Target GDC**

All

**Description**

Sets attributes of logical frame. Attributes are independent with each display layer (C, W, ML, MR, BL and BR).

If this command is called, the display origin address of the specified layer will be set as the same value as the specified display frame top address. (Sets to upper left corner of the frame.)

When layer C or layer L0 is specified, *fmode* and *loa1* are not used.

When layer W or layer L1 is specified, *fmode*, *loa1* and *lh* is not used.

Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- **GDC_DISP_LAYER_C**
- **GDC_DISP_LAYER_W**
- **GDC_DISP_LAYER_ML**
- **GDC_DISP_LAYER_MR**
- **GDC_DISP_LAYER_BL**

- **GDC_DISP_LAYER_BR**

[Macros for extended display]

- **GDC_DISP_LAYER_L0**
- **GDC_DISP_LAYER_L1**
- **GDC_DISP_LAYER_L2**
- **GDC_DISP_LAYER_L3**
- **GDC_DISP_LAYER_L4**
- **GDC_DISP_LAYER_L5**

When ML or MR is displayed by using "Macros for standard display", both ML and MR are displayed. Similarly, when BL or BR is displayed by using "Macros for standard display", both BL and BR are displayed.
When layer L5 is used as a blend coefficient layer, this layer must be displayed in 8-bit color mode.
In case of extend display mode or extend overlay mode, layer L4 and L5 are not available in 8-bit color mode (except layer L5 is used as a blend coefficient layer).

(*1) The range check of the parameter is not performed, set a valid value.

Notes

When the width of the drawing frame specified with the **XGdcDrawDimension** command doesn't match with the width of the display frame specified with this command, it is not correctly displayed.

### 6.4.6 GdcDispOn [Asserts Video Signal Output]

**Interface**

void **GdcDispOn** (void)

**Arguments**

None

**Return value**

None

**Target GDC**

All

**Description**

Outputs video signals.

Screen display is started at this command call, so this command must be called after all the rest display parameters are set. The displaying might fall into disorder when this command is called in the case that the other display parameters are not set appropriately. Nothing is displayed prior to this command call.

### 6.4.7 GdcDispOff [Negates Video Signal Output]

**Interface**

void **GdcDispOff** (void)

**Arguments**

None

**Return value**

None

**Target GDC**

All

**Description**

Disables screen display of video signals. Though the video signal output is suppressed immediately after this command is called, this command does not affect drawing processing by the Graphics Controller.

### 6.4.8 GdcDispLayerOn [Asserts Screen Display]

**Interface**

GDC_BOOL **GdcDispLayerOn** (GDC_UCHAR *layer*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_W** | Layer W |
| **GDC_DISP_LAYER_M** | Layer M |
| **GDC_DISP_LAYER_B** | Layer B |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

All

**Description**

Displays the layer specified by *layer*.

When the following macros are specified, both layer ML and MR, both BL and BR are displayed simultaneously.

- **GDC_DISP_LAYER_M**
- **GDC_DISP_LAYER_B**

Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- **GDC_DISP_LAYER_C**
- **GDC_DISP_LAYER_W**

- GDC_DISP_LAYER_M
- GDC_DISP_LAYER_B

[Macros for extended display]

- GDC_DISP_LAYER_L0
- GDC_DISP_LAYER_L1
- GDC_DISP_LAYER_L2
- GDC_DISP_LAYER_L3
- GDC_DISP_LAYER_L4
- GDC_DISP_LAYER_L5

When the layer L5 is used as a blend coefficient layer, this layer must be displayed.

### 6.4.9 GdcDispLayerOff [Negates Screen Display]

**Interface**

GDC_BOOL **GdcDispLayerOff** (GDC_UCHAR *layer*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_W** | Layer W |
| **GDC_DISP_LAYER_M** | Layer M |
| **GDC_DISP_LAYER_B** | Layer B |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

All

**Description**

The layer specified with *layer* is made not-display.

When the following macros are specified, both layer ML and MR, both BL and BR are not displayed simultaneously.

- **GDC_DISP_LAYER_M**
- **GDC_DISP_LAYER_B**

Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- **GDC_DISP_LAYER_C**
- **GDC_DISP_LAYER_W**

- GDC_DISP_LAYER_M
- GDC_DISP_LAYER_B

[Macros for extended display]

- GDC_DISP_LAYER_L0
- GDC_DISP_LAYER_L1
- GDC_DISP_LAYER_L2
- GDC_DISP_LAYER_L3
- GDC_DISP_LAYER_L4
- GDC_DISP_LAYER_L5

### 6.4.10 GdcDispPos [Sets Display Start Position]

**Interface**

GDC_BOOL **GdcDispPos** (GDC_UCHAR *layer*, GDC_UCHAR *bank*,
GDC_USHORT *dx*, GDC_USHORT *dy*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_ML** | Layer ML |
| **GDC_DISP_LAYER_MR** | Layer MR |
| **GDC_DISP_LAYER_BL** | Layer BL |
| **GDC_DISP_LAYER_BR** | Layer BR |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*bank* Logical frame bank selection

| | |
|---|---|
| **GDC_DISP_BANK_0** | Bank 0 |
| **GDC_DISP_BANK_1** | Bank 1 |

*dx* x coordinate of display start position, range from "0" to "4095" (*1)

*dy* y coordinate of display start position, range from "0" to "4095" (*1)
The value of parameter is not checked of the range, specify a value within the decided range.

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_BANK** | Invalid bank has been specified |

**Target GDC**

All

### Description

Sets the display start position relatively from the base position of logical frame.
After this command execution, it set as the target of display of the bank specified by *bank*.
Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- **GDC_DISP_LAYER_C**
- **GDC_DISP_LAYER_ML**
- **GDC_DISP_LAYER_MR**
- **GDC_DISP_LAYER_BL**
- **GDC_DISP_LAYER_BR**

[Macros for extended display]

- **GDC_DISP_LAYER_L0**
- **GDC_DISP_LAYER_L2**
- **GDC_DISP_LAYER_L3**
- **GDC_DISP_LAYER_L4**
- **GDC_DISP_LAYER_L5**

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.11 GdcDispDoFlip [Flips Display Bank]

**Interface**

GDC_BOOL **GdcDispDoFlip** (GDC_UCHAR *layer*, GDC_UCHAR *bank*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_ML** | Layer ML |
| **GDC_DISP_LAYER_MR** | Layer MR |
| **GDC_DISP_LAYER_BL** | Layer BL |
| **GDC_DISP_LAYER_BR** | Layer BR |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*bank* Logical frame bank selection

| | |
|---|---|
| **GDC_DISP_BANK_0** | Bank 0 |
| **GDC_DISP_BANK_1** | Bank 1 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_BANK** | Invalid bank has been specified |

**Target GDC**

All

**Description**

Switches bank to be displayed (Flipping).
Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- **GDC_DISP_LAYER_ML**
- **GDC_DISP_LAYER_MR**

- GDC_DISP_LAYER_BL
- GDC_DISP_LAYER_BR

[Macros for extended display]

- GDC_DISP_LAYER_L2
- GDC_DISP_LAYER_L3
- GDC_DISP_LAYER_L4
- GDC_DISP_LAYER_L5

### 6.4.12 GdcOverlayPriorityMode [Sets Overlay Display Mode]

**Interface**

GDC_BOOL **GdcOverlayPriorityMode** (GDC_UCHAR *mode*)

**Arguments**

*mode* Layer C overlay mode

**GDC_OVERLAY_C_PRIORITY**
Simple priority mode (default)
**GDC_OVERLAY_C_BLEND**
Blend mode

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

All

**Description**

Sets overlay display mode. When the simple priority mode is selected, layer C is always displayed at the top of the layers. When the blend mode is selected, all of the layers except layer C are ordered in their priority order, and then layer C is transparently blended with them to be displayed (Overlay blend mode).

### 6.4.13 GdcOverlayBlend [Sets Blend Parameter for Overlay Blend]

**Interface**

GDC_BOOL **GdcOverlayBlend** (GDC_UCHAR *select*, GDC_UCHAR *blend*)

**Arguments**

*select* Overlay blend selection

| | |
|---|---|
| **GDC_BLEND_RATIO_C** | Blend target is layer C color |
| **GDC_BLEND_RATIO_WMB** | Blend target is layer W/M/B color |

*blend* Blending coefficient, only upper 4 bits are valid, range from "0x00" to "0xf0" (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

All

**Description**

Sets the blend coefficient to determine the layer C color when the overlay mode is blend mode.

The followings are the meanings of blend coefficient and expression to determine the layer C color.

[Blend Coefficient]

| *blend* | Blend Coefficient |
|---|---|
| 0x00 | 0 |
| 0x10 | 1/16 |
| 0x20 | 2/16 |
| 0x30 | 3/16 |
| : | : |
| 0xf0 | 15/16 |

[Blend Expression]

- For **GDC_BLEND_RATIO_C**
  (layer_C_color * blend_coefficient)
  + (layer_W/M/B_compound_color * (1- blend_coefficient))
- For **GDC_BLEND_RATIO_WMB**
  (layer_C_color * (1- blend_coefficient))
  + (layer_W/M/B_ compound_color * blend_coefficient)

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.14 GdcDispDisplayMode [Sets Display Mode]

**Interface**

GDC_BOOL **GdcDispDisplayMode** (GDC_UCHAR *mode*)

**Arguments**

*mode* Display mode

| | |
|---|---|
| **GDC_STANDARD_MODE** | Standard Display Mode (default) |
| **GDC_OVERLAY_EXT_MODE** | Extend Overlay Mode |
| **GDC_WINDOW_MODE** | Window Mode |
| **GDC_EXTEND_MODE** | Extend Display Mode |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets all layers (except layer L1) to same display mode.

There are four kinds of display modes. In each display mode, the layer order that can be set up is different. Difference among these modes are shown in Table 6.4.14a and Table 6.4.14b. And also displayed image of Standard Display Mode and Extend Display Mode are shown in Figure 6.4.14.

**Table 6.4.14a Functions of Each Display Mode**

| Function | Extend Display Mode | Window Mode | Extend Overlay Mode | Standard Display Mode |
|---|---|---|---|---|
| Overlay | 6 Layers | 6 Layers | 4 Layers + Right and Left Division | 4 Layers + Right and Left Division |
| Displaying Window | 6 Layers | 6 Layers | 1 Layer | 1 Layer |
| Displaying Order of Layer | Changeable Optionally | Layer L0 Only Changeable | Changeable Optionally | Layer L0 Only Changeable |
| Number of Color Palette | 4 | 2 | 4 | 2 |

**Table 6.4.14b Layer Order of Each Display Mode**

| Layer | Standard Display Mode and Window Mode | Extend Display Mode and Extend Overlay Mode |
|---|---|---|
| L0 | Set Highest Layer or Lowest Layer (*1) | Any display order can be set |
| L1 | Set Highest Layer or Under Layer L0 (*1) | Ditto |
| L2 | Set Under Layer L1 | Ditto |
| L3 | Ditto | Ditto |
| L4 | Ditto | Ditto |
| L5 | Ditto | Ditto |

(*1) The order of layer L0 and L1 is determined by the **GdcDispLayerOrder** command.

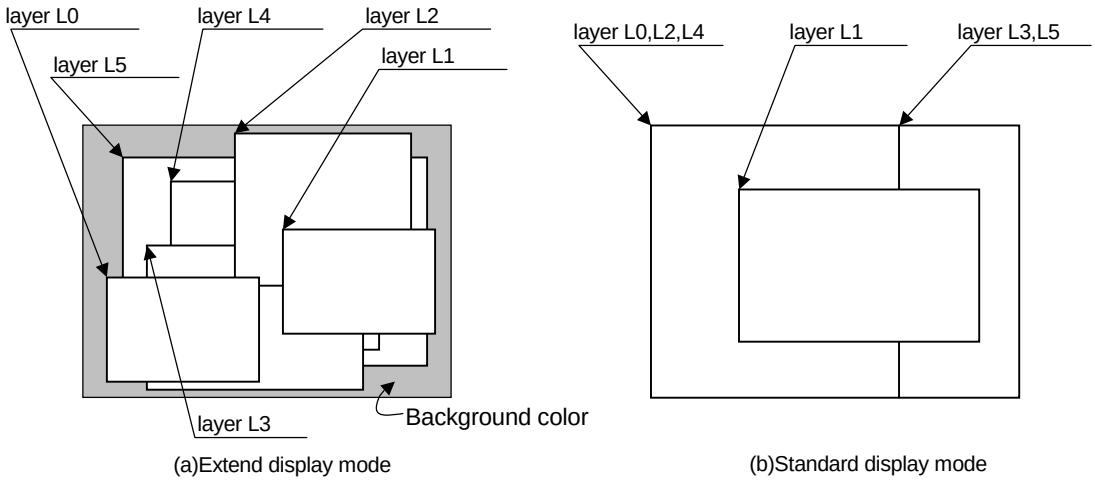



**Figure 6.4.14 Displayed Image of Standard Display Mode and Extend Display Mode**

### 6.4.15 GdcDispDisplayLayerMode [Sets Layer Display Mode]

**Interface**

GDC_BOOL **GdcDispDisplayLayerMode** (GDC_UCHAR *layer*,
GDC_UCHAR *mode*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*mode* Layer display mode

| | |
|---|---|
| **GDC_STANDARD_MODE** | Standard Display Mode (default) |
| **GDC_OVERLAY_EXT_MODE** | Extend Overlay Mode |
| **GDC_WINDOW_MODE** | Window Mode |
| **GDC_EXTEND_MODE** | Extend Display Mode |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

MB86293 or later

**Description**

Changes display mode in each layer.

Layer L1 can use the function in extend display mode by the default.

In extend display mode setting, window mode and an extend overlay mode can be set at the same time. **GDC_EXTEND_MODE** has the following meanings.

**GDC_EXTEND_MODE = GDC_WINDOW_MODE | GDC_OVERLAY_EXT_MODE**

When layer L5 is used as a blend coefficient layer, this layer must be set extend display mode.

### 6.4.16 GdcDispSetBackColor [Sets Background Color]

**Interface**

GDC_BOOL **GdcDispSetBackColor** (GDC_COL24 *color*)

**Arguments**

| | |
|---|---|
| *color* | 24 bits background color |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86293 or later

**Description**

Sets background color. This color is used to fill area where outside of layers.

This command is available in all display modes.

### 6.4.17 GdcDispSetLayerWindow [Sets Position and Size of Window Mode Layer]

**Interface**

GDC_BOOL **GdcDispSetLayerWindow** (GDC_UCHAR *layer*,
GDC_USHORT *x*, GDC_USHORT *y*,
GDC_USHORT *w*, GDC_USHORT *h*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*x* x coordinate in the device coordinates, range from "0" to "4095" (*1)

*y* y coordinate in the device coordinates, range from "0" to "4095" (*1)

*w* Window width, in pixels, range from "1" to "4096", default is a value of hdp of the "initialize parameter" (*1)

*h* Window height, in pixels, range from "1" to "4096", default is a value of vdp of the "initialize parameter" (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets x, y position, width and height of the layer which has been set up to Window Mode.

Setting to layer L1 is also available by the **GdcDispTimingWindow** command.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.4.18 GdcLayerOverlayPriorityMode [Sets Overlay Display Mode in Every Layer]

**Interface**

GDC_BOOL **GdcLayerOverlayPriorityMode** (GDC_UCHAR *layer*,
GDC_UCHAR *mode*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*mode* Overlay display mode

| | |
|---|---|
| **GDC_OVERLAY_PRIORITY** | Overlay with transparent color (default) |
| **GDC_OVERLAY_BLEND** | Overlay with blend |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets overlay display mode of each layer.

To use overlay with transparent color (**GDC_OVERLAY_PRIORITY**), set to Extend Display Mode or Extend Overlay Mode by the **GdcDispDisplayLayerMode** command.

To use overlay with blend (**GDC_OVERLAY_BLEND**), set to Extend Display Mode or Extend Overlay Mode by the **GdcDispDisplayLayerMode** command, and set to Blend Mode by the **GdcLayerOverlayBlend** command.

If Standard Display Mode or Window Mode is selected, this command is not effective.

Setting of layer C by the **GdcOverlayPriorityMode** command reflects on layer L0.

On the contrary, setting to layer L0 by this command refracts on layer C.

When layer L5 is used as a blend coefficient layer, this layer is not target of overlaying by transparent color or blend.

### 6.4.19 GdcLayerOverlayBlend [Sets Blend Mode in Every Layer]

**Interface**

GDC_BOOL **GdcLayerOverlayBlend** (GDC_UCHAR *layer*, GDC_UCHAR *select*, GDC_UCHAR *correct*, GDC_UCHAR *source*, GDC_UCHAR *blend*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*select* Blend calculating method

**GDC_BLEND_CURRENT_RATIO**

layer color * blending coefficient +
lower layer color * (1 - blending coefficient)

**GDC_BLEND_ONE_MINUS_CURRENT_RATIO**

layer color * (1 - blending coefficient) +
lower layer color * blending coefficient

*correct* Correction by 1/256 value

**GDC_BLEND_NO_CORRECT**

Uses the blending coefficient

**GDC_BLEND_CORRECT**

When the blending coefficient is not "0", add "1/256"
(When using the blending coefficient of 100%)

*source* Selects source data of the blending coefficient

**GDC_BLEND_RATIO_CONSTANT**

Uses the fixed value as blending coefficient specified by *blend*

**GDC_BLEND_RATIO_L5**

Uses pixel value of layer L5 for the blending coefficient

| | |
|---|---|
| *blend* | Blending coefficient, range from "0x00" to "0xff"<br>(Used when *source* is **GDC_BLEND_RATIO_CONSTANT**) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets the blend mode in the case of using overlay with blend.

Setting by the **GdcOverlayBlend** command reflects on layer L0. *correct* and *source* are set to **GDC_BLEND_NO_CORRECT** and **GDC_BLEND_RATIO_CONSTANT** respectively when the mode is set by the **GdcOverlayBlend** command.

Setting of *source* is not valid when layer L5 is set.

When the layer L5 is used as a blend coefficient layer, set layer L5 to Extend Display Mode by the **GdcDispDisplayLayerMode** command.

When the layer L1 is capture mode and layer L5 is used as blend coefficient layer, layer L1 is not blended correctly.

The meaning of blend coefficient is as follows.

[Blend Coefficient]

| *blend* | Blend Coefficient |
|---|---|
| 0x00 | 0 |
| 0x01 | 1/256 |
| 0x02 | 2/256 |
| 0x03 | 3/256 |
| : | : |
| 0xff | 255/256 |

### 6.4.20 GdcDispLayerOrder [Sets Layer Display Order]

**Interface**

GDC_BOOL **GdcDispLayerOrder** (GDC_UCHAR *layer0*, GDC_UCHAR *layer1*,
GDC_UCHAR *layer2*, GDC_UCHAR *layer3*,
GDC_UCHAR *layer4*, GDC_UCHAR *layer5*)

**Arguments**

*layer0* Top level layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |
| **GDC_NO_LAYER** | Not select layer |

*layer1* Selects the second level layer (as for the value, the same in case of *layer0*)

*layer2* Selects the third level layer (as for the value, the same in case of *layer0*)

*layer3* Selects the fourth level layer (as for the value, the same in case of *layer0*)

*layer4* Selects the fifth level layer (as for the value, the same in case of *layer0*)

*layer5* Selects the sixth level layer (as for the value, the same in case of *layer0*)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets display order of layers. Refer to the Graphics Controller hardware specifications for details

Not to display some of layers, specify **GDC_NO_LAYER**.

In order to set up the display order of layers at pleasure, it is necessary to set to Extend Display Mode or Extend Overlay Mode by the **GdcDispDisplayLayerMode** command. If display mode is set to Standard Display Mode or Window Mode, restrictions occur in order of layers.

For details of that restriction, refer to "Table 6.4.14b Layer Order of Each Display Mode" in 6.4.14 GdcDispDisplayMode [Sets Display Mode] .

The same layer cannot be specified for each layer. If the same layer is specified for each layer, result of display is not guaranteed.

When the layer L5 is used as a blend coefficient layer, this layer should not be selected.

Figure 6.4.20a and 6.4.20b (on next page) are shown displayed image of layer overlay. Figure 6.4.20a shows displayed image when all layers are set to Standard Display Mode (display order is layer L1, L0, L2, L3, L4 and L5). Figure 6.4.20b shows when layer L0 and L5 are Standard Display Mode and other layers are Extend Display Mode (display order is layer L1, L0, L2, **GDC_NO_LAYER**, L4 and L5).

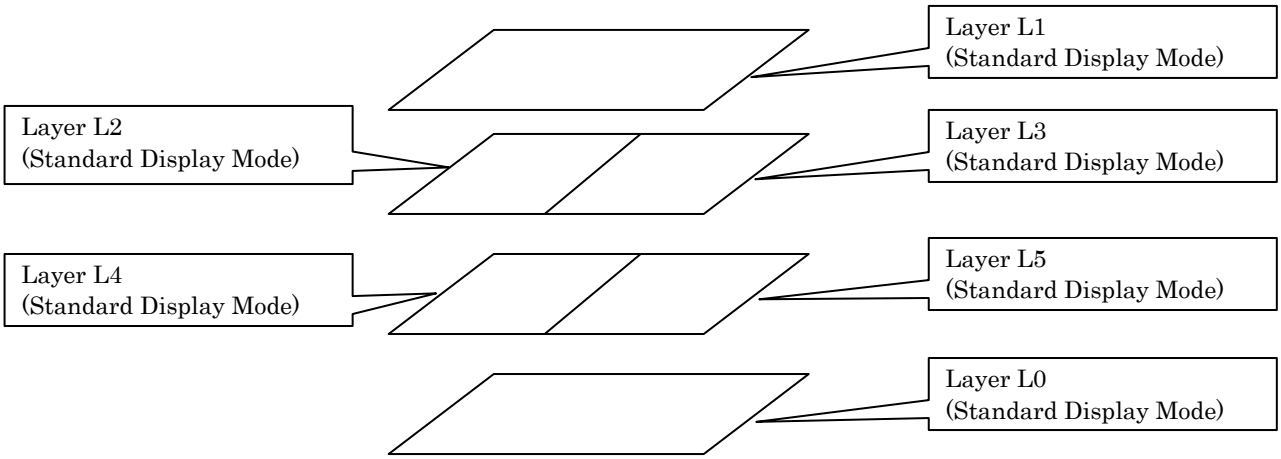



**Figure 6.4.20a Display Image when Display Order is Layer L1, L0, L2, L3, L4 and L5**

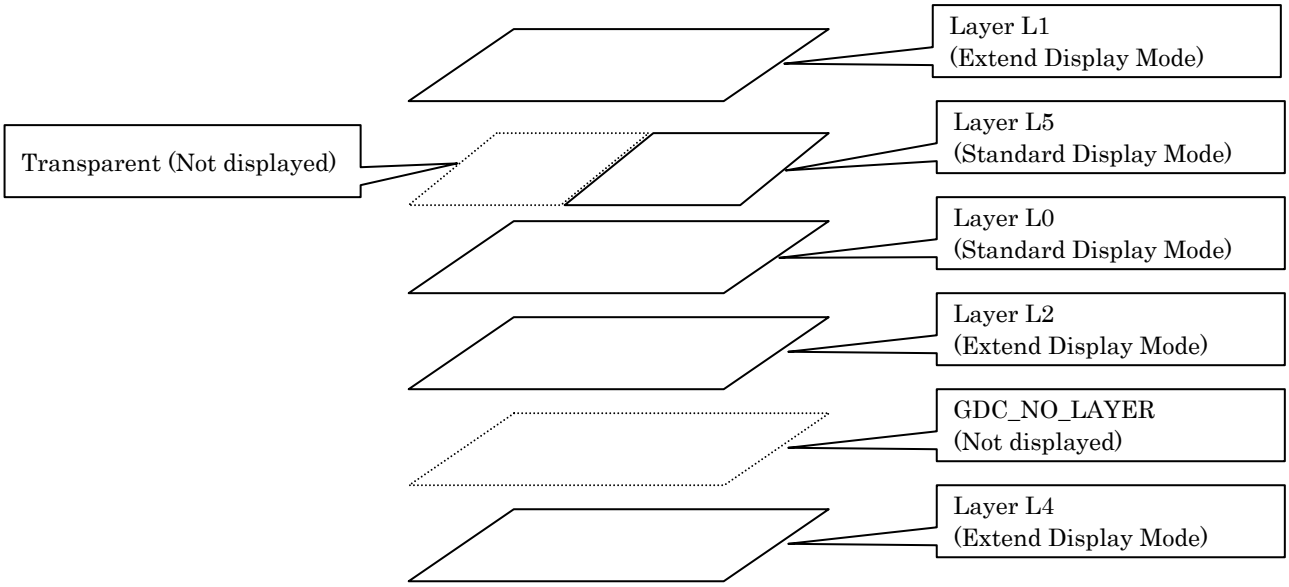



**Figure 6.4.20b Display Image when Display Order is Layer L1, L0, L2, GDC_NO_LAYER, L4 and L5**

### 6.4.21 GdcMultiDisplayMode [Shifts to Dual Display Mode]

**Interface**

void **GdcMultiDisplayMode** (void)

**Arguments**

None

**Return value**

None

**Target GDC**

MB86296S

**Description**

Shifts to a dual display mode. Nothing is processed and it returns from the command when a present display mode is a dual display mode. Processes as follows when you call this command.

1. The video signal output is controlled. (Display off.)
2. The ratio of dividing frequency of the display clock is set for a dual display mode.
3. Shifts to a dual display mode.
4. Changes the display status for the dual display mode set before this command call.
5. Returns to the output status of the video signal before this command is called.

### 6.4.22 GdcSingleDisplayMode [Shifts to Single Display Mode]

**Interface**

void **GdcSingleDisplayMode** (void)

**Arguments**

None

**Return value**

None

**Target GDC**

MB86296S

**Description**

Shifts to a single display mode. Nothing is processed and it returns from the command when a present display mode is a single display mode. Processes as follows when you call this command.

1. The video signal output is controlled. (Display off.)
2. The ratio of dividing frequency of the display clock is set for a single display mode.
3. Shifts to a single display mode.
4. Changes the display status for the single display mode set before this command call.
5. Returns to the output status of the video signal before this command is called.

### 6.4.23 GdcMultiDispLayerOn [Sets Layer Display for Dual Display Mode]

**Interface**

GDC_BOOL **GdcMultiDispLayerOn** (GDC_ULONG *disp_no*, GDC_UCHAR *layer*)

**Arguments**

*disp_no* Specifications of display number

| | |
|---|---|
| **GDC_DISPLAY_0** | Display #0 |
| **GDC_DISPLAY_1** | Display #1 |

*layer* Layer specification

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

MB86296S

**Description**

Sets the layer display for a dual display mode.

Sets the display status after shifted to a dual display when you call this command at a single display mode (The display status at that time is not changed). Changes the display status immediately when you call this command at a dual display mode.

### 6.4.24 GdcMultiDispLayerOff [Sets Layer Not-display for Dual Display Mode]

**Interface**

GDC_BOOL **GdcMultiDispLayerOff** (GDC_ULONG *disp_no*, GDC_UCHAR *layer*)

**Arguments**

*disp_no* Specifications of display number

| | |
|---|---|
| **GDC_DISPLAY_0** | Display #0 |
| **GDC_DISPLAY_1** | Display #1 |

*layer* Layer specification

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

MB86296S

**Description**

Sets the layer not-display for a dual display mode.

Sets the display status after shifted to a dual display when you call this command at a single display mode (The display at that time is not changed). Changes the display status immediately when you call this command at a dual display mode.

### 6.4.25 GdcMultiDispCursorOn [Sets Cursor Display for Dual Display Mode]

**Interface**

GDC_BOOL **GdcMultiDispCursorOn** (GDC_ULONG *disp_no*,
GDC_UCHAR *numCursor*)

**Arguments**

*disp_no* Specifications of display number

| | |
|---|---|
| **GDC_DISPLAY_0** | Display #0 |
| **GDC_DISPLAY_1** | Display #1 |

*numCursor* Cursor number (0 or 1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**Target GDC**

MB86296S

**Description**

Sets the cursor display for a dual display mode.

Sets the display status after shifted to a dual display when you call this command at a single display mode. (The display at that time is not changed.) Changes the display status immediately when you call this command at a dual display mode.

### 6.4.26 GdcMultiDispCursorOff [Sets Cursor Not-display for Dual Display Mode]

**Interface**

GDC_BOOL **GdcMultiDispCursorOff** (GDC_ULONG *disp_no*,
GDC_UCHAR *numCursor*)

**Arguments**

| | | |
|---|---|---|
| *disp_no* | Specifications of display number | |
| | **GDC_DISPLAY_0** | Display #0 |
| | **GDC_DISPLAY_1** | Display #1 |
| *numCursor* | Cursor number (0 or 1) | |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**Target GDC**

MB86296S

**Description**

Sets the cursor not-display for a dual display mode.

Sets the display status after shifted to a dual display when you call this command at a single display mode. (The display at that time is not changed.) Changes the display status immediately when you call this command at a dual display mode.

### 6.4.27 GdcDispSetYCMatrix [Sets YCbCr to RGB Transformation Matrix]

**Interface**

GDC_BOOL **GdcDispSetYCMatrix** (const GDC_ULONG **matrix*)

**Arguments**

| | |
|---|---|
| *matrix* | Pointer to matrix values to be set |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86296S

**Description**

Specify the conversion parameter on displaying when converting the value from the YCbCr form to RGB form by the matrix form.

Specify the pointer to the array corresponding to the following 3x4 matrix M for matrix. Express elements (from a11 to a33) by signed fixed-point form (Sign=1bit, Int=2bit, Fraction=8bit, and total 11bit) with the 8bit signed fraction part.

Express elements (from b1 to b3) by signed integer form (Sign=1bit, Int=8bit, and total 9bit).

$$M=\begin{pmatrix} a11 & a12 & a13 & b1 \\ a21 & a22 & a23 & b2 \\ a31 & a32 & a33 & b3 \end{pmatrix}$$

# 6.5 Color Setting Commands

## 6.5.1 GdcColorPalette [Sets Palette Colors]

### Interface

GDC_BOOL **GdcColorPalette** (GDC_UCHAR *layer*, GDC_UCHAR *number*,
GDC_UCHAR *size*, const GDC_COL32 **lpColor*)

### Arguments

*layer* Layer selection

| | |
|---|---|
| **GDC_C_LAYER_PALETTE** | Layer C |
| **GDC_MB_LAYER_PALETTE** | Layer MB |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_L0_LAYER_PALETTE** | Layer L0 (same as layer C) |
| **GDC_L1_LAYER_PALETTE** | Layer L1 (same as layer MB) |
| **GDC_L2_LAYER_PALETTE** | Layer L2 |
| **GDC_L3_LAYER_PALETTE** | Layer L3 |

*number* Top palette number, range from "0" to "255"

*size* Number of palette, range from "0" to "255"

*lpColor* Pointer to the color data

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

### Target GDC

All

### Description

Sets color index code to palette table. If size is set to "0", all "256" entries of selected palette are set.

Setting of palette is available without regard to display mode. However, layer L2 and L3 palettes are available only in Extend Display Mode (only in MB86293 or later).

### 6.5.2 GdcColorTransparent [Sets Transparent Color]

**Interface**

GDC_BOOL **GdcColorTransparent** (GDC_UCHAR *layer*, GDC_COLOR32 *color*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_ML** | Layer ML |
| **GDC_DISP_LAYER_MR** | Layer MR |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*color* Transparent color code

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |

**Target GDC**

All

**Description**

Sets transparent color code.

Layer L0, L2, L3 are correspond to layer C, ML and MR.

The following values are used according to color mode of layer.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*
- 24-bit color mode: lower 24 bits of *color*

In order to use color code "0" (color palette code "0") as transparent color, set the transparent mode of color code "0" to **GDC_COLOR_TRANSPARENT** with the **GdcColorZeroMode** command.

Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the

application program, the result is not guaranteed.

[Macros for standard display]

- GDC_DISP_LAYER_C
- GDC_DISP_LAYER_ML
- GDC_DISP_LAYER_MR

[Macros for extended display]

- GDC_DISP_LAYER_L0
- GDC_DISP_LAYER_L1
- GDC_DISP_LAYER_L2
- GDC_DISP_LAYER_L3
- GDC_DISP_LAYER_L4
- GDC_DISP_LAYER_L5

### 6.5.3 GdcColorZeroMode [Sets Color Code "0" as Transparent Mode]

**Interface**

GDC_BOOL **GdcColorZeroMode** (GDC_UCHAR *layer*, GDC_UCHAR *mode*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_DISP_LAYER_C** | Layer C |
| **GDC_DISP_LAYER_ML** | Layer ML |
| **GDC_DISP_LAYER_MR** | Layer MR |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_DISP_LAYER_L0** | Layer L0 |
| **GDC_DISP_LAYER_L1** | Layer L1 |
| **GDC_DISP_LAYER_L2** | Layer L2 |
| **GDC_DISP_LAYER_L3** | Layer L3 |
| **GDC_DISP_LAYER_L4** | Layer L4 |
| **GDC_DISP_LAYER_L5** | Layer L5 |

*mode* Color code "0" mode

**GDC_COLOR_NOTRANSPARENT**
Not use as transparent color (default)

**GDC_COLOR_TRANSPARENT**
Use as transparent color

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

All

**Description**

Selects handling of color value "0" (palette number "0") from the following.

- Not use as transparent color, and use as normal color value (palette number)
- Use as transparent color

Either "Macros for standard display" or "Macros for extended display" can be used to specify a layer when MB86293 or later Graphics Controller is used. If both "Macros for standard display" and "Macros for extended display" are used in the application program, the result is not guaranteed.

[Macros for standard display]

- GDC_DISP_LAYER_C
- GDC_DISP_LAYER_ML
- GDC_DISP_LAYER_MR

[Macros for extended display]

- GDC_DISP_LAYER_L0
- GDC_DISP_LAYER_L1
- GDC_DISP_LAYER_L2
- GDC_DISP_LAYER_L3
- GDC_DISP_LAYER_L4
- GDC_DISP_LAYER_L5

### 6.5.4 GdcChromaKeyMode [Sets Chroma-key Mode]

**Interface**

GDC_BOOL **GdcChromaKeyMode** (GDC_UCHAR *mode*, GDC_UCHAR *source*)

**Arguments**

*mode* Chroma-key mode selection

| | |
|---|---|
| **GDC_ENABLE** | Chroma-key operation enable |
| **GDC_DISABLE** | Chroma-key operation disable (default) |

*source* Source key color selection

| | |
|---|---|
| **GDC_CHROMAKEY_C** | Target to be compared is layer C color |
| **GDC_CHROMAKEY_DISP** | Target to be compared is display color (default) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

All

**Description**

Sets whether chroma-key is operated. When chroma-key is operated, select target to be compared between layer C color and display color. When chroma-key is not operated, setting of target of key color to be compared has no meaning.

### 6.5.5 GdcColorKey [Sets Key Color for Chroma-key]

**Interface**

GDC_BOOL **GdcColorKey** (GDC_COL16 *color*)

**Arguments**

| | |
|---|---|
| *color* | Key color for chroma-key operation |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

All

**Description**

Sets the key color for chroma-key operation. The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.5.6 GdcColorPaletteOffset [Sets of Color Palette Offset]

**Interface**

GDC_BOOL **GdcColorPaletteOffset** (GDC_UCHAR *layer*, GDC_ULONG *sub_no*)

**Arguments**

*layer* Layer selection

| | |
|---|---|
| **GDC_C_LAYER_PALETTE** | Layer C |
| **GDC_MB_LAYER_PALETTE** | Layer MB |
| **GDC_L0_LAYER_PALETTE** | Layer L0, same as layer C |
| **GDC_L1_LAYER_PALETTE** | Layer L1, same as layer MB |
| **GDC_L2_LAYER_PALETTE** | Layer L2 |
| **GDC_L3_LAYER_PALETTE** | Layer L3 |

*sub_no* Sub-palette number, serial number of "16" divided parts of "256" palette, set in range from "0" to "15" and number more than "16" is not valid.

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_LAYER** | Invalid layer has been specified |
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

MB86293 or later

**Description**

Sets the sub-palette number of the color palette.

Sub-palette number is the serial number of "16" divided parts of "256" palettes.

By switching sub-palette, colors used for a display can be changed at once.

[Example of usage of the **GdcColorPaletteOffset** command]

Example for changing color set number to each sub-palette numbers is shown in Figure 6.5.6.

In this example, "16" sub-palettes have been already assigned to layer L0 palette.

**GdcColorPaletteOffset(GDC_L0_LAYER_PALETTE**, "sub-palette number")

| | Layer L0 Palette | |
|---|---|---|
| Sub-palette number 0 | Palette 0 | Palette 0 of sub-palette number 0 |
| | : | : |
| | : | : |
| | : | : |
| | Palette 15 | Palette 15 of sub-palette number 0 |
| Sub-palette number 1 | Palette 16 | Palette 0 of sub-palette number 1 |
| | : | : |
| | : | : |
| | : | : |
| | Palette 31 | Palette 15 of sub-palette number 1 |
| | : | : |
| | : | : |
| | : | : |
| Sub-palette number 15 | Palette 240 | Palette 0 of sub-palette number 15 |
| | : | : |
| | : | : |
| | : | : |
| | Palette 255 | Palette 15 of sub-palette number 15 |

**Figure 6.5.6 Example of Usage of GdcColorPaletteOffset Command**

# 6.6 Cursor Control Commands

## 6.6.1 GdcCursorAddress [Sets Cursor Pattern Memory Address]

### Interface

GDC_BOOL **GdcCursorAddress** (GDC_UCHAR *numCursor*, GDC_ULONG *ladrs*)

### Arguments

| | |
|---|---|
| *numCursor* | Cursor number ("0" or "1") |
| *ladrs* | Cursor pattern address, offset from top of the graphics memory |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

### Target GDC

All

### Description

Sets the address of the graphics memory where the cursor pattern is stored.

The dimension of a cursor pattern is 64 * 64 pixels. The color is 8-bit color only. Two cursors are available. Memory size required two cursor patterns are stored is 8192(= 64*64*1*2) bytes.

### 6.6.2 GdcCursorPattern [Sets Cursor Pattern]

**Interface**

GDC_BOOL **GdcCursorPattern** (GDC_UCHAR *numCursor*,
const GDC_COL8 **lpCursor*)

**Arguments**

| | |
|---|---|
| *numCursor* | Cursor number ("0" or "1") |
| *lpCursor* | Pointer to cursor pattern |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**Target GDC**

All

**Description**

Sets a cursor pattern. Transfer a cursor pattern data in main memory pointed via *lpCursor* to the graphics memory that address is specified by the **GdcCursorAddress** command.

Size of cursor pattern is fixed to 64*64 pixels.

### 6.6.3 GdcCursorDisplay [Controls Cursor Display]

**Interface**

GDC_BOOL **GdcCursorDisplay** (GDC_UCHAR *numCursor*,
GDC_UCHAR *enable*)

**Arguments**

| | | |
|---|---|---|
| *numCursor* | Cursor number ("0" or "1") | |
| *enable* | Cursor display enable/disable | |
| | **GDC_ENABLE** | Cursor enable |
| | **GDC_DISABLE** | Cursor disable (default) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

**Target GDC**

All

**Description**

Controls cursor display enable/disable.

### 6.6.4 GdcCursorPos [Sets Cursor Display Position]

**Interface**

GDC_BOOL **GdcCursorPos** (GDC_UCHAR *numCursor*,
GDC_USHORT *x*, GDC_USHORT *y*)

**Arguments**

| | |
|---|---|
| *numCursor* | Cursor number ("0" or "1") |
| *x* | x coordinate of cursor display position, range from "0" to "4095", x coordinate in upper left corner of cursor, default is 0 (*1) |
| *y* | y coordinate of cursor display position, range from "0" to "4095", y coordinate in upper left corner of cursor, default is 0 (*1) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**Target GDC**

All

**Description**

Sets display position of cursor.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.6.5 GdcCursorPriority [Sets Cursor Display Priority]

**Interface**

GDC_BOOL **GdcCursorPriority** (GDC_UCHAR *numCursor*, GDC_UCHAR *mode*)

**Arguments**

*numCursor* Cursor number ("0" or "1")

*mode* Cursor display priority mode

| | |
|---|---|
| **GDC_PRIORITY_C_LAYER** | Prioritizes layer C |
| **GDC_PRIORITY_CURSOR** | Prioritizes cursor (default) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_CURSOR_NUMBER**

Invalid cursor number has been specified

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

**Target GDC**

All

**Description**

Selects which is prioritized in display, layer C or cursor.

### 6.6.6 GdcCursorColorTransparent [Sets Cursor Transparent Color]

**Interface**

GDC_BOOL **GdcCursorColorTransparent** (GDC_COL8 *color*)

**Arguments**

| | |
|---|---|
| *color* | Color code which is used as transparent color |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

All

**Description**

Sets a transparent color code for cursor.

### 6.6.7 GdcCursorColorZeroMode [Sets Cursor Color Code "0" as Transparent Mode]

**Interface**

GDC_BOOL **GdcCursorColorZeroMode** (GDC_UCHAR *mode*)

**Arguments**

*mode* Color code "0" mode

**GDC_COLOR_NOTRANSPARENT**
Not use as transparent color

**GDC_COLOR_TRANSPARENT**
Use as transparent color (default)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_PARAMETER** | Invalid parameter has been specified |

**Target GDC**

All

**Description**

Selects the color option applied for color code "0" in cursor pattern. Color code "0" can be used as either transparent color or ordinary color code.

# 6.7 Display List Control Commands

## 6.7.1 XGdcSetDLBuf [Sets Current DL Buffer]

### Interface

GDC_BOOL **XGdcSetDLBuf** (GDC_CTX **drvctx*, GDC_ULONG *bufNo*)

### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *bufNo* | DL buffer number |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

**GDC_ERR_ILLEGAL_DL_BUF_NO** Illegal DL buffer number has been specified

### Target GDC

All

### Description

Sets current DL buffer to the DL buffer specified by *bufNo*.

Display list in DL buffer specified by *bufNo* is canceled.

### 6.7.2 XGdcQueryCurrentDLBuf [Queries Current DL Buffer]

#### Interface

GDC_ULONG **XGdcQueryCurrentDLBuf** (GDC_CTX **drvctx*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

#### Return value

The number of current DL buffer

#### Target GDC

All

#### Description

Returns the number of current DL buffer.

### 6.7.3 XGdcGetDLBufNum [Gets Number of DL Buffers]

#### Interface

GDC_ULONG **XGdcGetDLBufNum** (GDC_CTX **drvctx*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

#### Return value

The number of DL buffers

#### Target GDC

All

#### Description

Returns the total number of DL buffers.

### 6.7.4 XGdcGetDLBufInfo [Gets DL Buffer Structure Information]

#### Interface

GDC_DLBUF_STRUCT *XGdcGetDLBufInfo (GDC_CTX **drvctx*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

#### Return value

Pointer to DL buffer structure information

#### Target GDC

All

#### Description

Returns the pointer to DL buffer structure information.

### 6.7.5 XGdcFlush [Transfers Display List in Current DL Buffer]

#### Interface

GDC_ULONG **XGdcFlush** (GDC_CTX **drvctx*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

#### Return value

Bytes of transferred display list

#### Target GDC

All

#### Description

Transfers display list in current DL buffer to the Graphics Controller.

This command returns immediately without waiting for the transmission when DMA or local display list transfer is used. When transmission by CPU is used, this command returns after the transmission.

This command returns the number of transferred bytes of display list.

### 6.7.6 XGdcFlushEx [Transfers Display List in Specified DL Buffer]

**Interface**

GDC_ULONG **XGdcFlushEx** (GDC_CTX **drvctx*, GDC_ULONG *bufNo*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *bufNo* | The number of a DL buffer |

**Return value**

Bytes of transferred display list

**Error code**

**GDC_ERR_ILLEGAL_DL_BUF_NO** Illegal DL buffer number has been specified

**Target GDC**

All

**Description**

Transfers display list in the DL buffer specified by *bufNo* to the Graphics Controller.

This command returns immediately without waiting for the transmission when DMA or local display list transfer is used. When transmission by CPU is used, this command returns after the transmission.

This command returns the number of transferred bytes of display list

### 6.7.7 XGdcCancelDisplayList [Cancels Display List]

#### Interface

void **XGdcCancelDisplayList** (GDC_CTX **drvctx*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

#### Return value

None

#### Target GDC

All

#### Description

Cancels display list in current DL buffer.

# 6.8 Drawing Frame Setting Commands

## 6.8.1 XGdcDrawDimension [Sets Drawing Frame]

### Interface

GDC_BOOL **XGdcDrawDimension** (GDC_CTX **drvctx*,
GDC_UCHAR *cmode*, GDC_ULONG *dadrs*,
GDC_USHORT *dw*, GDC_USHORT *dh*)

### Arguments

*drvctx* Pointer to context

*cmode* Color mode

| | |
|---|---|
| **GDC_16BPP_FORMAT** | 16-bit color mode |
| **GDC_8BPP_FORMAT** | 8-bit color mode |

*dadrs* Drawing frame base address, specify offset from top of the graphics memory

*dw* Drawing frame width, in pixels (*1)
Range from "32" to "4096", 32 in pixels, in 16-bit color mode
Range from "64" to "4096", 64 in pixels, in 8-bit color mode

*dh* Drawing frame height, in pixels, range from "1" to "4096" (*1)

### Return value

**GDC_TRUE** Complete
**GDC_FALSE** Incomplete

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_COLOR_MODE**
Invalid color mode has been specified

**GDC_ERR_ILLEGAL_DIMENSION**
Illegal vertical/horizontal size

### Target GDC

All

### Description

Sets color mode, base address, and size of drawing frame.

(*1) The range check of the parameter is not performed, set a valid value. Error code **GDC_ERR_ILLEGAL_DIMENSION** is set, **GDC_FALSE** is set to the return value, and it returns when either *dw* or *dh* is "0".

### Notes

When the width of the display frame specified with the **GdcDispDimension** command doesn't match with the width of the drawing frame specified with this command, it is not correctly displayed.

### 6.8.2 XGdcSetZPrecision [Sets Precision of Z Value]

**Interface**

GDC_BOOL **XGdcSetZPrecision** (GDC_CTX **drvctx*, GDC_ULONG *mode*)

**Arguments**

| | | |
|---|---|---|
| *drvctx* | Pointer to context | |
| *mode* | Precision of z value | |
| | **GDC_Z_16BIT** | Precision of z value is 16-bit (default) |
| | **GDC_Z_8BIT** | Precision of z value is 8-bit |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets precision of z value.

### 6.8.3 XGdcBufferCreateZ [Sets Z-buffer Base Address]

**Interface**

GDC_BOOL **XGdcBufferCreateZ** (GDC_CTX **drvctx*, GDC_ULONG *zadrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *zadrs* | Z-buffer base address, specify offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets the base address of Z-buffer. The vertical/horizontal size of Z-buffer is assumed to be equal to that of drawing frame.

When precision of z value is 16-bit, memory size of 16-bit per 1 pixel is needed.

When precision of z value is 8-bit, memory size of 8-bit per 1 pixel is needed.

### 6.8.4 XGdcBufferCreateC [Sets Base Address of Polygon Drawing Control Buffer]

**Interface**

GDC_BOOL **XGdcBufferCreateC** (GDC_CTX **drvctx*, GDC_ULONG *cadrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *cadrs* | Polygon drawing control buffer base address, specify offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets the base address of polygon drawing control buffer. The vertical/horizontal size of this control buffer is assumed to be equal to that of drawing frame. For each pixel, 1 bit of data is required for this buffer.

### 6.8.5 XGdcBufferClearZ [Clears Z-buffer]

**Interface**

GDC_BOOL **XGdcBufferClearZ** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Clears Z-buffer for the size of the present drawing frame. It is necessary to clear Z-buffer before drawing by using this command when Z-buffer is used for hidden surface elimination.

**Notes**

Execute the **XGdcDrawDimension** command before the calling this command because size of Z-buffer is determined by the **XGdcDrawDimension** command.

### 6.8.6 XGdcBufferClearC [Clears Polygon Drawing Control Buffer]

**Interface**

GDC_BOOL **XGdcBufferClearC** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Clears polygon drawing control buffer.

**Notes**

Execute the **XGdcDrawDimension** command before the calling this command because size of polygon drawing control buffer is determined by the **XGdcDrawDimension** command.

### 6.8.7 XGdcDrawClipFrame [Sets Drawing Clip Border]

**Interface**

GDC_BOOL **XGdcDrawClipFrame** (GDC_CTX **drvctx*,
GDC_USHORT *x0*, GDC_USHORT *y0*,
GDC_USHORT *x1*, GDC_USHORT *y1*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x0* | x coordinate of left top edge of clip border, range from "0" to "4095" (*1) |
| *y0* | y coordinate of left top edge of clip border, range from "0" to "4095" (*1) |
| *x1* | x coordinate of right bottom edge of clip border, range from "0" to "4095" (*1) |
| *y1* | y coordinate of right bottom edge of clip border, range from "0" to "4095" (*1) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets clip border of drawing. Clip border is set as a Blt located relatively from the base point of drawing frame. Drawing to the area outside of this clip border is not performed.

(*1) The range check of the parameter is not performed, set a valid value.

### 6.8.8 XGdcSetAlphaMapBase [Sets Base Address of Alpha Map Area]

**Interface**

GDC_BOOL **XGdcSetAlphaMapBase** (GDC_CTX **drvctx*, GDC_ULONG *adrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *adrs* | Alpha map area base address, specify offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets the base address of alpha map area.

Alpha map area is an alpha blending coefficient area to be used by the **XGdcBltCopyAltAlpha** command. An offset from top of the graphics memory must be set to *adrs*.

For details about alpha map, refer to hardware specifications of the Graphics Controller.

# 6.9 Primitive Drawing Commands for Device Coordinate System

## 6.9.1 XGdcPrimType [Starts Drawing Procedure]

**Interface**

GDC_BOOL **XGdcPrimType** (GDC_CTX **drvctx*, GDC_UCHAR *type*)

**Arguments**

*drvctx* Pointer to context

*type* Primitive type

| | |
|---|---|
| **GDC_POINTS** | Points |
| **GDC_LINES** | Lines |
| **GDC_POLYLINE** | Polyline |
| **GDC_LINES_FAST** | Fast 2D lines |
| **GDC_POLYLINE_FAST** | Fast 2D polyline |
| **GDC_TRIANGLES** | Triangles |
| **GDC_TRIANGLE_STRIP** | Triangle strip |
| **GDC_TRIANGLE_FAN** | Triangle fan |
| **GDC_POLYGON** | Polygon |
| **GDC_TRIANGLES_FAST** | Fast 2D triangles |
| **GDC_TRIANGLE_STRIP_FAST** | Fast 2D triangle strip |
| **GDC_TRIANGLE_FAN_FAST** | Fast 2D triangle fan |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_NOT_READY** | Graphics Driver is not initialized |
| **GDC_ERR_INVALID_PRIMITIVE** | Invalid primitive has been specified |

**Target GDC**

All

**Description**

Sets the primitive type to be drawn by the **XGdcDrawVertex2D[i]** or the **XGdcDrawVertex3D[f]** command. Once either of these commands is executed, same type of primitive will keep being drawn till the **XGdcPrimEnd** command will be executed.

### 6.9.2 XGdcPrimEnd [Completes Drawing Procedure]

**Interface**

GDC_BOOL **XGdcPrimEnd** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Stops drawing the primitive applied by the **XGdcPrimType** command.

### 6.9.3 XGdcTexCoord2D[f/Nf] [Sets Coordinates of 2D Texture]

**Interface**

void **XGdcTexCoord2D** (GDC_CTX **drvctx*,
GDC_FIXED32 *u*, GDC_FIXED32 *v*)
void **XGdcTexCoord2Df** (GDC_CTX **drvctx*,
GDC_SFLOAT *u*, GDC_SFLOAT *v*)
void **XGdcTexCoord2DNf** (GDC_CTX * *drvctx*,
GDC_SFLOAT *u*, GDC_SFLOAT *v*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *u* | u coordinate of texture mapped on the vertex |
| *v* | v coordinate of texture mapped on the vertex |

**Return value**

None

**Target GDC**

All

**Description**

Sets the texture coordinates for the vertex to be drawn by the vertex coordinate setting command. Once this command is executed, the same texture coordinates is continuously applied till this command will be executed.

The **XGdcTexCoord2D** command must be used when the type of texture coordinates is GDC_FIXED32.

The **XGdcTexCoord2Df** command must be used when the type of texture coordinates is GDC_SFLOAT.

The **XGdcTexCoord2DNf** command must be used when the type of texture coordinates is GDC_SFLOAT and normalized. In this case, the range of texture coordinates must be within "0.0" to "1.0". The minimum size of texture coordinates is "0.0" and the maximum size is "1.0".

These commands are applicable to the following primitives. If these commands are used for except the following primitives, the result is not guaranteed.

**GDC_TRIANGLES**
**GDC_TRIANGLE_STRIP**
**GDC_TRIANGLE_FAN**

### 6.9.4 XGdcTexCoord3D[f/Nf] [Sets Coordinates of 3D Texture]

**Interface**

void **XGdcTexCoord3D** (GDC_CTX **drvctx*,
GDC_FIXED32 *u*, GDC_FIXED32 *v*, GDC_FIXED32 *rw*)

void **XGdcTexCoord3Df** (GDC_CTX * *drvctx*,
GDC_SFLOAT *u*, GDC_SFLOAT *v*, GDC_SFLOAT *rw*)

void **XGdcTexCoord3DNf** (GDC_CTX * *drvctx*,
GDC_SFLOAT *u*, GDC_SFLOAT *v*, GDC_SFLOAT *rw*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *u* | u coordinate of texture mapped on the vertex |
| *v* | v coordinate of texture mapped on the vertex |
| *rw* | Reciprocal of w coordinate of texture mapped on the vertex |

**Return value**

None

**Target GDC**

All

**Description**

Sets the texture coordinates for the vertex to be drawn by the vertex coordinate setting command. Once this command is executed, the same texture coordinates is continuously applied till this command will be executed.

The **XGdcTexCoord3D** command must be used when the type of texture coordinates is GDC_FIXED32.

The **XGdcTexCoord3Df** command must be used when the type of texture coordinates is GDC_SFLOAT.

The **XGdcTexCoord3DNf** command must be used when the type of texture coordinates is GDC_SFLOAT and normalized. In this case, the range of texture coordinates must be within "0.0" to "1.0". The minimum size of texture coordinates is "0.0" and the maximum size is "1.0".

These commands are applicable to the following primitives. If these commands are used for except the following primitives, the result is not guaranteed.

**GDC_TRIANGLES**
**GDC_TRIANGLE_STRIP**

### GDC_TRIANGLE_FAN

### 6.9.5 XGdcDrawVertex2D[i] [Sets Coordinates of 2D Vertex]

**Interface**

GDC_BOOL **XGdcDrawVertex2D** (GDC_CTX **drvctx*,
GDC_FIXED32 *x*, GDC_FIXED32 *y*)

GDC_BOOL **XGdcDrawVertex2Di** (GDC_CTX * *drvctx*,
GDC_LONG *x*, GDC_LONG *y*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate of 2D vertex |
| *y* | y coordinate of 2D vertex |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets 2D vertex coordinate and drawing a designated primitive.

Current values of color and texture coordinates are used in drawing, which has been set by the drawing color setting command and texture coordinates setting command respectively.

The **XGdcDrawVertex2D** command must be used when the type of vertex coordinate is GDC_FIXED32.

The **XGdcDrawVertex2Di** command must be used when the type of vertex coordinate is GDC_LONG.

The **XGdcDrawVertex2Di** command is applicable to the following primitives. If this command is used for except the following primitives, the result is not guaranteed.

**GDC_LINES_FAST**
**GDC_POLYLINE_FAST**
**GDC_POLYGON**

GDC_TRIANGLES_FAST
GDC_TRIANGLE_STRIP_FAST
GDC_TRIANGLE_FAN_FAST

### 6.9.6 XGdcDrawVertex3D[f] [Sets Coordinates of 3D Vertex]

**Interface**

GDC_BOOL **XGdcDrawVertex3D** (GDC_CTX **drvctx*,
GDC_FIXED32 *x*, GDC_FIXED32 *y*, GDC_USHORT *z*)

GDC_BOOL **XGdcDrawVertex3Df** (GDC_CTX * *drvctx*,
GDC_SFLOAT *x*, GDC_SFLOAT *y*, GDC_SFLOAT *z*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate of 3D vertex |
| *y* | y coordinate of 3D vertex |
| *z* | z coordinate of 3D vertex |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets 3D vertex coordinate and drawing a designated primitive.

Current values of color and texture coordinates are used in drawing, which has been set by drawing color setting command and texture coordinates setting command respectively.

The **XGdcDrawVertex3D** command must be used when the type of vertex coordinate is GDC_FIXED32.

The **XGdcDrawVertex3Df** command must be used when the type of vertex coordinate is GDC_SFLOAT.

The **XGdcDrawVertex3Df** command is applicable to the following primitives. If this command is used for except the following primitives, the result is not guaranteed.

**GDC_TRIANGLES**

**GDC_TRIANGLE_STRIP**

**GDC_TRIANGLE_FAN**

When drawing a polygon primitive (**GDC_POLYGON**), z coordinate of the parameter is ignored.

### 6.9.7 XGdcDrawPrimitive [Draws Multiple 3D Triangles]

**Interface**

GDC_BOOL **XGdcDrawPrimitive** (GDC_CTX **drvctx*, GDC_ULONG *type*,
const GDC_VERTEX **lpVertices*,
int *count*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *type* | Primitive type |
| | **GDC_TRIANGLES** Triangles |
| | **GDC_TRIANGLE_STRIP** Triangle strip |
| | **GDC_TRIANGLE_FAN** Triangle fan |
| *lpVertices* | Pointer to vertex parameter list (coordinates, color texture coordinates) |
| *count* | Number of vertices, "3" or more |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_VERTEX_COUNT**

Illegal number of vertex

**Target GDC**

All

**Description**

Draws a primitive specified in the type formed with multiple vertices designated by *lpVertices*.

# 6.10 Primitive Drawing Commands for Object Coordinate System

## 6.10.1 XGdcGeoPrimType [Starts Drawing Procedure]

### Interface

GDC_BOOL **XGdcGeoPrimType** (GDC_CTX **drvctx*, GDC_UCHAR *type*)

### Arguments

| | | |
|---|---|---|
| *drvctx* | Pointer to context | |
| *type* | Primitive type | |
| | **GDC_POINTS** | Points |
| | **GDC_LINES** | Lines |
| | **GDC_POLYLINE** | Polyline |
| | **GDC_TRIANGLES** | Triangles |
| | **GDC_TRIANGLE_STRIP** | Triangle strip |
| | **GDC_TRIANGLE_FAN** | Triangle fan |
| | **GDC_POLYGON** | Polygon |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

| | |
|---|---|
| **GDC_ERR_INVALID_PRIMITIVE** | Invalid primitive has been specified |

### Target GDC

MB86291 or later

### Description

Sets primitive drawn with the **XGdcGeoDrawVertex2D[f/i]** command or the **XGdcGeoDrawVertex3D[f/i]** command.

Once this command is executed, the same primitive is drawn until the **XGdcGeoPrimEnd** command is executed.

In MB86293 or later, to draw triangles with 8bit Gouraud shading, proceed in the following procedures. (The drawing of except in the following procedure is not guaranteed.)

(1) Set the following attributes before the calling of this command

- Set 4x4 matrix to convert from object coordinates to clip coordinates by the **XGdcGeoLoadMatrix[f]** command
- Set color mode to 8bpp by the **XGdcDrawDimension** command
- Set Gouraud shading (**GDC_SHADE_SMOOTH**) to 8bit shading mode (**GDC_8BIT_SHADE_MODE**) by the **XGdcSetAttrSurf** command
- Disable texture mapping/tiling (**GDC_SELECT_PLAIN**) by the **XGdcSetAttrSurf** command

(2) Call this command, specify any of the following for *type*

- **GDC_TRIANGLES**
- **GDC_TRIANGLE_STRIP**
- **GDC_TRIANGLE_FAN**

(3) Set 8 bits color of each vertex by the **XGdcVertexColor32** command

(4) Set coordinates of each vertex by the **XGdcGeoDrawVertex2D** command or the **XGdcGeoDrawVertex2Di** command

(5) Drawing is ended by the **XGdcGeoPrimEnd** command

(6) Set flat shading (**GDC_SHADE_FLAT**) to 8bit shading mode (**GDC_8BIT_SHADE_MODE**) by the **XGdcSetAttrSurf** command

**Notes**

After drawing triangles with 8bit shading mode set to Gouraud shading, restore flat shading to 8bit shading mode by the **XGdcSetAttrSurf** command.

In the state that 8bit shading mode is set to Gouraud shading, the following drawing attributes set by the **XGdcGeoSetAttrSurf** command become invalid.

- Top-left rule is not applied (top-left rule non-applied mode)
- Drawing shadow primitive

### 6.10.2 XGdcGeoPrimEnd [Completes Drawing Procedure]

**Interface**

GDC_BOOL **XGdcGeoPrimEnd** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_NOT_START_GEO_PRIM_TYPE**

The **XGdcGeoPrimEnd** command has called before the call of the **XGdcGeoPrimType** command and the **XGdcGeoDrawVertex** command

**GDC_ERR_NOT_START_GEO_DRAW_VERTEX**

The **XGdcGeoPrimEnd** command has called before the call of the **XGdcGeoDrawVertex** command

**Target GDC**

MB86291 or later

**Description**

Terminates a series of processes to draw primitives following the **XGdcGeoPrimType** command.

### 6.10.3 XGdcGeoDrawVertex2D[f/i] [Sets XY Coordinates of Vertex]

**Interface**

GDC_BOOL **XGdcGeoDrawVertex2D** (GDC_CTX **drvctx*,
GDC_FIXED32 *x*, GDC_FIXED32 *y*)

GDC_BOOL **XGdcGeoDrawVertex2Df** (GDC_CTX * *drvctx*,
GDC_SFLOAT *x*, GDC_SFLOAT *y*)

GDC_BOOL **XGdcGeoDrawVertex2Di** (GDC_CTX * *drvctx*,
GDC_LONG *x*, GDC_LONG *y*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate of the vertex |
| *y* | y coordinate of the vertex |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_NOT_START_GEO_PRIM_TYPE**

The **XGdcGeoPrimEnd** command has called before the call of the **XGdcGeoPrimType** command and the **XGdcGeoDrawVertex** command

**Target GDC**

MB86291 or later

**Description**

Specifies a vertex coordinate in object coordinates and drawing a primitive currently set. In this case, z is treated as "0".

Current values of color and texture coordinates are used in drawing, which has been set by drawing color setting command and texture coordinates setting command respectively.

The **XGdcGeoDrawVertex2D** command must be used when the type of vertex coordinate is GDC_FIXED32.

The **XGdcGeoDrawVertex2Df** command must be used when the type of vertex coordinate is GDC_SFLOAT.

The **XGdcGeoDrawVertex2Di** command must be used when the type of vertex

coordinate is GDC_LONG.

### 6.10.4 XGdcGeoDrawVertex3D[f/i] [Sets XYZ Coordinates of Vertex]

**Interface**

GDC_BOOL **XGdcGeoDrawVertex3D** (GDC_CTX **drvctx*,
GDC_FIXED32 *x*,GDC_FIXED32 *y*, GDC_FIXED32 *z*)

GDC_BOOL **XGdcGeoDrawVertex3Df** (GDC_CTX * *drvctx*,
GDC_SFLOAT *x*, GDC_SFLOAT *y*, GDC_SFLOAT *z*)

GDC_BOOL **XGdcGeoDrawVertex3Di** (GDC_CTX * *drvctx*,
GDC_LONG *x*, GDC_LONG *y*, GDC_FIXED32 *z*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate of the vertex |
| *y* | y coordinate of the vertex |
| *z* | z coordinate of the vertex |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_NOT_START_GEO_PRIM_TYPE**

The **XGdcGeoPrimEnd** command has been called before the call of the **XGdcGeoPrimType** command and the **XGdcGeoDrawVertex** command

**Target GDC**

MB86291 or later

**Description**

Sets vertex coordinate in object coordinates and drawing a primitive currently set. In this case, z is treated as zero.

Current values of color and texture coordinates are used in drawing, which has been set by drawing color setting command and texture coordinates setting command respectively.

The **XGdcGeoDrawVertex3D** command must be used when the type of vertex coordinate is GDC_FIXED32.

The **XGdcGeoDrawVertex3Df** command must be used when the type of vertex

coordinate is GDC_SFLOAT.

The **XGdcGeoDrawVertex3Di** command must be used when the type of vertex coordinate is GDC_LONG.

**Notes**

When drawing a polygon primitive (**GDC_POLYGON**) by MB86291/86292, z coordinate of the parameter is ignored.

### 6.10.5 XGdcGeoTexCoord2D[N/Nf] [Sets Texture Coordinates]

#### Interface

void **XGdcGeoTexCoord2DN** (GDC_CTX **drvctx*,
GDC_FIXED32 *u*, GDC_FIXED32 *v*)

void **XGdcGeoTexCoord2DNf** (GDC_CTX * *drvctx*,
GDC_SFLOAT *u*, GDC_SFLOAT *v*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *u* | Texture u coordinate of the vertex |
| *v* | Texture v coordinate of the vertex |

#### Return value

None

#### Target GDC

MB86291 or later

#### Description

Sets a texture coordinates (2 dimensions) of vertex in drawing with the vertex coordinate setting command. Once this command is executed, the same texture coordinates is used in drawing unless texture coordinates is changed by this command. This command treat texture coordinates as normalized ("1.0" is maximum size of current texture).

The **XGdcGeoTexCoord2DN** command must be used when the type of texture coordinates is GDC_FIXED32.

The **XGdcGeoTexCoord2DNf** command must be used when the type of texture coordinates is GDC_SFLOAT.

These commands are applicable to the following primitives. If these commands are used except the following primitives, the result is not guaranteed.

**GDC_TRIANGLES**
**GDC_TRIANGLE_STRIP**
**GDC_TRIANGLE_FAN**

However, MB86293 or later can also be used the following primitives:

**GDC_POLYGON**

### 6.10.6 XGdcVertexColor[32/3f] [Sets Color of Vertex]

#### Interface

void **XGdcVertexColor32** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

void **XGdcVertexColor3f** (GDC_CTX * *drvctx*,
GDC_SFLOAT *r*, GDC_SFLOAT *g*, GDC_SFLOAT *b*)

#### Arguments

*drvctx* Pointer to context

*color* [In 16-bit color mode]
Packed format in which each color element (r, g and b) is normalized to [0,255]. In this case, r, g and b are 8 bits respectively. (Figure 6.10.6a)

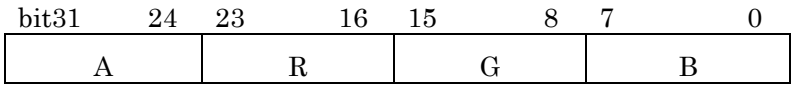



**Figure 6.10.6a 16-bit Vertex Color Format**

[In 8-bit color mode]
Specify 8-bit color code in the 16-23rd bit.

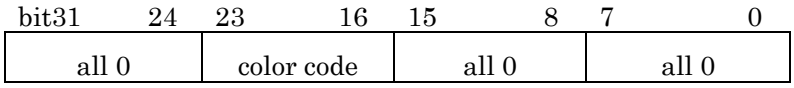



**Figure 6.10.6b 8-bit Vertex Color Format**

*r*, *g*, *b* Normalized values in which each color element (r, g, and b) is normalized to [0,1].

#### Return value

None

#### Target GDC

MB86291 or later, 8bit Gouraud shading is only MB86293 or later

#### Description

Sets a color of vertex. Once this command is executed, the same color is used in drawing for object coordinates unless the color is changed by this command.
When you work 8bit Gouraud shading, use the **XGdcVertexColor32** command. (The **XGdcVertexColor3f** command is not valid)
This command is used when shading mode is smooth shading. If the shading

mode is flat shading, use the **XGdcColor** command.

The **XGdcVertexColor32** command must be used when the type of vertex color is GDC_COLOR32.

The **XGdcVertexColor3f** command must be used when the type of vertex color is GDC_SFLOAT.

When drawing a polygon (**GDC_POLYGON**), setup of this command is ignored.

# 6.11 Drawing Attribute Setting Commands

## 6.11.1 XGdcColor [Sets Vertex Color/Foreground Color]

### Interface

GDC_BOOL **XGdcColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Vertex and foreground color |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

### Target GDC

All

### Description

Sets vertex color and foreground color applied for bitmap drawing and broken line drawing to be executed by set coordinates of vertex command. Once this command is executed, the same color is continuously applied till this command will be executed.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.11.2 XGdcBackColor [Sets Background Color]

**Interface**

GDC_BOOL **XGdcBackColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Background color |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets background color applied for binary pattern drawing and broken line drawing. Once this command is executed, the same color is continuously applied till this command will be executed.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

In order to make background color transparent, sets the following bit to "1" according to color mode.

- 8-bit color mode: bit15 of *color*
- 16-bit color mode: bit15 of *color*

### 6.11.3 XGdcClipMode [Sets Clipping Mode]

**Interface**

GDC_BOOL **XGdcClipMode** (GDC_CTX **drvctx*, GDC_ULONG *mode*)

**Arguments**

| | | |
|---|---|---|
| *drvctx* | Pointer to context | |
| *mode* | Clipping mode | |
| | **GDC_CLIP_X_ON** | Validates clipping toward x axis |
| | **GDC_CLIP_Y_ON** | Validates clipping toward y axis |
| | **GDC_CLIP_DISABLE** | Invalidates clipping |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets clipping mode.

**GDC_CLIP_X_ON** and **GDC_CLIP_Y_ON** can be set at the same time with OR operator.

### 6.11.4 XGdcSetAttrLine [Sets Line Drawing Attribute]

**Interface**

GDC_BOOL **XGdcSetAttrLine** (GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*drvctx* Pointer to context

*target* Line drawing attribute

| | |
|---|---|
| **GDC_DEPTH_TEST** | Z value compare mode |
| **GDC_DEPTH_FUNC** | Z value compare function |
| **GDC_DEPTH_WRITE_MASK** | Z value write permission mask |
| **GDC_BLEND_MODE** | Blending mode |
| **GDC_BROKEN_LINE** | Broken line mode |
| **GDC_LINE_WIDTH** | Line width |
| **GDC_ANTI_ALIAS** | Antialias option |
| **GDC_LINE_ENDPOINT** | End of the line control |
| **GDC_ROP_MODE** | Logical operation |

[In MB86291 or later, following macros are available]

**GDC_BROKEN_LINE_OFFSET**
Offset control of broken line pattern

**GDC_BROKEN_LINE_PERIOD**
Period set of broken line pattern

[In MB86293 or later, following macros are available]

**GDC_SHADOW_DEPTH_TEST**
Z value compare mode of shadow

**GDC_SHADOW_DEPTH_FUNC**
Z value compare function of shadow

**GDC_SHADOW_DEPTH_WRITE_MASK**
Z value write permission mask of shadow

**GDC_SHADOW_BLEND_MODE**
Blending mode of shadow

**GDC_SHADOW_BROKEN_LINE**
Broken line mode of shadow

**GDC_SHADOW_LINE_WIDTH**
Line width of shadow

**GDC_SHADOW_BROKEN_LINE_PERIOD**
Period set of broken line pattern of shadow

**GDC_SHADOW_ROP_MODE**
Logical operation of shadow

**GDC_BORDER_DEPTH_TEST**
Z value compare mode of border

**GDC_BORDER_DEPTH_FUNC**
Z value compare function of border

**GDC_BORDER_DEPTH_WRITE_MASK**
Z value write permission mask of border

**GDC_BORDER_BLEND_MODE**
Blending mode of border

**GDC_BORDER_BROKEN_LINE**
Broken line mode of border

**GDC_BORDER_LINE_WIDTH**
Line width of border

**GDC_BORDER_BROKEN_LINE_PERIOD**
Period set of broken line pattern of border

**GDC_BORDER_ROP_MODE**
Logical operation of border

*param* Parameter corresponding to *target* (*1)

**Return value**

**GDC_TRUE** Complete
**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_LINE_WIDTH**
Illegal width of line

**GDC_ERR_INVALID_ATTRIBUTE**
Invalid attribute has been specified

**Target GDC**

All

**Description**

Sets attribute for drawing lines.

**Notes**

When 8-bit color mode is used in drawing frame, logical operations do not use actual color (GDC_COL32 format color), but use color code (GDC_COL8 format). For example, when the pixel color (palette color code) of drawing frame is value "1" and drawing color (palette color code) is value "2", **GDC_ROP_OR** operation is stored value "3" in drawing frame.

(*1) Line drawing attribute (*target*) and parameter (*param*) corresponding to each line drawing attribute is shown below.

[Explanatory notes]

| Line drawing attribute | Description of line drawing attribute |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_DEPTH_TEST** | Sets z value compare mode. |
| **GDC_ENABLE** | Validates z value comparison. |
| **GDC_DISABLE** | Invalidates z value comparison. |
| **GDC_DEPTH_FUNC** | Selects z value comparison type. |
| **GDC_DEPTH_NEVER** | Always not drawn. |
| **GDC_DEPTH_ALWAYS** | Always drawn. |
| **GDC_DEPTH_LESS** | Drawn if current z value is less than Z-buffer value. |
| **GDC_DEPTH_LEQUAL** | Drawn if current z value equal to or less than Z-buffer value. |
| **GDC_DEPTH_EQUAL** | Drawn if current z value equal to Z-buffer value. |
| **GDC_DEPTH_GEQUAL** | Drawn if current z value equal to or greater than Z-buffer value. |
| **GDC_DEPTH_GREATER** | Drawn if current z value greater than Z-buffer value. |
| **GDC_DEPTH_NOTEQUAL** | Drawn if current z value is not equal to Z-buffer value. |

| | |
|---|---|
| **GDC_DEPTH_WRITE_MASK** | Enables write access to Z-buffer.<br>If **GDC_ENABLE**, according to the result of z value comparison, z value is written to Z-buffer. |
| **GDC_ENABLE** | Disable Z-buffer write. |
| **GDC_DISABLE** | Enable Z-buffer write. |
| **GDC_BLEND_MODE** | Sets blending mode of pixel write. |
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ALPHA** | Enables alpha blending. |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |
| **GDC_BROKEN_LINE** | Selects broken line mode. |
| **GDC_ENABLE** | Draws a broken line utilizing applied line pattern. |
| **GDC_DISABLE** | Draws a solid line. |
| **GDC_LINE_WIDTH** | Sets line width. |
| **GDC_LINE_WIDTH_1** | Draws a line of 1 pixel width. |
| **GDC_LINE_WIDTH_2** | Draws a line of 2 pixels width. |
| : | : |
| **GDC_LINE_WIDTH_32** | Draws a line of 32 pixels width. |
| **GDC_ANTI_ALIAS** | Sets antialias mode. |
| **GDC_ENABLE** | Enables antialias operation. |
| **GDC_DISABLE** | Disables antialias operation. |

| | |
|---|---|
| **GDC_LINE_ENDPOINT** | Controls the end point of line in **GDC_LINES** and **GDC_LINES_FAST** commands. End point is not drawn in **GDC_POLYLINE** and **GDC_POLYLINE_FAST** of the **XGdcPrimType** command regardless of this setting. In **GDC_POLYLINE** of the **XGdcGeoPrimType** command, this setting is available. |
| **GDC_ENABLE** | Draws the end point. |
| **GDC_DISABLE** | Not draws the end point. |
| **GDC_ROP_MODE** | Sets logical operation mode of the body |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1"<br>s: drawing color<br>d: destination color |

| | |
|---|---|
| **GDC_BROKEN_LINE_OFFSET** | Specifies the way of drawing broken line (only for MB86291 or later). |
| **GDC_ENABLE** | Starts new drawing broken line pattern. |
| **GDC_DISABLE** | Continues from the last drawing broken line pattern. |
| **GDC_BROKEN_LINE_PERIOD** | Sets broken line pattern period (only for MB86291 or later). |
| **GDC_BROKEN_LINE_32** | 32 bits period. |
| **GDC_BROKEN_LINE_24** | 24 bits period. |
| **GDC_SHADOW_DEPTH_TEST** | Sets z value compare mode of shadow (only for MB86293 or later). |
| **GDC_ENABLE** | Validates z value comparison. |
| **GDC_DISABLE** | Invalidates z value comparison. |
| **GDC_SHADOW_DEPTH_FUNC** | Selects z value comparison type of shadow (only for MB86293 or later). |
| **GDC_DEPTH_NEVER** | Always not drawn. |
| **GDC_DEPTH_ALWAYS** | Always drawn. |
| **GDC_DEPTH_LESS** | Drawn if current z value is less than Z-buffer value. |
| **GDC_DEPTH_LEQUAL** | Drawn if current z value equal to or less than Z-buffer value. |
| **GDC_DEPTH_EQUAL** | Drawn if current z value equal to Z-buffer value. |
| **GDC_DEPTH_GEQUAL** | Drawn if current z value equal to or greater than Z-buffer value. |
| **GDC_DEPTH_GREATER** | Drawn if current z value greater than Z-buffer value. |
| **GDC_DEPTH_NOTEQUAL** | Drawn if current z value is not equal to Z-buffer value. |

**GDC_SHADOW_DEPTH_WRITE_MASK**

Enables write access to Z-buffer of shadow (only for MB86293 or later).

| | |
|---|---|
| **GDC_ENABLE** | Disable Z-buffer write. |
| **GDC_DISABLE** | Enable Z-buffer write. |

**GDC_SHADOW_BLEND_MODE** Sets blending mode of pixel write of shadow (only for MB86293 or later).

| | |
|---|---|
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ALPHA** | Enables alpha blending. |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |

**GDC_SHADOW_BROKEN_LINE** Selects broken line mode of shadow (only for MB86293 or later).

| | |
|---|---|
| **GDC_ENABLE** | Draws a broken line utilizing applied line pattern. |
| **GDC_DISABLE** | Draws a solid line. |

**GDC_SHADOW_LINE_WIDTH** Sets line width of shadow (only for MB86293 or later).

| | |
|---|---|
| **GDC_LINE_WIDTH_1** | Draws a line of 1 pixel width. |
| **GDC_LINE_WIDTH_2** | Draws a line of 2 pixels width. |
| : | : |
| **GDC_LINE_WIDTH_32** | Draws a line of 32 pixels width. |

**GDC_SHADOW_BROKEN_LINE_PERIOD**

Sets broken line pattern period of shadow (only for MB86293 or later).

| | |
|---|---|
| **GDC_BROKEN_LINE_32** | 32 bits period. |
| **GDC_BROKEN_LINE_24** | 24 bits period. |

| | |
|---|---|
| **GDC_SHADOW_ROP_MODE** | Sets logical operation mode of the shadow. Same logical operation mode is applied to Shadow primitives and shadow composition primitives. In the portion with which shadow primitive and shadow composition primitive overlap, logical operation works between shadow and shadow composition. |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1"<br>s: drawing color<br>d: destination color |

| | |
|---|---|
| GDC_BORDER_DEPTH_TEST | Sets z value compare mode of border (only for MB86293 or later). |
| GDC_ENABLE | Validates z value comparison. |
| GDC_DISABLE | Invalidates z value comparison. |
| GDC_BORDER_DEPTH_FUNC | Selects z value comparison type of border (only for MB86293 or later). |
| GDC_DEPTH_NEVER | Always not drawn. |
| GDC_DEPTH_ALWAYS | Always drawn. |
| GDC_DEPTH_LESS | Drawn if current z value is less than Z-buffer value. |
| GDC_DEPTH_LEQUAL | Drawn if current z value equal to or less than Z-buffer value. |
| GDC_DEPTH_EQUAL | Drawn if current z value equal to Z-buffer value. |
| GDC_DEPTH_GEQUAL | Drawn if current z value equal to or greater than Z-buffer value. |
| GDC_DEPTH_GREATER | Drawn if current z value greater than Z-buffer value. |
| GDC_DEPTH_NOTEQUAL | Drawn if current z value is not equal to Z-buffer value. |
| GDC_BORDER_DEPTH_WRITE_MASK | Enables write access to Z-buffer of border (only for MB86293 or later). |
| GDC_ENABLE | Disable Z-buffer write. |
| GDC_DISABLE | Enable Z-buffer write. |
| GDC_BORDER_BLEND_MODE | Sets blending mode of pixel write of border (only for MB86293 or later). |
| GDC_BLEND_COPY | Regular drawing operation (writes pixel color to drawing frame). |
| GDC_BLEND_ALPHA | Enables alpha blending. |
| GDC_BLEND_ROP | Draws with logical arithmetic. |

| | |
|---|---|
| **GDC_BORDER_BROKEN_LINE** | Selects broken line mode of border (only for MB86293 or later). |
| **GDC_ENABLE** | Draws a broken line utilizing applied line pattern. |
| **GDC_DISABLE** | Draws a solid line. |
| **GDC_BORDER_LINE_WIDTH** | Sets line width of border (only for MB86293 or later). |
| **GDC_LINE_WIDTH_1** | Draws a line of 1 pixel width. |
| **GDC_LINE_WIDTH_2** | Draws a line of 2 pixels width. |
| : | : |
| **GDC_LINE_WIDTH_32** | Draws a line of 32 pixels width. |
| **GDC_BORDER_BROKEN_LINE_PERIOD** | Sets broken line pattern period of border (only for MB86293 or later). |
| **GDC_BROKEN_LINE_32** | 32 bits period. |
| **GDC_BROKEN_LINE_24** | 24 bits period. |

| | |
|---|---|
| **GDC_BORDER_ROP_MODE** | Sets logical operation mode of the border |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1"<br>s: drawing color<br>d: destination color |

### 6.11.5 XGdcSetAttrSurf [Sets Surface Drawing Attribute]

**Interface**

GDC_BOOL **XGdcSetAttrSurf** (GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*drvctx* Pointer to context

*target* Surface drawing attribute

| | |
|---|---|
| **GDC_SHADE_MODE** | Shading mode |
| **GDC_DEPTH_TEST** | Z value compare mode |
| **GDC_DEPTH_FUNC** | Z value compare type |
| **GDC_DEPTH_WRITE_MASK** | Z value write mask |
| **GDC_BLEND_MODE** | Blending mode |
| **GDC_TEXTURE_SELECT** | Texture mode |
| **GDC_ROP_MODE** | Logical operation mode |

[In MB86293 or later, following macros are available]

**GDC_8BIT_SHADE_MODE**

8bit shading mode

**GDC_ALPHA_SHADE_MODE**

Alpha shading mode

**GDC_SHADOW_DEPTH_TEST**

Z value compare mode of shadow

**GDC_SHADOW_DEPTH_FUNC**

Z value compare type of shadow

**GDC_SHADOW_DEPTH_WRITE_MASK**

Z value write mask of shadow

**GDC_SHADOW_BLEND_MODE**

Blending mode of shadow

**GDC_SHADOW_ROP_MODE**

Logical operation mode of shadow primitives

**GDC_NON_TOPLEFT_SHADE_MODE**

Shading mode of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_DEPTH_TEST**

Z value compare mode of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_DEPTH_FUNC**

Z value compare type of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_DEPTH_WRITE_MASK**

Z value write mask of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_BLEND_MODE**

Blending mode of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_TEXTURE_SELECT**

Texture mode of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_ALPHA_SHADE_MODE**

Alpha shading mode of top-left rule non-applied primitives

**GDC_NON_TOPLEFT_ROP_MODE**

Logical operation mode of top-left rule non-applied primitives

*param* Parameter corresponding to *target* (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**

Invalid attribute has been specified

**Target GDC**

All

**Description**

Sets attribute for surface drawing (except texture mapping attribute).

**Notes**

When drawing frame is 8-bit color mode, logical operations do not use actual color (GDC_COL32 format color), but use color code (GDC_COL8 format). For example, when the pixel color (palette color code) of drawing frame is value "1" and drawing color (palette color code) is value "2", **GDC_ROP_OR** operation is stored value "3" in drawing frame.

(*1) Surface drawing attribute (*target*) and parameter (*param*) corresponding to each surface drawing attribute is shown below.

[Explanatory notes]

| Surface drawing attribute | Description of surface drawing attribute |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_SHADE_MODE** | Sets shading mode. When setting 8bit shading mode, set by **GDC_8BIT_SHADE_MODE** macro. |
| **GDC_SHADE_FLAT** | Draws by flat shading. |
| **GDC_SHADE_SMOOTH** | Draws by Gouraud shading. |
| **GDC_DEPTH_TEST** | Sets z value compare mode. |
| **GDC_ENABLE** | Validate z value comparison. |
| **GDC_DISABLE** | Invalidate z value comparison. |
| **GDC_DEPTH_FUNC** | Selects z value comparison type. |
| **GDC_DEPTH_NEVER** | Always not drawn. |
| **GDC_DEPTH_ALWAYS** | Always drawn. |
| **GDC_DEPTH_LESS** | Drawn if current z value is less than Z-buffer value. |
| **GDC_DEPTH_LEQUAL** | Drawn if current z value is equal to or less than Z-buffer value. |
| **GDC_DEPTH_EQUAL** | Drawn if current z value is equal to Z-buffer value. |
| **GDC_DEPTH_GEQUAL** | Drawn if current z value is equal to or greater than Z-buffer value. |
| **GDC_DEPTH_GREATER** | Drawn if current z value is greater than Z-buffer value. |
| **GDC_DEPTH_NOTEQUAL** | Drawn if current z value is not equal to Z-buffer value. |
| **GDC_DEPTH_WRITE_MASK** | Enables write access to Z-buffer. If **GDC_ENABLE**, according to the result of z value comparison, z value is written to Z-buffer. |
| **GDC_ENABLE** | Disables Z-buffer write. |
| **GDC_DISABLE** | Enables Z-buffer write. |

| | |
|---|---|
| **GDC_BLEND_MODE** | Sets blending mode of pixel write. |
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ALPHA** | Enables alpha blending. |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |
| **GDC_TEXTURE_SELECT** | Sets texture mapping mode. |
| **GDC_SELECT_TEXTURE** | Draws with texture mapping. |
| **GDC_SELECT_TILE** | Draws with tiling. |
| **GDC_SELECT_PLAIN** | Invalidates texture mapping. |
| **GDC_ROP_MODE** | Sets logical operation mode of the body |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1"<br>s: drawing color<br>d: destination color |
| **GDC_8BIT_SHADE_MODE** | Sets 8bit shading mode. |
| **GDC_SHADE_FLAT** | Draws by flat shading. |
| **GDC_SHADE_SMOOTH** | Draws by Gouraud shading. In MB86293 or later, when drawing triangles by 8bit color mode, it is possible to draw with the value linear-interpolating 8bit color values specified by each vertex. |

| | |
|---|---|
| **GDC_ALPHA_SHADE_MODE** | Sets alpha shading mode. To works alpha shading, alpha-blending mode must be enabled (specify **GDC_BLEND_ALPHA** for **GDC_BLEND_MODE**) and shading mode must be enabled (specify **GDC_SHADE_SMOOTH** for **GDC_SHADE_MODE**). Alpha value is specified for each vertex by the **XGdcVertexColor32** command.<br>This mode is available in 16-bit color mode, only in MB86293 or later. |
| **GDC_SHADE_FLAT** | Works flat shading of alpha value. All pixels of the surface are blending by the same alpha value. |
| **GDC_SHADE_SMOOTH** | Works smooth shading of alpha value. Alpha values of each pixel are generated by interpolation between alpha values of each vertex. So, each pixel of the surface is blended individually. In addition, to works this function, also specify **GDC_SHADE_SMOOTH** for **GDC_SHADE_MODE**. |
| **GDC_SHADOW_DEPTH_TEST** | Sets z value compare mode of shadow (only for MB86293 or later). |
| **GDC_ENABLE** | Validate z value comparison. |
| **GDC_DISABLE** | Invalidate z value comparison. |
| **GDC_SHADOW_DEPTH_FUNC** | Selects z value comparison type of shadow (only for MB86293 or later). |
| **GDC_DEPTH_NEVER** | Always not drawn. |
| **GDC_DEPTH_ALWAYS** | Always drawn. |
| **GDC_DEPTH_LESS** | Drawn if current z value is less than Z-buffer value. |
| **GDC_DEPTH_LEQUAL** | Drawn if current z value is equal to or less than Z-buffer value. |
| **GDC_DEPTH_EQUAL** | Drawn if current z value is equal to Z-buffer value. |
| **GDC_DEPTH_GEQUAL** | Drawn if current z value is equal to or greater than Z-buffer value. |
| **GDC_DEPTH_GREATER** | Drawn if current z value is greater than Z-buffer value. |
| **GDC_DEPTH_NOTEQUAL** | Drawn if current z value is not equal to Z-buffer value. |

| | |
|---|---|
| **GDC_SHADOW_DEPTH_WRITE_MASK** | Enables write access to Z-buffer of shadow (only for MB86293 or later).<br>If **GDC_ENABLE**, according to the result of z value comparison, z value is written to Z-buffer. |
| **GDC_ENABLE** | Disables Z-buffer write. |
| **GDC_DISABLE** | Enables Z-buffer write. |
| **GDC_SHADOW_BLEND_MODE** | Sets blending mode of pixel write of shadow (only for MB86293 or later). |
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ALPHA** | Enables alpha blending. |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |

| | |
|---|---|
| **GDC_SHADOW_ROP_MODE** | Sets logical operation mode of the shadow. |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1" |

s: drawing color
d: destination color

| | |
|---|---|
| **GDC_NON_TOPLEFT_SHADE_MODE** | Sets shading mode of top-left rule non-applied primitive (only for MB86293 or later). |
| **GDC_SHADE_FLAT** | Flat shading. |
| **GDC_SHADE_SMOOTH** | Gouraud shading. |

| | |
|---|---|
| **GDC_NON_TOPLEFT_DEPTH_TEST** | Sets z value compare mode of top-left rule non-applied primitive (only for MB86293 or later). |
| **GDC_ENABLE** | Validate z value comparison. |
| **GDC_DISABLE** | Invalidate z value comparison. |

**GDC_NON_TOPLEFT_DEPTH_FUNC**

Selects z value comparison type of top-left rule non-applied primitive (only for MB86293 or later).

| | |
|---|---|
| **GDC_DEPTH_NEVER** | Always not drawn. |
| **GDC_DEPTH_ALWAYS** | Always drawn. |
| **GDC_DEPTH_LESS** | Drawn if current z value is less than Z-buffer value. |
| **GDC_DEPTH_LEQUAL** | Drawn if current z value is equal to or less than Z-buffer value. |
| **GDC_DEPTH_EQUAL** | Drawn if current z value is equal to Z-buffer value. |
| **GDC_DEPTH_GEQUAL** | Drawn if current z value is equal to or greater than Z-buffer value. |
| **GDC_DEPTH_GREATER** | Drawn if current z value is greater than Z-buffer value. |
| **GDC_DEPTH_NOTEQUAL** | Drawn if current z value is not equal to Z-buffer value. |

**GDC_NON_TOPLEFT_DEPTH_WRITE_MASK**

Enables write access to Z-buffer of top-left rule non-applied primitive (only for MB86293 or later).
If **GDC_ENABLE**, according to the result of z value comparison, z value is written to Z-buffer.

| | |
|---|---|
| **GDC_ENABLE** | Disables Z-buffer write. |
| **GDC_DISABLE** | Enables Z-buffer write. |

**GDC_NON_TOPLEFT_BLEND_MODE**

Sets blending mode of pixel write of top-left rule non-applied primitive (only for MB86293 or later).

| | |
|---|---|
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ALPHA** | Enables alpha blending. |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |

**GDC_NON_TOPLEFT_TEXTURE_SELECT**

Sets texture mapping mode of top-left rule non-applied primitive (only for MB86293 or later).

| | |
|---|---|
| **GDC_SELECT_TEXTURE** | Draws with texture mapping. |
| **GDC_SELECT_TILE** | Draws with tiling. |
| **GDC_SELECT_PLAIN** | Invalidates texture mapping. |

**GDC_NON_TOPLEFT_ALPHA_SHADE_MODE**

Sets alpha shading mode of top-left rule non-applied primitives.
To works this function, alpha-blending mode must be enabled (specify **GDC_BLEND_ALPHA** for **GDC_NON_TOPLEFT_BLEND_MODE**).
Alpha value is specified for each vertex by the **XGdcVertexColor32** command.
This mode is available in 16-bit color mode, only in MB86293 or later.

| | |
|---|---|
| **GDC_SHADE_FLAT** | Works flat shading of alpha value. All pixels of the surface are blending by the same alpha value. |
| **GDC_SHADE_SMOOTH** | Works smooth shading of alpha value. Alpha values of each pixel are generated by interpolation between alpha values of each vertex. So, each pixel of the surface is blended individually. In addition, to works this function, also specify **GDC_SHADE_SMOOTH** for **GDC_NON_TOPLEFT_SHADE_MODE**. |

**GDC_NON_TOPLEFT_ROP_MODE**

Sets logical operation mode of top-left rule non-applied primitives.

| | |
|---|---|
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1" |

s: drawing color
d: destination color

### 6.11.6 XGdcSetAttrTexture [Sets Texture Mapping Attribute]

**Interface**

GDC_BOOL **XGdcSetAttrTexture** (GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*drvctx* Pointer to context

*target* Texture mapping attribute

| | |
|---|---|
| **GDC_TEXTURE_PERSPECTIVE** | Perspective correction |
| **GDC_TEXTURE_FILTER** | Texture filter |
| **GDC_TEXTURE_WRAP_S** | S coordinate wrap |
| **GDC_TEXTURE_WRAP_T** | T coordinate wrap |
| **GDC_TEXTURE_BLEND** | Texture blend mode |
| **GDC_TEXTURE_ALPHA** | Texture alpha mode |

[In MB86293 or later, following macros are available]

| | |
|---|---|
| **GDC_TEXTURE_FAST_MODE** | Bi-linear fast mode |

*param* Parameter corresponding to *target* (*1)

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**

Invalid attribute has been specified

**Target GDC**

All

**Description**

Sets attribute for texture mapping.

(*1) Texture mapping attribute (*target*) and parameter (*param*) corresponding to each texture-mapping attribute is shown below.

[Explanatory notes]

| Texture mapping attribute | Description of texture mapping attribute |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_TEXTURE_PERSPECTIVE** | Selects perspective correction mode. |
| **GDC_ENABLE** | Validates perspective correction. |
| **GDC_DISABLE** | Invalidates perspective correction. |
| **GDC_TEXTURE_FILTER** | Selects texture filter mode. |
| **GDC_TEXTURE_POINT** | Point sampling mode. |
| **GDC_TEXTURE_BILINEAR** | Bi-linear filtering mode. |
| **GDC_TEXTURE_WRAP_S** | Defines S coordinate wrapping option when S coordinate value exceed the texture size. |
| **GDC_TEXTURE_REPEAT** | Repeats the texture pattern. |
| **GDC_TEXTURE_CLAMP** | Sets the most outside texture color. |
| **GDC_TEXTURE_BORDER** | Sets defined border color. |
| **GDC_TEXTURE_WRAP_T** | Sets T coordinate wrapping option when T coordinate value exceeds the texture size. |
| **GDC_TEXTURE_REPEAT** | Repeats the texture pattern. |
| **GDC_TEXTURE_CLAMP** | Sets the most outside texture color. |
| **GDC_TEXTURE_BORDER** | Sets defined border color. |
| **GDC_TEXTURE_BLEND** | Sets blending mode of texture color and polygon color.<br>This is applicable only when texture-mapping mode is selected. |
| **GDC_TEXTURE_DECAL** | Texture color is drawn. |
| **GDC_TEXTURE_MODULATE** | Blended color is drawn. |
| **GDC_TEXTURE_STENCIL** | If MSB of texture color is "1", texture color is drawn otherwise polygon color is drawn. |

| | |
|---|---|
| GDC_TEXTURE_ALPHA | Sets alpha blending mode between drawn color and current pixel color of the drawing frame. This is applicable only when texture mapping and alpha blending are selected. |
| GDC_TEXTURE_ALPHA_ALL | Alpha blending between post texture mapping color and current pixel color of the drawing frame. |
| GDC_TEXTURE_ALPHA_STENCIL | If MSB of texture color is "1", texture color is drawn, otherwise not drawn. |
| GDC_TEXALPHA_ALPHA_STENCILALPHA | If MSB of texture color is "1", alpha blending between texture color and current pixel color in the drawing frame is performed, otherwise not drawn. |
| GDC_TEXTURE_FAST_MODE | Sets bi-linear fast mode (for MB86293 or later except MB86295). |
| GDC_ENABLE | Texture mapping is executed at high speed by using a four times texture area as large as the default. |
| GDC_DISABLE | A default texture area is used. |

### 6.11.7 XGdcSetAttrBlt [Sets BitBlt Attribute]

**Interface**

GDC_BOOL **XGdcSetAttrBlt** (GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

| | | |
|---|---|---|
| *drvctx* | Pointer to context | |
| *target* | Bitmap drawing attribute | |
| | **GDC_BLEND_MODE** | Blend mode |
| | **GDC_TRANSPARENT_MODE** | Transparent mode (for MB86291 or later) |
| | **GDC_ROP_MODE** | Logical operation mode |
| *param* | Parameter corresponding to *target* (*1) | |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**

Invalid attribute has been specified

**Target GDC**

All (See Notes)

**Description**

Sets attribute when copying and drawing BitBlt.

**Notes**

- Transparent mode is available in MB86291 or later. Transparent mode is valid when drawing by the following Driver Command.
  - **XGdcBltCopy**
  - **XGdcBltCopyAlt[Sync]**
  - **XGdcBltDraw[8/16]**
  - **XGdcBitPatternDraw**
  - **XGdcBitPatternDrawByte**
- The logical operation function is invalid at the transparent mode.
- Specify the transparent color by the **XGdcBltColorTransparent** command.

- When drawing frame is 8-bit color mode, logical operations do not use actual color (GDC_COL32 format color), but use color code (GDC_COL8 format). For example, when the pixel color (palette color code) of drawing frame is value "1" and drawing color (palette color code) is value "2", **GDC_ROP_OR** operation is stored value "3" in drawing frame.

(*1) Bitmap drawing attribute (*target*) and parameter (*param*) corresponding to each bitmap drawing attributes are shown below.

[Explanatory notes]

| Bitmap drawing attribute | Description of bitmap drawing attributes |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_BLEND_MODE** | Sets blend mode. |
| **GDC_BLEND_COPY** | Regular drawing operation (writes pixel color to drawing frame). |
| **GDC_BLEND_ROP** | Draws with logical arithmetic. |
| **GDC_TRANSPARENT_MODE** | Sets transparent mode (only for MB86291 or later). |
| **GDC_ENABLE** | The color that is set by the **XGdcBltColorTransparent** command regards as transparent color. |
| **GDC_DISABLE** | The color that is set by the **XGdcBltColorTransparent** command is not treated as transparent color. |

| | |
|---|---|
| **GDC_ROP_MODE** | Sets logical operation mode of the body |
| **GDC_ROP_CLEAR** | Sets all bits to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | Sets all bits to "1"<br>s: drawing color<br>d: destination color |

### 6.11.8 XGdcSetAlpha [Sets Alpha Blending Coefficient]

**Interface**

GDC_BOOL **XGdcSetAlpha** (GDC_CTX **drvctx*, GDC_UCHAR *alpha*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *alpha* | Alpha blending coefficient, range from "0" to "255" |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets the ratio of alpha blending.

Transmissivity is 0%, when *alpha* is "0". And transmissivity is 100% when *alpha* is "255".

### 6.11.9 XGdcSetLinePattern [Sets Broken Line Pattern]

**Interface**

GDC_BOOL **XGdcSetLinePattern** (GDC_CTX **drvctx*, GDC_ULONG *pattern*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *pattern* | Broken line pattern |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets broken line pattern with 32 bits pattern when drawing broken line.

Upper 24 bits pattern values are set as a broken line pattern, when the period of broken line pattern is 24 bits.

Broken line is drawn with foreground color when the bit of pattern is "1", and drawn with background color when the bit is "0".

Example of broken line with broken line pattern "0xaaaaaaaa" is shown in Figure 6.11.9.

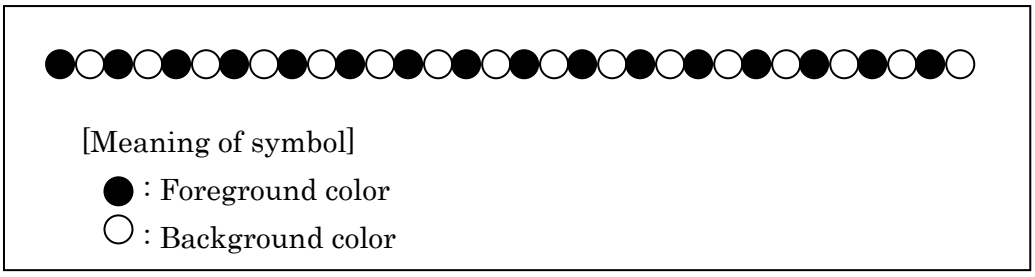



**Figure 6.11.9 Example of Broken Line**

### 6.11.10 XGdcSetTextureBorder [Sets Texture Border Color]

#### Interface

GDC_BOOL **XGdcSetTextureBorder**(GDC_CTX **drvctx*, GDC_COLOR32 *color*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Texture border color |

#### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

#### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

#### Target GDC

All

#### Description

Sets the border color of the texture applied in border mode of texture wrap.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.11.11 XGdcSetRop [Sets Logical Operation Mode]

#### Interface

GDC_BOOL **XGdcSetRop**(GDC_CTX **drvctx*, GDC_UCHAR *mode*)

#### Arguments

*drvctx* Pointer to context

*mode* Logical arithmetic mode

| | |
|---|---|
| **GDC_ROP_CLEAR** | All bits are set to "0" |
| **GDC_ROP_AND** | s & d |
| **GDC_ROP_AND_REVERSE** | s & !d |
| **GDC_ROP_COPY** | s |
| **GDC_ROP_AND_INVERTED** | !s & d |
| **GDC_ROP_NOP** | d |
| **GDC_ROP_XOR** | s ^ d |
| **GDC_ROP_OR** | s \| d |
| **GDC_ROP_NOR** | !(s \| d) |
| **GDC_ROP_EQUIV** | !(s ^ d) |
| **GDC_ROP_INVERT** | !d |
| **GDC_ROP_OR_REVERSE** | s \| !d |
| **GDC_ROP_COPY_INVERTED** | !s |
| **GDC_ROP_OR_INVERTED** | !s \| d |
| **GDC_ROP_NAND** | !(s & d) |
| **GDC_ROP_SET** | All bits are set to "1" |
| | s: drawing value |
| | d: destination value |

#### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

#### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

#### Target GDC

All

#### Description

Sets logical operation for all drawing processes.

This operation is performed between the pixel color to be drawn and current pixel color in the drawing frame. Result of this operation is to be drawn to the drawing frame.

This operation is applicable only when drawing attribute of line or surface or BitBlt's **GDC_BLEND_ROP** option of **GDC_BLEND_MODE** is selected.

To set logical operation mode individually for lines, surfaces and rectangles, use the **XGdcSetAttrLine** command, the **XGdcSetAttrSurf** command, and the **XGdcSetAttrBlt** command, respectively.

**Notes**

- In MB86293 or later, keep in mind that it will perform logical arithmetic in the shadows when the primitive between shadow composition overlaps with the shadow, since the same logical arithmetic mode as the shadow of a line and the primitive between shadow composition is applied if the logical arithmetic mode of the shadow is set up.
- In MB86293 or later, same logical operation is applied to shadow primitives and shadow composition primitives. Therefore, in the portion with which shadow primitive and shadow composition primitive overlap, logical operation works between shadow and shadow composition.
  When 8-bit color mode is used in drawing frame, logical operations do not use actual color (GDC_COL32 format color), but use color code (GDC_COL8 format). For example, when the pixel color (palette color code) of drawing frame is value "1" and drawing color (palette color code) is value "2", **GDC_ROP_OR** operation is stored value "3" in drawing frame.

# 6.12 Attribute Setting Commands for Object Coordinate System

## 6.12.1 XGdcGeoSetAttrLine [Sets Line Drawing Attribute for Object Coordinate System]

**Interface**

GDC_BOOL **XGdcGeoSetAttrLine**(GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*drvctx* Pointer to context

*target* Line drawing attribute

**GDC_GEO_THICK_LINE_CORRECT**
Sets correction mode of thick line connection

**GDC_GEO_BROKEN_LINE_CORRECT**
Sets correction mode of broken line pattern

**GDC_GEO_BROKEN_LINE_CORRECT_LENGTH**
Sets pixel number of fixed address broken line pattern

**GDC_GEO_UNIFORM_LINE_WIDTH**
Sets uniform mode of line width

**GDC_GEO_THICK_LINE_VERTICAL**
Sets thick/broken line vertical mode

**GDC_GEO_BORDER_LINE**
Sets drawing mode of border primitive

**GDC_GEO_SHADOW_MODE**
Sets drawing mode of shadow primitive

*param* Parameter corresponding to *target* (*1)

**Return value**

**GDC_TRUE** Complete
**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**
Invalid attribute has been specified

**Target GDC**

MB86293 or later

**Description**

Sets drawing attributes for object coordinate system line primitives.

Table 6.12.1 shows available combinations of drawing attributes.

**Notes**

Drawing attributes which have been set by this command effect on lines which no z coordinate are specified. Therefore, this command effect on only lines that vertices are specified the **XGdcGeoDrawVertex2D[f/i]** command.

(*1) Line drawing attribute (*target*) and parameter (*param*) corresponding to each line drawing attribute is shown below.

[Explanatory notes]

| Line drawing attribute | Description of line drawing attribute |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |

**GDC_GEO_THICK_LINE_CORRECT**

Sets thick line connection correct mode.

| | |
|---|---|
| **GDC_ENABLE** | Enables thick line connection correct. |
| **GDC_DISABLE** | Disables thick line connection correct. |

**GDC_GEO_BROKEN_LINE_CORRECT**

Sets broken line connection correct mode.

| | |
|---|---|
| **GDC_ENABLE** | Refer to the same broken line pattern with front and back number pixel of broken line connection part ("broken line pattern address fixation mode"). Number of pixel is set by **GDC_GEO_BROKEN_LINE_CORRECT_LENGTH**. |
| **GDC_DISABLE** | Not correct broken line pattern. |

**GDC_GEO_BROKEN_LINE_CORRECT_LENGTH**

Sets the pixel number of fixed address broken line pattern. A recommended value is the same as line width.
This parameter is available when the correction mode of broken line pattern is "broken line pattern address fixation mode".

| | |
|---|---|
| "0" to "32" | Number of pixels. |

**GDC_GEO_UNIFORM_LINE_WIDTH**

Sets uniform mode of line width.

| | |
|---|---|
| **GDC_ENABLE** | Enables uniform of line width.<br>When thick/broken line perpendicular to ideal line is drawn, this mode must be specified. |
| **GDC_DISABLE** | Disables uniform of line width. |

**GDC_GEO_THICK_LINE_VERTICAL**

Sets thick/broken line vertical mode.

| | |
|---|---|
| **GDC_ENABLE** | Draws perpendicular section of thick/broken line to an ideal line. |
| **GDC_DISABLE** | Draws perpendicular section of thick/broken line to a base axis. |

| | |
|---|---|
| **GDC_GEO_BORDER_LINE** | Sets drawing mode of border primitive. |
| **GDC_ENABLE** | Draws border primitive. For the border color, the color specified by the **XGdcGeoBorderColor** command is used. |
| **GDC_DISABLE** | Not draw border primitive. |
| **GDC_GEO_SHADOW_MODE** | Sets drawing mode of shadow primitive. |
| **GDC_ENABLE** | Draws shadow primitive. For the shadow color, the color specified by the **XGdcGeoShadowColor** command is used. |
| **GDC_DISABLE** | Not draw shadow primitive. |

**Table 6.12.1 Combinations of Drawing Attributes for Lines**

| | | GDC_GEO_THICK_LINE_VERTICAL | |
|---|---|---|---|
| | | GDC_ENABLE | GDC_DISABLE |
| GDC_GEO_THICK_LINE_CORRECT | GDC_ENABLE | Y | N |
| | GDC_DISABLE | Y | Y |
| GDC_GEO_UNIFORM_LINE_WIDTH | GDC_ENABLE | Y | N |
| | GDC_DISABLE | N | Y |
| GDC_GEO_BORDER_LINE | GDC_ENABLE | Y | N |
| | GDC_DISABLE | Y | Y |
| GDC_GEO_SHADOW_MODE | GDC_ENABLE | Y | N |
| | GDC_DISABLE | Y | Y |

Y: available N: not available (no guarantee the result)

### 6.12.2 XGdcGeoSetAttrSurf [Sets Surface Drawing Attribute for Object Coordinate System]

**Interface**

GDC_BOOL **XGdcGeoSetAttrSurf** (GDC_CTX **drvctx*,
GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*drvctx* Pointer to context

*target* Surface drawing attribute

**GDC_GEO_FACE_CULL**

Enable/disable culling back face of triangle

**GDC_GEO_FACE_INVERT**

Specify direction of surface of triangle

[In MB86293 or later, following macros are available]

**GDC_GEO_NON_TOPLEFT**

Sets drawing algorithm

**GDC_GEO_SHADOW_MODE**

Sets drawing mode of shadow primitive

*param* Parameter corresponding to *target* (*1)

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**

Invalid attribute has been specified

**Target GDC**

MB86291 or later

**Description**

Sets surface drawing attribute in object coordinate system.

Culling back face of triangle and specify direction of surface of triangle doesn't affect polygons (**GDC_POLYGON**).

(*1) Surface drawing attribute (*target*) and parameter (*param*) corresponding to each surface drawing attribute is shown below.

[Explanatory notes]

| Surface drawing attribute | Description of surface drawing attribute |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_GEO_FACE_CULL** | Specifies culling back face of triangle. |
| **GDC_ENABLE** | Enables culling back face of triangle. |
| **GDC_DISABLE** | Disables culling back face of triangle. |
| **GDC_GEO_FACE_INVERT** | Specifies direction of surface of triangle. Counterclockwise surface is front facing by default. |
| **GDC_ENABLE** | Invert direction of surface from default. |
| **GDC_DISABLE** | Direction of surface is default. |
| **GDC_GEO_NON_TOPLEFT** | Sets drawing algorithm (for MB86293 or later). |
| **GDC_ENABLE** | Non top-left applying rule is used. |
| **GDC_DISABLE** | Non top-left applying rule is not used. |
| **GDC_GEO_SHADOW_MODE** | Sets drawing mode of shadow primitive (for MB86293 or later). |
| **GDC_ENABLE** | Draws shadow primitive. For the shadow color, the color specified by the **XGdcGeoShadowColor** command is used. |
| **GDC_DISABLE** | Not draw shadow primitive. |

### 6.12.3 XGdcGeoLoadMatrix[f] [Sets Matrix]

**Interface**

GDC_BOOL **XGdcGeoLoadMatrix** (GDC_CTX **drvctx*,
const GDC_FIXED32 **ptMatrix*)

GDC_BOOL **XGdcGeoLoadMatrixf** (GDC_CTX **drvctx*,
const GDC_SFLOAT **ptMatrix*)

**Arguments**

*drvctx* Pointer to context

*ptMatrix* A pointer to an array {m1, m2, m3, …, m16} which corresponds to the 4x4 matrix M such as,

$$M=\begin{pmatrix} m1 & m5 & m9 & m13 \\ m2 & m6 & m10 & m14 \\ m3 & m7 & m11 & m15 \\ m4 & m8 & m12 & m16 \end{pmatrix}$$

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets a 4x4 matrix that transforms an object coordinates to a clip coordinates.
Each element in the matrix is put in the following order.

$$M=\begin{pmatrix} m1 & m5 & m9 & m13 \\ m2 & m6 & m10 & m14 \\ m3 & m7 & m11 & m15 \\ m4 & m8 & m12 & m16 \end{pmatrix}$$

Elements (m4, m8, m12, m16) in the matrix specify whether the projection type is orthographic or perspective. Therefore the projection type is set automatically by

the result of their values.

If (m4, m8, m12, m16) == (0,0,0,1), then orthographic projection.

Else if (m4, m8, m12, m16) != (0,0,0,1) then perspective projection.

The **XGdcGeoLoadMatrix** command must be used when the element of the matrix is GDC_FIXED32 type.

The **XGdcGeoLoadMatrixf** command must be used when the element of the matrix is GDC_SFLOAT type.

### 6.12.4 XGdcGeoNdcDcViewportCoef[f] [Sets Coefficients of NdcDc Transformation for XY]

**Interface**

GDC_BOOL **XGdcGeoNdcDcViewportCoef** (GDC_CTX **drvctx*,
GDC_FIXED32 *scalex*, GDC_FIXED32 *offsetx*,
GDC_FIXED32 *scaley*, GDC_FIXED32 *offsety*)

GDC_BOOL **XGdcGeoNdcDcViewportCoeff** (GDC_CTX **drvctx*,
GDC_SFLOAT *scalex*, GDC_SFLOAT *offsetx*,
GDC_SFLOAT *scaley*, GDC_SFLOAT *offsety*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *scalex* | Magnification in direction of x coordinate |
| *offsetx* | Offset in direction of x coordinate |
| *scaley* | Magnification in direction of y coordinate |
| *offsety* | Offset in direction of y coordinate |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the magnifications and offsets in direction of x, y coordinate that is used for transforming Normalized Device Coordinates (NDC) to Device Coordinates (DC).

The **XGdcGeoNdcDcViewportCoef** command must be used when the argument is GDC_FIXED32 type.

The **XGdcGeoNdcDcViewportCoeff** command must be used when the argument is GDC_SFLOAT type.

### 6.12.5 XGdcGeoNdcDcDepthCoef[f] [Sets Coefficients of NdcDc Transformation for Z]

**Interface**

GDC_BOOL **XGdcGeoNdcDcDepthCoef** (GDC_CTX **drvctx*,
GDC_FIXED32 *scalez*, GDC_FIXED32 *offsetz*)

GDC_BOOL **XGdcGeoNdcDcDepthCoeff** (GDC_CTX **drvctx*,
GDC_SFLOAT *scalez*, GDC_SFLOAT *offsetz*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *scalez* | Magnification of z |
| *offsetz* | Offset of z |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the magnification and offset of z that is used for transforming Normalized Device Coordinates (NDC) to Device Coordinates (DC).

The **XGdcGeoNdcDcDepthCoef** command must be used when the argument is GDC_FIXED32 type.

The **XGdcGeoNdcDcDepthCoeff** command must be used when the argument is GDC_SFLOAT type.

### 6.12.6 XGdcGeoViewVolumeXYClip[f] [Sets View Volume Boundary for XY]

**Interface**

GDC_BOOL **XGdcGeoViewVolumeXYClip** (GDC_CTX **drvctx*,
GDC_FIXED32 *xmin*, GDC_FIXED32 *xmax*,
GDC_FIXED32 *ymin*, GDC_FIXED32 *ymax*)

GDC_BOOL **XGdcGeoViewVolumeXYClipf** (GDC_CTX **drvctx*,
GDC_SFLOAT *xmin*, GDC_SFLOAT *xmax*,
GDC_SFLOAT *ymin*, GDC_SFLOAT *ymax*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *xmin* | Minimum clip value of x |
| *xmax* | Maximum clip value of x |
| *ymin* | Minimum clip value of y |
| *ymax* | Maximum clip value of y |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the view volume boundary in the clip coordinates for x, y.

The **XGdcGeoViewVolumeXYClip** command must be used when the argument is GDC_FIXED32 type.

The **XGdcGeoViewVolumeXYClipf** command must be used when the argument is GDC_SFLOAT type.

### 6.12.7 XGdcGeoViewVolumeZClip[f] [Sets View Volume Boundary for Z]

**Interface**

GDC_BOOL **XGdcGeoViewVolumeZClip** (GDC_CTX **drvctx*,
GDC_FIXED32 *zmin*, GDC_FIXED32 *zmax*)

GDC_BOOL **XGdcGeoViewVolumeZClipf** (GDC_CTX **drvctx*,
GDC_SFLOAT *zmin*, GDC_SFLOAT *zmax*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *zmin* | Minimum clip value of z |
| *zmax* | Maximum clip value of z |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the view volume boundary in the clip coordinates for z.

The **XGdcGeoViewVolumeZClip** command must be used when the argument is GDC_FIXED32 type.

The **XGdcGeoViewVolumeZClipf** command must be used when the argument is GDC_SFLOAT type.

### 6.12.8 XGdcGeoViewVolumeWminClip[f] [Sets View Volume Boundary for W]

**Interface**

GDC_BOOL **XGdcGeoViewVolumeWminClip** (GDC_CTX **drvctx*,
GDC_FIXED32 *wmin*)

GDC_BOOL **XGdcGeoViewVolumeWminClipf** (GDC_CTX **drvctx*,
GDC_SFLOAT *wmin*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *wmin* | Minimum clip value of w |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the view volume boundary in the clip coordinates for w.

As the front clip face (minimum clipping in z coordinate) closes with the viewpoint limitlessly, w also approximates to "0" limitlessly.

Since w is used to calculate "1/w" internally, *wmin* must be the one that does not occur overflow in division.

w has only minimum value. *wmin* is not minus value.

The **XGdcGeoViewVolumeWminClip** command must be used when the argument is GDC_FIXED32 type.

The **XGdcGeoViewVolumeWminClipf** command must be used when the argument is GDC_SFLOAT type.

### 6.12.9 XGdcGeoSetLogOutBase [Sets Base Address for Log Output of Device Coordinates]

**Interface**

GDC_BOOL **XGdcGeoSetLogOutBase** (GDC_CTX **drvctx*, GDC_ULONG *adrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *adrs* | Base address for log output of device coordinates, offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

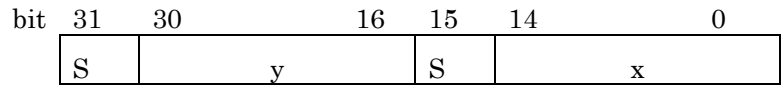
MB86293 or later

**Description**

Sets base address for log output of device coordinates.

Specify *adrs* for offset address from top of the graphics memory.

Log format consists of packed number of x and y coordinate of vertex (refer to Figure 6.12.9).




S: signed bit

y: y coordinate values (integer)

x: x coordinate values (integer)

**Figure 6.12.9 Log Format of Device Coordinates**

Log needs 4 bytes for every vertex.

Therefore, memory area must be allocated with size of number of vertex * 4 byte.

These vertices are specified between the **XGdcGeoPrimType** command and the **XGdcGeoPrimEnd** command.

Each time a log is outputted, the address for log output is added 4 bytes.

If this memory area is used repeatedly, sets the base address for log output again. The address for log output must be set after drawing is finished by "drawing command end interruption" or the **GdcGetPixelEngineStatus** command or the **GdcGeoGetPixelEngineStatus** command.

### 6.12.10 XGdcGeoSetLogOutMode [Sets Log Output Mode of Device Coordinates]

#### Interface

void **XGdcGeoSetLogOutMode** (GDC_CTX **drvctx*, GDC_ULONG *mode*)

#### Arguments

*drvctx* Pointer to context

*mode* Log output mode of device coordinates

**GDC_GEO_LOGOUT_ENABLE**

Outputs logs

**GDC_GEO_LOGOUT_DISABLE**

Not output logs (default)

**GDC_GEO_LOGOUT_ONLY**

Output logs without drawing

#### Return value

None

#### Target GDC

MB86293 or later

#### Description

Sets log output mode of device coordinates. Each mode is explained below.

- **GDC_GEO_LOGOUT_ENABLE**
  Log is outputted and drawing is executed.

- **GDC_GEO_LOGOUT_DISABLE**
  Log is not outputted and drawing is executed.

- **GDC_GEO_LOGOUT_ONLY**
  Log is outputted and drawing is not executed.
  This mode is available only when drawing point primitive. When **GDC_GEO_LOGOUT_ONLY** is specified for drawing other primitives, log is not outputted.

### 6.12.11 XGdcGeoShadowXY [Sets XY Offset of Shadow]

**Interface**

GDC_BOOL **XGdcGeoShadowXY** (GDC_CTX **drvctx*,
GDC_ULONG *type*, GDC_LONG *offsetx*, GDC_LONG *offsety*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *type* | Kind of primitive<br>**GDC_GEO_SHADOW**<br>Shadow primitive<br>**GDC_GEO_SHADOW_COMPOSITION**<br>Shadow composition primitive |
| *offsetx* | x offset of shadow (or shadow composition) primitive for body primitive, pixel unit |
| *offsety* | y offset of shadow (or shadow composition) primitive for body primitive, pixel unit |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets offset of shadow from body. Offset must be specified with pixel unit.

Shadow composition primitive is the 2nd shadow of lines. When you wish shadow composition primitive is not drawn, specify same offsets for shadow and shadow composition. Shadow composition primitive is not drawn in triangles. Figure 6.12.11 shows relation between shadow primitive and shadow composition primitive.

When offset is positive number, position of x is right side of body, y is lower side of body.

When offset is negative number, position of x is left side of body, y is upper side of

body.

Offset position of shadow form body must be set before drawing shadow primitive.

Shadow primitive drawing function is available in object coordinate system drawing.

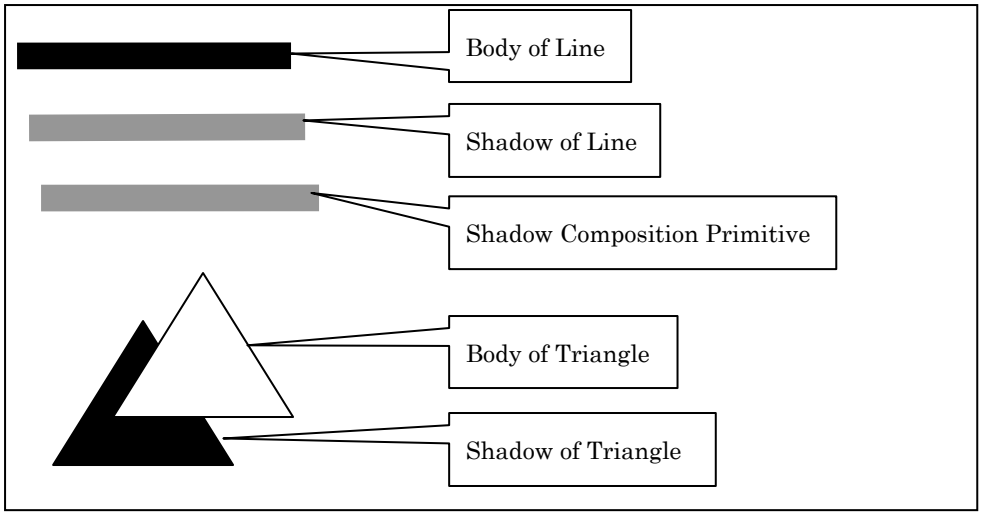



**Figure 6.12.11 Relation between Shadow Primitive and Shadow Composition Primitive**

### 6.12.12 XGdcGeoOverlapZ [Sets Z Value of Primitives (Body/Shadow/Border/Correction in Top-left Rule Non-applied Mode)]

**Interface**

GDC_BOOL **XGdcGeoOverlapZ** (GDC_CTX **drvctx*, GDC_ULONG *origin_offset*, GDC_ULONG *non_topleft_offset*, GDC_ULONG *border_offset*, GDC_ULONG *shadow_offset*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *origin_offset* | Z value of body primitive |
| *non_topleft_offset* | Z value of correction primitive in top-left rule non-applied mode |
| *border_offset* | Z value of border primitive |
| *shadow_offset* | Z value of shadow primitive |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets z value of body, shadow, border and correction primitive in top-left rule non-applied mode.

Z value must be set before drawing these primitives.

When precision of z value is 8-bit, lower 8 bits of each parameter are effective.

When precision of z value is 16-bit, lower 16 bits of each parameter are effective.

### 6.12.13 XGdcGeoShadowColor [Sets Color of Shadow]

**Interface**

GDC_BOOL **XGdcGeoShadowColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Color of shadow |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets color of shadow. This setting is effective in drawing lines with shadow, triangles with shadow, and polygons with shadow.

Once this command is executed, the same color is continuously applied until this command is executed to change the color again.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.12.14 XGdcGeoShadowBackColor [Sets Background Color of Shadow]

**Interface**

GDC_BOOL **XGdcGeoShadowBackColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Background color of shadow |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets background color of shadow. Background color of shadow is effective when drawing lines with shadow and its shadow is broken line.

Background color is corresponding to "0" in bits of broken line pattern when drawing shadow as broken line (refer to Figure 6.12.14).

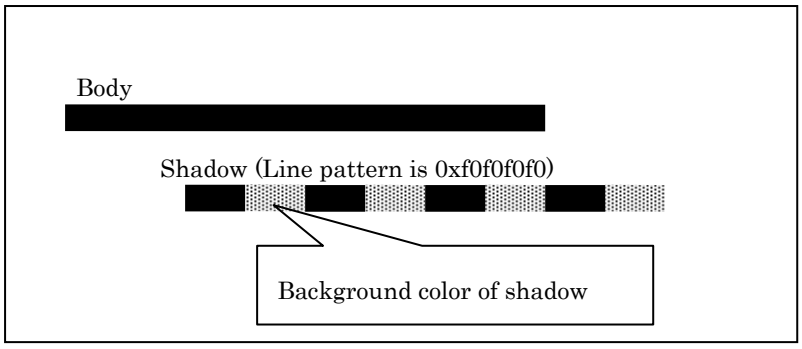



**Figure 6.12.14 Background Color of Broken Shadow Line**

Once this command is executed, the same color is continuously applied until this command is executed to change the color again.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.12.15 XGdcGeoBorderColor [Sets Color of Border]

#### Interface

GDC_BOOL **XGdcGeoBorderColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

#### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Color of border |

#### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

#### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

#### Target GDC

MB86293 or later

#### Description

Sets color of border. This setting is effective in drawing lines with border.

Once this command is executed, the same color is continuously applied until this command is executed to change the color again.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.12.16 XGdcGeoBorderBackColor [Sets Background Color of Border]

**Interface**

GDC_BOOL **XGdcGeoBorderBackColor** (GDC_CTX **drvctx*, GDC_COLOR32 *color*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Background color of border |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86293 or later

**Description**

Sets background color of border. Background color of border is effective when drawing lines with border and its border is broken line.

Background color is corresponding to 0 in bits of broken line pattern when drawing border as broken line (refer to Figure 6.12.16).

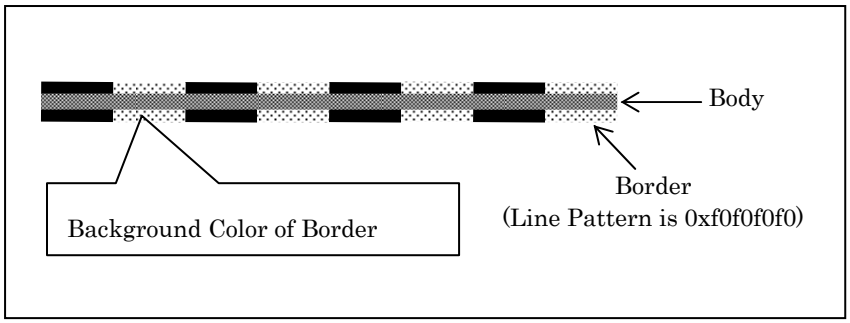



**Figure 6.12.16 Background Color of Broken Border Line**

Once this command is executed, the same color is continuously applied until this command is executed to change the color again.

The following values are used according to color mode.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.12.17 XGdcGeoSetupMode [Sets Setup Mode]

**Interface**

GDC_BOOL **XGdcGeoSetupMode** (GDC_CTX **drvctx*, GDC_ULONG *mode*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *mode* | Setup mode |
| | **GDC_GEO_TRIANGLES_SETUP_INT** |
| | Integer setup mode |
| | **GDC_GEO_TRIANGLES_SETUP_FLOAT** |
| | Floating-point setup mode |

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_INVALID_PARAMETER** Invalid parameter has been specified

**Target GDC**

MB86296S

**Description**

Set the setup mode (the method of inclination calculation) on drawing the triangle in the object coordinate system. When **GDC_GEO_TRIANGLES_SETUP_FLOAT** is specified for mode, the drawing quality in the edge part improves because the inclination of triangle drawing after that is calculated by the floating-point. For instance, it is effective for the figure in animation where texture mapping is used. If **GDC_GEO_TRIANGLES_SETUP_FLOAT** is specified for mode, it takes a lot of hardware processing time, compare to the case when **GDC_GEO_TRIANGLES_SETUP_INT** is specified for mode. Call this command and adjust in the application program side to decide whether drawing quality or the processing time (performance) to be selected.

# 6.13 Texture Image Management Commands

## 6.13.1 XGdcTextureMemoryMode [Sets Texture Memory Mode]

### Interface

GDC_BOOL **XGdcTextureMemoryMode** (GDC_CTX **drvctx*, GDC_UCHAR *mode*)

### Arguments

*drvctx* Pointer to context

*mode* Texture memory mode

**GDC_TEX_MEM_MODE_EXT**

Read from the graphics memory

**GDC_TEX_MEM_MODE_INT**

Read from the internal texture memory

### Return value

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

### Target GDC

MB86290A/86291/86292

### Description

Sets whether the internal texture memory or the graphics memory, the source memory referring the image data from.

This command doesn't need to be called so that the graphics memory is always used for the texture memory for MB86293 or later.

### 6.13.2 XGdcTextureLoadInt[8/16] [Loads Image Data to Internal Texture Memory]

**Interface**

GDC_BOOL **XGdcTextureLoadInt8** (GDC_CTX **drvctx*,
const GDC_COL8 **lpTexture*,
GDC_ULONG *oadrs*)

GDC_BOOL **XGdcTextureLoadInt16** (GDC_CTX **drvctx*,
const GDC_COL16 * *lpTexture*,
GDC_ULONG *oadrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *lpTexture* | Pointer to image data |
| *oadrs* | Offset address of the memory where texture pattern is stored, specify offset from top of the internal texture memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_DATA_TOO_BIG**
Data is too big

**GDC_ERR_ILLEGAL_DIMENSION**
Illegal vertical/horizontal size

**Target GDC**

MB86290A/86291/86292

**Description**

Loads a texture/tile image data to the internal texture memory.
The **XGdcTextureLoadInt8** and the **XGdcTextureLoadInt16** command handle 8bpp and 16bpp images respectively.
Specify *oadrs* for offset from top of the internal texture memory.
The size of the internal texture memory is 8KB (address range from "0x0000" to "0x1fff"). When the texture/tile image is loaded to out of the range, error occurs.

If the size of the image to be loaded is never set to an appropriate value by the **XGdcTextureDimension** command, sets **GDC_ERR_ILLEGAL_DIMENSION** to the error code and sets **GDC_FALSE** to the return value, then returns.

**Notes**

Prior to this command execution, sets the dimension of an image to be loaded by the **XGdcTextureDimension** command.

### 6.13.3 XGdcTextureLoadExt[8/16] [Loads Image Data to Graphics Memory]

**Interface**

GDC_BOOL **XGdcTextureLoadExt8** (GDC_CTX **drvctx*,
const GDC_COL8 **lpTexture*, GDC_ULONG *adrs*)

GDC_BOOL **XGdcTextureLoadExt16** (GDC_CTX **drvctx*,
const GDC_COL16 **lpTexture*, GDC_ULONG *adrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *lpTexture* | Pointer to refer image data |
| *adrs* | Offset address of the memory where image data is stored, offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**

Illegal vertical/horizontal size

**Target GDC**

**XGdcTextureLoadExt8** : MB86293 or later

**XGdcTextureLoadExt16** : All

**Description**

Copies texture/tile image data to the graphics memory.

The **XGdcTextureLoadExt8** and the **XGdcTextureLoadExt16** command handle 8bpp and 16bpp images respectively.

Specify *adrs* for offset from top of the graphics memory.

If the size of the image to be loaded is never set to an appropriate value by the **XGdcTextureDimension** command, sets **GDC_ERR_ILLEGAL_DIMENSION** to the error code and sets **GDC_FALSE** to the return value, then returns.

**Notes**

Prior to this command execution, sets the dimension of an image to be loaded by the **XGdcTextureDimension** command.

### 6.13.4 XGdcTextureLoadExt16Fast [Loads Image Data to Graphics Memory for Bi-linear Fast Mode]

**Interface**

GDC_BOOL **XGdcTextureLoadExt16Fast** (GDC_CTX **drvctx*,
const GDC_COL16 **lpTexture*, GDC_ULONG *adrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *lpTexture* | Pointer to image data |
| *adrs* | Address of the memory where image data is loaded, specify offset from top of the graphics memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_DL_BUF_TOO_SMALL**
Size of DL buffer is smaller than the minimum size

**GDC_ERR_ILLEGAL_DIMENSION**
Illegal vertical/horizontal size

**Target GDC**

MB86293 or later (MB86295S is excluded)

**Description**

Converts a 16bpp texture pattern to the format for bi-linear fast mode, and loads it to the graphics memory. Specify *adrs* for offset from top of the graphics memory.
If the size of the image to be loaded is never set to an appropriate value by the **XGdcTextureDimension** command, sets **GDC_ERR_ILLEGAL_DIMENSION** to the error code and sets **GDC_FALSE** to the return value, then returns.

**Notes**

- Prior to this command execution, sets the dimension of an image to be loaded by the **XGdcTextureDimension** command. The size of the loaded image is a size of an original image, it is not four times of original image.
- The size of the image converted for bi-linear fast mode becomes four times as

large as the original one.

### 6.13.5 XGdcTextureDimension [Sets Texture/Tile Information]

**Interface**

GDC_BOOL **XGdcTextureDimension** (GDC_CTX **drvctx*,
GDC_ULONG *adrs*, GDC_ULONG *oadrs*,
GDC_ULONG *w*, GDC_ULONG *h*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *adrs* | Address of texture/tile image data, specify offset from top of the graphics memory |
| *oadrs* | Offset, in the case with an external tile, it specifies by the offset from top address of the tile image data. Other cases are specified in the offset address from top of the internal texture memory. |
| *w* | Pattern data width, in pixels, power of "2" only |
| *h* | Pattern data height, in pixels, power of "2" only |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**
Illegal vertical/horizontal size

**Target GDC**

All

**Description**

Sets following texture/tile information.

- Base address of referred texture/tile image data
- Offset
- Image data width
- Image data height

*adrs* and *oadrs* must be specified with suitable value for the loaded position of the

referred image data according to Table 6.13.5a.

**Table 6.13.5a Values to be Specified to *adrs* and *oadrs***

| Stored Position of Referred Texture/Tile Image Data (*1) | *adrs* | *oadrs* |
|---|---|---|
| Internal Texture | 0 | Offset from top address of the internal texture memory |
| External Texture | Base address of texture image data, which is offset from top of the graphics memory | 0 |
| Internal Tile | 0 | Offset from top address of the internal texture memory |
| External Tile (*2) | Base address of tile image data, which is offset from top of the graphics memory | Image data store position, offset from base address of tile image data |

(*1) Available destination address of texture/tile image data is changed according to the Graphics Controller.

(*2) External tile function is only for MB86293 or later with which Internal Texture Memory is not equipped.

Range of image data width and image data height according to the Graphics Controller are shown in Table 6.13.5b.

**Table 6.13.5b Range of Image Data Width and Height**

| | MB86290A/291/292 | MB86293 or Later |
|---|---|---|
| Internal Texture Memory | 16,32,64 | Nothing |
| Graphics Memory | 16,32,64,128,256 | 16 to 4096, power of "2" only |

### 6.13.6 XGdcBltTexture [Loads BitBlt Texture from Graphics Memory to Internal Texture Memory]

**Interface**

GDC_BOOL **XGdcBltTexture** (GDC_CTX **drvctx*,
GDC_ULONG *sadrs*, GDC_ULONG *sstride*,
GDC_USHORT *x*, GDC_USHORT *y*,
GDC_USHORT *w*, GDC_USHORT *h*, GDC_ULONG *oadrs*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *sadrs* | Base address of the transferring source frame, specify offset address from the top of the graphics memory |
| *sstride* | Stride of the transferring source frame, in pixels of width of frame (*1)<br>[Range of the value]<br>In 16-bit color mode: in pixels, range from "16" to "4096"<br>In 8-bit color mode: in pixels, range from "8" to "4096" |
| *x* | x coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *y* | y coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *w* | Width of BitBlt area of x coordinate direction, in pixels,<br>"16", "32", "64" |
| *h* | Height of BitBlt area of x coordinate direction, in pixels,<br>"16", "32", "64" |
| *oadrs* | Address at image data store destination, specify offset address from the top of the internal texture memory |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**

Illegal vertical/horizontal size

**Target GDC**

MB86290A/86291/86292

**Description**

Load the image data loaded into the graphics memory into the internal texture memory.

Specify offset address from the top of the internal texture memory for *oadrs*.

(*1) The range check of the parameter is not performed, set a valid value.

# 6.14 Binary Pattern Drawing Commands

## 6.14.1 XGdcBitPatternDraw [Draws Binary Pattern (No Clipping)]

### Interface

GDC_BOOL **XGdcBitPatternDraw** (GDC_CTX **drvctx*,
GDC_USHORT *x*, GDC_USHORT *y*,
GDC_USHORT *w*, GDC_USHORT *h*,
const GDC_BINIMAGE **lpPattern*)

### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate on the drawing frame where the top left point of the binary pattern is drawn, range from "0" to "4095" (*1) |
| *y* | y coordinate on the drawing frame where the top left point of the binary pattern is drawn, range from "0" to "4095" (*1) |
| *w* | Width of the binary pattern data, in pixels, range from "1" to "2016" |
| *h* | Height of the binary pattern data, in pixels, "1" or more |
| *lpPattern* | Pointer to the binary pattern data |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_DATA_TOO_BIG**

Too large data

**GDC_ERR_DL_BUF_TOO_SMALL**

Size of DL buffer is smaller than the minimum size

### Target GDC

All

**Description**

Draws a binary pattern.

The binary pattern specified by *lpPattern* must be GDC_BINIMAGE format that packed in 32 pixels.

It is drawn to the corresponding pixel by the following colors by the value of each bit of the binary pattern data.

- Binary pattern is "1": The foreground color specified by the **XGdcColor** command
- Binary pattern is "0": The background color specified by the **XGdcBackColor** command

When *w* exceeds "2016", sets error code **GDC_ERR_DATA_TOO_BIG** and returns **GDC_FALSE**.

When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

(*1) The range check of the parameter is not performed, set a valid value.

**Notes**

- Do not specify parameters (for *x*, *y*, *w* and *h*) that draw to outside of drawing frame since clipping by hardware is not performed.
- No guarantee the drawing operation outside the drawing frame of the size from (0,0) to (4095,4095). Check the coordinates value in the application program side if necessary.

### 6.14.2 XGdcBitPatternDrawByte [Draws Binary Pattern (Clipping)]

**Interface**

GDC_BOOL **XGdcBitPatternDrawByte** (GDC_CTX **drvctx*,
GDC_ULONG *x0*, GDC_ULONG *y0*, int *x1*, int *y1*,
GDC_ULONG *w*, GDC_ULONG *h*,
const GDC_UCHAR **sadr*, GDC_ULONG *mwidth*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x0* | x coordinate in the binary pattern, range from "0" to "2016-*w*" (*1) |
| *y0* | y coordinate in the binary pattern, range from "0" to "4095" (*1) |
| *x1* | x coordinate on the drawing frame where the top left point of the binary pattern is drawn , range from "-2147483648" to "2147483647" |
| *y1* | y coordinate on the drawing frame where the top left point of the binary pattern is drawn , range from "-2147483648" to "2147483647" |
| *w* | Width of the binary pattern data, in pixels, range from "1" to "2016" |
| *h* | Height of the binary pattern data, in pixels, range from "1" to "4096" |
| *sadr* | Pointer to the binary pattern data |
| *mwidth* | Width of area where the binary pattern data is installed, in bytes, "1" or more |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_DATA_TOO_BIG**

Too large data

**GDC_ERR_INVALID_PARAMETER**

Invalid parameter has been specified

**GDC_ERR_DL_BUF_TOO_SMALL**

Size of DL buffer is smaller than the minimum size

**Target GDC**

All

**Description**

Draws binary pattern specified by rectangle (*x0*, *y0*) - (*x0*+*w*, *y0*+*h*) to drawing frame. This rectangle is a part of the binary pattern specified by *sadr*. The pixel format of the binary pattern must be pack of 8 pixels a byte.

It is drawn to the corresponding pixel by the following colors by the value of each bit of the binary pattern data.

- Binary pattern is "1": The foreground color specified by the **XGdcColor** command
- Binary pattern is "0": The background color specified by the **XGdcBackColor** command

When *w* exceeds "2016", sets error code **GDC_ERR_DATA_TOO_BIG** and returns **GDC_FALSE**. And also, when *x0*+*w* is more than *mwidth**8, sets error code **GDC_ERR_INVALID_PARAMETER** and returns **GDC_FALSE.**

When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

*x1* and *y1* can be specified in a signed 32-bit value, but only the pixels within the range from "0" to "4095" are drawn.

Both the width and the height of a binary pattern to be drawn actually are calculated again with *w*, *h*, and coordinates (*x1*, *y1*) of the drawing destination.

When either the width or the height becomes "0" as the result of re-calculation, nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

When *h* exceeds "4096", the value is clipped with "4096".

(*1) The range check of the parameter is not performed, set a valid value.

### 6.14.3 XGdcBitPatternMode [Sets Enlargement/Reduction Mode]

**Interface**

GDC_BOOL **XGdcBitPatternMode** (GDC_CTX **drvctx*, GDC_UCHAR *mode*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *mode* | Enlargement/reduction mode (**GDC_BPSCALE_H_*** and **GDC_BPSCALE_V_*** are applicable at the same time) |
| | **GDC_BPSCALE_H_EQUIV** Horizontal magnification = 1 |
| | **GDC_BPSCALE_H_TWICE** Horizontal magnification = 2 |
| | **GDC_BPSCALE_H_HALF** Horizontal magnification = 1/2 |
| | **GDC_BPSCALE_V_EQUIV** Vertical magnification = 1 |
| | **GDC_BPSCALE_V_TWICE** Vertical magnification = 2 |
| | **GDC_BPSCALE_V_HALF** Vertical magnification = 1/2 |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Sets enlargement/reduction mode for binary pattern drawing.

# 6.15 BLT Commands

## 6.15.1 XGdcBltCopy [Copies BitBlt Area in Current Drawing Frame]

**Interface**

GDC_BOOL **XGdcBltCopy** (GDC_CTX **drvctx*,
GDC_ULONG *x0*, GDC_ULONG *y0*,
int *x1*, int *y1*, GDC_ULONG *w*, GDC_ULONG *h*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x0* | x coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *y0* | y coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *x1* | x coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *y1* | y coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *w* | Width of rectangle area of x coordinate direction, in pixels, range from "1" to "4096" |
| *h* | Height of rectangle area of x coordinate direction, in pixels, range from "1" to "4096" |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**

Illegal vertical/horizontal size

### Target GDC

All

### Description

Copies BitBlt area inside current drawing frame by rectangle block transfer. The transferring source is a current drawing frame in the graphics memory.

When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

When either *w* or *h* exceeds "4096", sets error code **GDC_ERR_ILLEGAL_DIMENSION** and sets return value to **GDC_FALSE** and returns.

*x1* and *y1* can be specified in a 32-bit signed value, but only the pixels within the range from "0" to "4095" are drawn.

Both the width and the height of a bitmap to be drawn actually are calculated again with *w*, *h*, and coordinates (*x1*, *y1*) of the drawing destination. When either the width or the height becomes "0" as the result of re-calculation, nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

In MB86290A, the following values cannot be specified because of specifications limitation.

- 2 <= *w* <= 4, in 16-bit color mode
- 2 <= *w* <= 8, in 8-bit color mode

### 6.15.2 XGdcBltCopyAlt[Sync] [Copies BitBlt Area between Arbitrary Drawing Frames]

**Interface**

GDC_BOOL **XGdcBltCopyAlt** (GDC_CTX **drvctx*,
GDC_ULONG *x0*, GDC_ULONG *y0*,
int *x1*, int *y1*,
GDC_ULONG *w*, GDC_ULONG *h*,
GDC_ULONG *sadr*, GDC_ULONG *sstride*,
GDC_ULONG *dadr*, GDC_ULONG *dstride*)

GDC_BOOL **XGdcBltCopyAltSync** (GDC_CTX **drvctx*,
GDC_ULONG *x0*, GDC_ULONG *y0*,
int *x1*, int *y1*,
GDC_ULONG *w*, GDC_ULONG *h*,
GDC_ULONG *sadr*, GDC_ULONG *sstride*,
GDC_ULONG *dadr*, GDC_ULONG *dstride*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x0* | x coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *y0* | y coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" |
| *x1* | x coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *y1* | y coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *w* | Width of BitBlt area of x coordinate direction, in pixels, range from "1" to "4096" |
| *h* | Height of BitBlt area of y coordinate direction, in pixels, range from "1" to "4096" |

*sadr* Base address of the transferring source frame, specify offset address from top of the graphics memory

*sstride* Stride of the transferring source frame, in pixels of width of frame, range from "1" to "4096" (*1)

*dadr* Base address of the drawing destination frame, specify offset address from top of the graphics memory

*dstride* Stride of the drawing destination frame, in pixels of width of frame, range from "1" to "4096" (*1)

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**

Illegal vertical/horizontal size

**Target GDC**

All

**Description**

Copies rectangle between different drawing frames by rectangle block transfer.
An any drawing frame in the graphics memory is a target at the transferring source and drawing destination. Color mode of transferring source and destination must be the same. The **XGdcBltCopyAltSync** command is synchronously executed to the vertical blanking interval, and executes the block transfer afterwards.
Source and destination field must not be overlapped to each other.
When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.
When either *w* or *h* exceeds "4096", sets error code **GDC_ERR_ILLEGAL_DIMENSION** and sets return value to **GDC_FALSE** and returns.
*x1* and *y1* can be specified in a 32-bit signed value, but only the pixels within the range from "0" to "4095" are drawn. Clipping operation by the

**XGdcDrawClipFrame** command is not applicable.

Both the width and the height of a bitmap to be drawn actually are calculated again with *w*, *h*, and coordinates (*x1*, *y1*) of the drawing destination.

When either the width or the height becomes "0" as the result of re-calculation, nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

In MB86290A, the following values cannot be specified because of specifications limitation.

- 2 <= *w* <= 4, in 16-bit color mode
- 2 <= *w* <= 8, in 8-bit color mode

(*1) The range check of the parameter is not performed, set a valid value.

### 6.15.3 XGdcBltDraw[8/16] [Copies BitBlt Area from Main Memory to Current Drawing Frame]

**Interface**

GDC_BOOL **XGdcBltDraw8** (GDC_CTX **drvctx*,
int *x*, int *y*,
GDC_ULONG *w*, GDC_ULONG *h*,
const GDC_COL8 **lpRect*)

GDC_BOOL **XGdcBltDraw16** (GDC_CTX **drvctx*,
int *x*, int *y*,
GDC_ULONG *w*, GDC_ULONG *h*,
const GDC_COL16 **lpRect*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *y* | y coordinate from frame origin left corner BitBlt area in drawing destination, range from "-2147483648" to "2147483647" |
| *w* | Width of BitBlt area of x coordinate direction, in pixels, range from "1" to "4096" |
| *h* | Height of BitBlt area of x coordinate direction, in pixels, range from "1" to "4096" |
| *lpRect* | Pointer to rectangle pattern data |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_DATA_TOO_BIG**

Too large data

**GDC_ERR_DL_BUF_TOO_SMALL**

Size of DL buffer is smaller than the minimum size

**Target GDC**

**XGdcBltDraw8** : All

**XGdcBltDraw16** : All

**Description**

Copies bitmap from main memory to the current drawing frame by rectangle block transfer.

Color mode of the bitmap is assumed to be the same as that of current drawing frame. The color mode of *lpRect* and the drawing frame should be the same.

When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

When *w* exceeds "4096", sets error code **GDC_ERR_DATA_TOO_BIG,** and sets return value to **GDC_FALSE** and returns.

*x* and *y* can be specified in a 32-bit signed value, but only the pixels within the range from "0" to "4095" are drawn.

Both the width and the height of a bitmap to be drawn actually are calculated again with *w*, *h*, and coordinates (*x*, *y*) of the drawing destination.

When either the width or the height becomes "0" as the result of re-calculation, nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

When *h* exceeds "4096", the value is clipped with "4096".

### 6.15.4 XGdcBltFill [Fills BitBlt Area]

**Interface**

GDC_BOOL **XGdcBltFill** (GDC_CTX **drvctx*,
int *x*, int *y*,
GDC_ULONG *w*, GDC_ULONG *h*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x* | x coordinate of the top left corner of BitBlt area to be filled, range from "-2147483648" to "2147483647" |
| *y* | y coordinate of the top left corner of BitBlt area to be filled, range from "-2147483648" to "2147483647" |
| *w* | Width of the BitBlt to be filled, in pixels, range from "1" to "4096" |
| *h* | Height of the BitBlt to be filled, in pixels, range from "1" to "4096" |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Fills a rectangle with the foreground color specified by the **XGdcColor** command.
When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.
When *w* or *h* exceeds "4096", the value is clipped with "4096".
*x* and *y* can be specified in a 32-bit signed value, but only the pixels within the range from "0" to "4095" are drawn.
Both the width and the height of a BitBlt to be drawn actually are calculated again with *w*, *h*, and coordinates (*x*, *y*) of the drawing destination.

When either the width or the height becomes "0" as the result of re-calculation, nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

### 6.15.5 XGdcBltColorTransparent [Sets Transparent Color at Copying BitBlt Area]

**Interface**

GDC_BOOL **XGdcBltColorTransparent** (GDC_CTX * *drvctx*,
GDC_COLOR32 *color*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *color* | Color code to be treated as transparent |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

MB86291 or later

**Description**

Sets the transparent color for Blt function command.

The value of *color* is used according to color mode as follows.

- 8-bit color mode: lower 8 bits of *color*
- 16-bit color mode: lower 16 bits of *color*

### 6.15.6 XGdcBltCopyAltAlpha [Copies BitBlt Area with Alpha Blending]

**Interface**

GDC_BOOL **XGdcBltCopyAltAlpha** (GDC_CTX **drvctx*,
GDC_USHORT *x0*, GDC_USHORT *y0*,
GDC_USHORT *x1*, GDC_USHORT *y1*,
GDC_USHORT *bx*, GDC_USHORT *by*,
GDC_USHORT *w*, GDC_USHORT *h*,
GDC_ULONG *sadr*, GDC_ULONG *sstride*,
GDC_ULONG *bstride*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |
| *x0* | x coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" (*1) |
| *y0* | y coordinate from frame origin left corner BitBlt area in transferring source, range from "0" to "4095" (*1) |
| *x1* | x coordinate from frame origin left corner BitBlt area in drawing destination, range from "0" to "4095" (*1) |
| *y1* | y coordinate from frame origin left corner BitBlt area in drawing destination, range from "0" to "4095" (*1) |
| *bx* | x coordinate from frame origin left corner BitBlt area in the alpha map, range from "0" to "4095" (*1) |
| *by* | y coordinate from frame origin left corner BitBlt area in the alpha map, range from "0" to "4095" (*1) |
| *w* | Width of rectangle area of x coordinate direction, in pixels, range from "1" to "4096" |
| *h* | Height of rectangle area of x coordinate direction, in pixels, range from "1" to "4096" |

*sadr* Base address of source frame, specify offset from top of the graphics memory

*sstride* Stride of transferring source frame, in pixels of width of frame, range from "1" to "4096" (*1)

*bstride* Stride of alpha map area, in pixels of width of frame, range from "1" to "4096" (*1)

**Return value**

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_ILLEGAL_DIMENSION**

Illegal vertical/horizontal size

**Target GDC**

MB86293 or later

**Description**

Execute alpha blending with source BitBlt area and destination area. Copies alpha blended area to the *x1*, *y1* coordinates of current drawing frame. Source BitBlt area is specified with *sadrs*, *sstride*, *x0*, *y0*, *w* and *h*. And destination Blt area is specified with *x1* and *y1*. Alpha blending coefficient area is specified with *bstride*, *bx*, *by*, *w* and *h*. And its top address is specified by the **XGdcSetAlphaMapBase** command.

When either *w* or *h* is "0", nothing is processed and **GDC_TRUE** is set to the return value, then it returns.

When either *w* or *h* exceeds "4096", sets error code **GDC_ERR_ILLEGAL_DIMENSION** and returns **GDC_FALSE**.

(*1) The range check of the parameter is not performed, set a valid value.

**Notes**

- Do not draw to the area where the range of the drawing frame is exceeded though the clipping with hardware is executed.
- When drawing outside the drawing frame of the size from (0,0) to (4095,4095), we do not guarantee the operation.

# 6.16 Execution Control Commands

## 6.16.1 XGdcVerticalSync [Generates Sync Command]

### Interface

GDC_BOOL **XGdcVerticalSync** (GDC_CTX **drvctx*)

### Arguments

| | |
|---|---|
| *drvctx* | Pointer to context |

### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

### Target GDC

All

### Description

Generates a display list command for waiting VSYNC (Sync command) and adds it to the end of current display list.

Display list after Sync command is executed synchronously with VSYNC.

### 6.16.2 XGdcInterrupt [Generates Interrupt Command]

**Interface**

GDC_BOOL **XGdcInterrupt** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Target GDC**

All

**Description**

Generates a display list command for requesting interrupt (Interrupt command) and adds it to the end of current display list.

# 6.17 Drawing Attributes Restore Command

## 6.17.1 XGdcRestoreAttr [Restores Drawing Attributes]

### Interface

GDC_BOOL **XGdcRestoreAttr** (GDC_CTX **drvctx*, GDC_ULONG *attr*)

### Arguments

*drvctx* Pointer to context

*attr* Specify attributes to be restored in the following, use OR operation to specify two or more

**GDC_RESTORE_COMMON**
Common drawing attributes

**GDC_RESTORE_GEOMETRY**
Geometry drawing attributes

**GDC_RESTORE_EXTEND**
Extended drawing attributes (for MB86293 or later)

**GDC_RESTORE_ALL**
All of them

### Return value

**GDC_TRUE** Complete

**GDC_FALSE** Incomplete

### Error code

**GDC_ERR_SWITCH_DL_BUF_FAILED**
The **XGdcSwitchDLBuf** command has finished abnormally

**GDC_ERR_INVALID_ATTRIBUTE**
Invalid attribute has been specified

### Target GDC

All

### Description

Generates display list for setting the Graphics Controller to drawing attributes in the context. In the system that Driver Command are used in two or more tasks, drawing attributes of the Graphics Controller set by a task may be changed by other tasks. Therefore, after a executing of the **XGdcFlush/XGdcFlushEx** command, it is necessary to re-set up each drawing attribute. By using this command, multiple attributes can be restored at once. In each macro that can be

specified for *attr*, the kind of the drawing attribute set again (hereafter referred as the "restoration group") is as follows. And also, the correspondence of the restoration group and the drawing attribute is shown in Table 6.17.1 (on next page).

- **GDC_RESTORE_COMMON**

  Means common attribute in all Graphics Controllers, concerns drawing frame, clipping, device coordinate system drawing, BitBlt, etc.

- **GDC_RESTORE_GEOMETRY**

  Means object coordinate system drawing, available in MB86291 or later.

- **GDC_RESTORE_EXTEND**

  Means attributes for MB86293 or later, shadow, border, device coordinate log, etc.

- **GDC_RESTORE_ALL**

  Restores all drawing attributes mentioned above.

  However, this command cannot restore the following attributes, use suitable command for these attributes if you need restore them.

  – Vertex color for Gouraud shading (use the **XGdcColor** or the **XGdcVertexColor[32/3f]** command)
  – Texture/tile pattern loaded in the graphics memory or the internal texture memory (use the **XGdcTextureLoadInt[8/16]**, the **XGdcTextureLoadExt[8/16]**, the **XGdcTextureLoadExt16Fast** or the **XGdcBltTexture** command)
  – 4x4 matrix (use the **XGdcGeoLoadMatrix[f]** command)

**Table 6.17.1 Drawing Attributes to be Restored**

| No. | Restoration Group (Macro name specified with the **XGdcRestoreAttr** command) | Drawing Attributes | Corresponding Drawing Attribute Setting Commands |
|---|---|---|---|
| 1 | **GDC_RESTORE_COMMON** | Color mode | **XGdcDrawDimension** |
| 2 | | Clipping mode | **XGdcClipMode** |
| 3 | | Enlargement/reduction mode of binary pattern | **XGdcBitPatternMode** |
| 4 | | Z precision | **XGdcSetZPrecision** |
| 5 | | For lines on device coordinate system | **XGdcSetAttrLine** |
| 6 | | For surfaces on device coordinate system | **XGdcSetAttrSurf** |
| 7 | | For texture mapping | **XGdcSetAttrTexture** |
| 8 | | For BitBLT | **XGdcSetAttrBlt** |
| 9 | | Base address of drawing frame | **XGdcDrawDimension** |
| 10 | | Width of drawing frame | **XGdcDrawDimension** |
| 11 | | Base address of Z-buffer | **XGdcBufferCreateZ** |
| 12 | | Base address of texture memory | **XGdcTextureDimension** |
| 13 | | Base address of polygon flag buffer | **XGdcBufferCreateC** |
| 14 | | x value at upper left of clipping frame | **XGdcDrawClipFrame** |
| 15 | | y value at upper left of clipping frame | **XGdcDrawClipFrame** |
| 16 | | x value at lower right of clipping frame | **XGdcDrawClipFrame** |
| 17 | | y value at lower right of clipping frame | **XGdcDrawClipFrame** |
| 18 | | Dimension of texture pattern | **XGdcTextureDimension** |
| 19 | | Dimension of tile pattern | **XGdcTextureDimension** |
| 20 | | Address of texture/tile pattern | **XGdcTextureDimension** |
| 21 | | Drawing color | **XGdcColor** |
| 22 | | Background color | **XGdcBackColor** |
| 23 | | Alpha blending coefficient | **XGdcSetAlpha** |
| 24 | | Broken line pattern | **XGdcSetLinePattern** |
| 25 | | Texture border color | **XGdcSetTextureBorder** |
| 26 | **GDC_RESTORE_GEOMETRY** | Transparent color of BitBLT | **XGdcBltColorTransparent** |
| 27 | | NdcDc conversion coefficient for x and y, GDC_FIXED32 | **XGdcGeoNdcDcViewportCoef** |
| 28 | | NdcDc conversion coefficient for x and y, GDC_SFLOAT | **XGdcGeoNdcDcViewportCoeff** |
| 29 | | NdcDc conversion coefficient for z, GDC_FIXED32 type | **XGdcGeoNdcDcDepthCoef** |
| 30 | | NdcDc conversion coefficient for z, GDC_SFLOAT type | **XGdcGeoNdcDcDepthCoeff** |
| 31 | | x,y coordinate for view volume clipping, GDC_FIXED32 type | **XGdcGeoViewVolumeXYClip** |
| 32 | | x,y coordinate for view volume clipping, GDC_SFLOAT type | **XGdcGeoViewVolumeXYClipf** |
| 33 | | z coordinate for view volume clipping, GDC_FIXED32 type | **XGdcGeoViewVolumeZClip** |
| 34 | | z coordinate for view volume clipping, GDC_SFLOAT type | **XGdcGeoViewVolumeZClipf** |
| 35 | | w coordinate for view volume clipping, GDC_FIXED32 type | **XGdcGeoViewVolumeWminClip** |
| 36 | | w coordinate for view volume clipping, GDC_SFLOAT type | **XGdcGeoViewVolumeWminClip** |
| 37 | | For lines on object coordinate system (antialias) | **XGdcSetAttrLine** |
| 38 | | For lines on object coordinate system (end of the line | **XGdcSetAttrLine** |
| 39 | | For lines on object coordinate system (offset control of broken line pattern) | **XGdcSetAttrLine** |
| 40 | | For lines on object coordinate system (correction mode of thick line connection) | **XGdcGeoSetAttrLine** |
| 41 | | For lines on object coordinate system (correction mode of broken line pattern) | **XGdcGeoSetAttrLine** |
| 42 | | For lines on object coordinate system (uniform mode of line width) | **XGdcGeoSetAttrLine** |
| 43 | | For lines on object coordinate system (thick/broken line vertical mode) | **XGdcGeoSetAttrLine** |
| 44 | | For lines on object coordinate system (drawing mode of border primitive) | **XGdcGeoSetAttrLine** |
| 45 | | For lines on object coordinate system (drawing mode of shadow primitive) | **XGdcGeoSetAttrLine** |
| 46 | | For surfaces on object coordinate system (culling back face of triangle) | **XGdcGeoSetAttrSurf** |
| 47 | | For surfaces on object coordinate system (direction of surface of triangle) | **XGdcGeoSetAttrSurf** |
| 48 | | For surfaces on object coordinate system (drawing | **XGdcGeoSetAttrSurf** |
| 49 | | For surfaces on object coordinate system (drawing mode of shadow primitive) | **XGdcGeoSetAttrSurf** |

(Continue)

**Table 6.17.1 Drawing Attributes to be Restored** (Continued)

| No. | Restoration Group (Macro name specified with the **XGdcRestoreAttr** command) | Drawing Attributes | Corresponding Drawing Attribute Setting Commands |
|---|---|---|---|
| 50 | **GDC_RESTORE_EXTEND** | Base address of alpha map | **XGdcSetAlphaMapBase** |
| 51 | | For lines on object coordinate system (antialias) | **XGdcSetAttrLine** |
| 52 | | For lines on object coordinate system (end of the line | **XGdcSetAttrLine** |
| 53 | | For lines on object coordinate system (offset control of broken line pattern) | **XGdcSetAttrLine** |
| 54 | | For lines on object coordinate system (correction mode of thick line connection) | **XGdcGeoSetAttrLine** |
| 55 | | For lines on object coordinate system (correction mode of broken line pattern) | **XGdcGeoSetAttrLine** |
| 56 | | For lines on object coordinate system (uniform mode of line width) | **XGdcGeoSetAttrLine** |
| 57 | | For lines on object coordinate system (thick/broken line vertical mode) | **XGdcGeoSetAttrLine** |
| 58 | | For lines on object coordinate system (drawing mode of border primitive) | **XGdcGeoSetAttrLine** |
| 59 | | For lines on object coordinate system (drawing mode of shadow primitive) | **XGdcGeoSetAttrLine** |
| 60 | | For surfaces on object coordinate system (culling back face of triangle) | **XGdcGeoSetAttrSurf** |
| 61 | | For surfaces on object coordinate system (direction of surface of triangle) | **XGdcGeoSetAttrSurf** |
| 62 | | For surfaces on object coordinate system (drawing | **XGdcGeoSetAttrSurf** |
| 63 | | For surfaces on object coordinate system (drawing mode of shadow primitive) | **XGdcGeoSetAttrSurf** |
| 64 | | Number of pixels for broken lines fixation | **XGdcGeoSetAttrLine** |
| 65 | | For shadow of lines | **XGdcSetAttrLine** |
| 66 | | For shadow of surfaces | **XGdcSetAttrSurf** |
| 67 | | Shadow color | **XGdcGeoShadowColor** |
| 68 | | Background color of shadow | **XGdcGeoShadowBackColor** |
| 69 | | For border of lines | **XGdcSetAttrLine** |
| 70 | | Border color of lines | **XGdcGeoBorderColor** |
| 71 | | Background border color of lines | **XGdcGeoBorderBackColor** |
| 72 | | For surfaces of non top-left primitives | **XGdcSetAttrSurf** |
| 73 | | x,y offset of shadow | **XGdcGeoShadowXY** |
| 74 | | x,y offset of shadow composition | **XGdcGeoShadowXY** |
| 75 | | Z value of body, shadow, border and correction primitive in non top-left mode | **XGdcGeoOverlapZ** |
| 76 | | Base address of area to be outputted device coordinate logs | **XGdcGeoSetLogOutBase** |

# 6.18 Video Capture Control Commands

Please use video capture control command (**GdcCap***) in this section after initializing a digital video decoder with an I²C control command (**GdcI2C***), when using the digital video decoder connected with an I²C interface. For details information, refer to the "Application Note".

## 6.18.1 GdcCapSetVideoCaptureMode [Sets Mode of Video Capture]

### Interface

GDC_BOOL **GdcCapSetVideoCaptureMode** (GDC_ULONG *mode*)

### Arguments

*mode* Sets modes of video capture. The value is set to VCM (Video Capture Mode) register as it is. For details information about VCM register, refer to the Graphics Controller hardware specifications of using.
Macros representing each mode are prepared. These can be used as the need arises. Sets each mode by combining (use OR operation) macros in Table 6.18.1.

**Table 6.18.1 Video Capture Mode Control Macro List**

| Macros | Meaning |
|---|---|
| **GDC_CAP_START** | Starts capturing |
| **GDC_CAP_STOP** | Stops capturing |
| **GDC_CAP_ENABLE_V_INTERPOLATION** | Performs the interpolation of perpendicular direction |
| **GDC_CAP_DISABLE_V_INTERPOLATION** | Not perform the interpolation of perpendicular direction |
| **GDC_CAP_NTSC** | Video=NTSC |
| **GDC_CAP_PAL** | Video=PAL |

### Return value

**GDC_TRUE** Complete

### Error code

None

### Target GDC

MB86291 or later (MB86293 is excluded)

### Description

Sets video capture mode by setting VCM (Video Capture Mode) register to *mode*.

### Notes

This command is scheduled to be abolished in the Graphics Driver of the next version. Therefore, use the following new Driver Command in application programs newly made. Refer to the description of each command for details.

- GdcCapSetAttrVideo command
- GdcCapStartVideoCapture command
- GdcCapStopVideoCapture command

### 6.18.2 GdcCapGetErrorStatus [Gets Error Status of Video Capture]

#### Interface

GDC_ULONG **GdcCapGetErrorStatus** (void)

#### Arguments

None

#### Return value

Video Capture Status (VCS) register in the following format:

| bit 31 | . . . | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Don't care | . . . | x | x | x | x | x |

'x': used bit

bit 0-4: Error status Error occurred = except "00000", Error none = all "0"

All other bits: Don't care

**Figure 6.18.2 Error Status of Video Capture**

#### Target GDC

MB86291 or later (MB86293 is excluded)

#### Description

Reads VCS (Video Capture Status) register and returns error status.

### 6.18.3 GdcCapClearErrorStatus [Clears Error Status of Video Capture]

**Interface**

GDC_BOOL **GdcCapClearErrorStatus** (void)

**Arguments**

None

**Return value**

**GDC_TRUE** Complete

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Clears error status by writing VCS (Video Capture Status) register to "0".

### 6.18.4 GdcCapSetVideoCaptureBuffer [Sets Video Capture Buffer]

**Interface**

GDC_BOOL **GdcCapSetVideoCaptureBuffer** (GDC_ULONG *saddr*,
GDC_ULONG *eaddr*, GDC_ULONG *stride*)

**Arguments**

| | |
|---|---|
| *saddr* | Top address of the video capture buffer, offset from top of the graphics memory |
| *eaddr* | Bottom address +1 of the video capture buffer, offset from top of the graphics memory |
| *stride* | Stride of video capture buffer, in blocks of 64 bytes |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets video capture buffer.
The top address needs to be in a 16-byte boundary.
Please specify the bottom address +1 of the video capture buffer as the end address.
The video capture buffer size needs a size that is a part for the image to take at least.

**Notes**

When using the video-captured image for texture mapping, set the width of the memory of the video capture buffer to the value of the power of "2" and over "64".

### 6.18.5 GdcCapSetImageArea [Sets Range of Image]

**Interface**

GDC_BOOL **GdcCapSetImageArea** (GDC_USHORT *x0*, GDC_USHORT *y0*, GDC_USHORT *x1*, GDC_USHORT *y1*)

**Arguments**

| | |
|---|---|
| *x0* | The upper left x coordinate of the image, range from "0" to "4095" |
| *y0* | The upper left y coordinate of the image, range from "0" to "4095" |
| *x1* | The lower right x coordinate of the image, range from "0" to "4095" |
| *y1* | The lower right y coordinate of the image, range from "0" to "4095" |

**Return value**

**GDC_TRUE** Complete

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets the dimension of the image to be stored in the video capture buffer.
The part of inputted image which from (*x0*, *y0*) and (*x1*, *y1*) is stored in the video capture buffer. The starting point (0,0) is upper left corner of the inputted image.
Please set coordinates $x0<x1$ and $y0<y1$ to specify the range of the image.
The number of pixels of the image width from *x0* and *x1* must be even.

**Notes**

This command is scheduled to be abolished in the next version of the graphics driver. Therefore, use the **GdcCapSetCaptureArea** command in application programs to be made newly. Refer to the description of the **GdcCapSetCaptureArea** command for details.

### 6.18.6 GdcCapGetImageAddress [Gets Address of Captured Image]

#### Interface

GDC_ULONG *__GdcCapGetImageAddress__ (void)

#### Arguments

None

#### Return value

Top address of captured image, offset address from top of the graphics memory

#### Target GDC

MB86294 or later

#### Description

Returns the top address of the captured image, which is the value of register L1DA. Before the calling of this command, stop the video capturing by the __GdcCapStopVideoCapture__ command as shown below, because the top address of captured image is progressed while the video capture is running.

```
GDC_ULONG *adrs;
GdcCapStopVideoCapture(); /* Stops the video capturing */
adrs = GdcCapGetImageAddress(); /* Gets address of captured image */
```

### 6.18.7 GdcCapSetWindowMode [Sets Layer L1 (W) Mode]

**Interface**

GDC_BOOL **GdcCapSetWindowMode** (GDC_ULONG *format*,
GDC_ULONG *mode*)

**Arguments**

*format* Sets color format of layer L1 (W).

| | |
|---|---|
| **GDC_CAP_RGB_MODE** | RGB mode (default) |
| **GDC_CAP_YC_MODE** | YC mode |

*mode* Sets whether layer L1 (W) is used as a normal display layer or a video capture.

| | |
|---|---|
| **GDC_CAP_NORMAL_MODE** | Normal mode (default) |
| **GDC_CAP_CAPTURE_MODE** | Capture mode |

**Return value**

**GDC_TRUE** Complete

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets mode of layer L1 (W). When capturing video data, be sure to set the mode.

Before executing this command, set attribute of layer L1 (W) by the **GdcDispDimension** command beforehand.

### 6.18.8 GdcCapSetVideoCaptureScale [Sets Scale of Video Capture]

**Interface**

GDC_BOOL **GdcCapSetVideoCaptureScale** (GDC_FIXED_SCALE *hscale*,
GDC_FIXED_SCALE *vscale*)

**Arguments**

| | |
|---|---|
| *hscale* | Horizontal scaling rate (default = 0x0800) |
| *vscale* | Vertical scaling rate (default = 0x0800) |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets scales for reducing video capture. Enlargement setting cannot be performed.
Sets horizontal scaling rate (*hscale*) that is cast to type GDC_FIXED_SCALE.
Sets vertical scaling rate (*vscale*) that is cast to type GDC_FIXED_SCALE.
Set to *hscale* by the value converted into the GDC_FIXED_SCALE type, which is the number of horizontal pixels of captured images divided by the number of horizontal pixels of images to be displayed. Set to *vscale* by the value converted into the GDC_FIXED_SCALE type, which is the number of vertical pixels of captured images divided by the number of vertical pixels of images to be displayed.
The type GDC_FIXED_SCALE is fixed point format that consists of 5 bits integer and 11 bits fraction. The range of the value is from "0xffff" (magnification = 1/31.99951171875) to "0x0800" (no magnification).
Initial value of *hscale* and *vscale* is "0x0800" (no magnification) respectively.
Examples of calculation of scaling rate are shown below (on next page).

[Example of calculation of scaling rate]

- When the image of size 720*576 is reduced to the size 648*490, scaling rate is calculated as below.

  Reduction of horizontal direction

  720pixel to 648pixel

  720/648=1.111

  1.111*2048=2275 (the value expressed in hexadecimal is "0x08e3")

  Reduction of vertical direction

  576line to 490line

  576/490=1.176

  1.176*2048=2408 (the value expressed in hexadecimal is "0x0968")

  A value to be set to *hscale* is "0x08e3".

  A value to be set to *vscale* is "0x0968".

- When 1/n times, scaling rate is calculated as below.

  *hscale* = n * 2048

  *vscale* = n * 2048

**Notes**

This command is scheduled to be abolished in the Graphics Driver of the next version. Therefore, use the **GdcCapSetCaptureArea** command in application programs newly made. Refer to the description of the **GdcCapSetCaptureArea** command for details.

### 6.18.9 GdcCapSetAttrMisc [Sets Attribute of Video Capture]

**Interface**

GDC_BOOL **GdcCapSetAttrMisc** (GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*target* Video capture attribute

| | |
|---|---|
| **GDC_CAP_ODD_MODE** | Capturing method |
| **GDC_CAP_CNV_MODE** | Non-interlace conversion mode |
| **GDC_CAP_BURST_MODE** | Video capture burst mode (for MB86295S or later) |
| **GDC_CAP_BUFFER_MODE** | Buffer mode (for MB86295S or later) |
| **GDC_CAP_CAPTURE_FORMAT** | I/O setting and writing format setting of video capture |

*param* Parameter corresponding to *target* (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_ATTRIBUTE** | Invalid attribute has been specified |

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets attribute of video capture.

**Notes**

Set the IOM register with the **GdcWriteHostRegister** command in case of using RGB input function of the video capture. Refer to the description of the **GdcWriteHostRegister** command for details.

Use only in case of no magnification when following combinations. Do not guarantee the operation in case of the reduction or enlargement by following combination.

- YCbCr input and RGB form writing (RGB = 5:5:5 or RGB = 8:8:8 form)
- RGB input and RGB form writing (RGB = 5:5:5 or RGB = 8:8:8 form)

When the writing format of the video capture data is a form of RGB = 8:8:8, the area in 4 bytes a pixel is needed, and set twice the value to the size and the width of the memory of the video capture buffer by the **GdcCapSetVideoCaptureBuffer**

command compared with the case of the YCbCr form.
Set color mode of the layer to 24-bit color mode by the **GdcDispDimension** command in case of displaying the image captured by the form at RGB = 8:8:8.
The amount of bus traffic increases at RGB = 8:8:8 form in the graphics memory, compared with the case at YCbCr form or RGB = 5:5:5 form, when the writing format of the video capture data is RGB = 8:8:8 form. There is a possibility that the displaying falls into disorder when the amount of bus traffic increases in the graphics memory, in that case, change the writing format to YCbCr form or RGB = 5:5:5 or, deal with by reducing the number of layers displayed at the same time.

(*1) Video capture attribute (*target*) and parameter (*param*) corresponding to each video capture attributes are shown below.

[Explanatory notes]

| Video capture attribute | Description of video capture attributes |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_CAP_ODD_MODE** | Specifies the capture method. |
| **GDC_CAP_EVEN_AND_ODD_MODE** | Captures both the odd number and the even number fields (default). |
| **GDC_CAP_ODD_ONLY_MODE** | Captures only the odd number field. |
| **GDC_CAP_CNV_MODE** | Specifies the non-interlace conversion mode of the image that is captured. |
| **GDC_CAP_CNV_BOB_MODE** | BOB mode (default) (*2) |
| **GDC_CAP_CNV_WEAVE_MODE** | WEAVE mode (*3) |

(*2) BOB mode: The mode is a frame in which the even field of the raster is averaged in interpolation then it is added to the odd field.

(*3) WEAVE mode: The mode is a frame in which the odd field and the even field are merged on the video capture buffer.

**GDC_CAP_BURST_MODE** Video capture burst mode

**GDC_CAP_BURST_STANDARD_MODE**
Standard mode (default)
**GDC_CAP_BURST_LONG_MODE**
Long mode

**GDC_CAP_BUFFER_MODE** Buffer mode

**GDC_CAP_BUFFER_RING_MODE**
Ring buffer (default)
**GDC_CAP_BUFFER_SINGLE_MODE**
Single buffer

**GDC_CAP_CAPTURE_FORMAT** I/O setting and writing format setting of video capture

**GDC_CAP_YCIN_YCOUT**
YCbCr input and YCbCr (16bpp) form writing (default)
Available for MB86291/86292/86294/86295S/86296S
**GDC_CAP_YCIN_RGB555OUT**
YCbCr input, RGB conversion, and RGB=5:5:5 form writing
Available for MB86295S or later in case of no magnification
**GDC_CAP_YCIN_RGB888OUT**
YCbCr input, RGB conversion, and RGB=8:8:8 form writing
Available only for MB86296S in case of no magnification
**GDC_CAP_RGBIN_YCOUT**
RGB input and YCbCr (16bpp) form writing
Available for MB86295S or later
**GDC_CAP_RGBIN_RGB555OUT**
RGB input and RGB=5:5:5 form writing
Available for MB86295S or later in case of no magnification
**GDC_CAP_RGBIN_RGB888OUT**
RGB input and RGB=8:8:8 form writing
Available only for MB86296S in case of no magnification
**GDC_CAP_NATIVE_RGBIN_RGB555OUT**
Native RGB input and RGB=5:5:5 form writing
Available only for MB86296S
**GDC_CAP_NATIVE_RGBIN_RGB888OUT**
Native RGB input and RGB=8:8:8 form writing
Available only for MB86296S

### 6.18.10 GdcCapSetInputDataCountNTSC [Sets Number of Video Capture Data for NTSC]

**Interface**

GDC_BOOL **GdcCapSetInputDataCountNTSC** (GDC_ULONG *blank_data*,
GDC_ULONG *valid_data*)

**Arguments**

| | |
|---|---|
| *blank_data* | The horizontal blanking interval, specified in number of dot clock cycles |
| *valid_data* | Number of data in the term of validity, specified in number of dot clock cycles |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets the input video stream number in using NTSC format.

This command is used to detect an error occurred. When the input data is not the same as the value set up by this command, an error occurs. The video capture status becomes the value other than "0" at this time.

The capturing is also continued when the error occurred.

### 6.18.11 GdcCapSetInputDataCountPAL [Sets Number of Video Capture Data for PAL]

**Interface**

GDC_BOOL **GdcCapSetInputDataCountPAL** (GDC_ULONG *blank_data*,
GDC_ULONG *valid_data*)

**Arguments**

| | |
|---|---|
| *blank_data* | The horizontal blanking interval, specified in number of dot clock cycles |
| *valid_data* | Number of data in the term of validity, specified in number of dot clock cycles |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets the input video stream number at the time of PAL format.

This command is used to detect an error occurred. When the input data is not same as the value set up by this command, an error occurs. The video capture status becomes the value other than zero at this time.

Also, capturing is continued when the error occurred.

## 6.18.12 GdcCapSetLPFMode [Sets Low Pass Filter Mode]

### Interface

GDC_BOOL **GdcCapSetLPFMode** (GDC_ULONG *vlpf_y*, GDC_ULONG *vlpf_c*,
GDC_ULONG *hlpf_y*, GDC_ULONG *hlpf_c*)

### Arguments

*vlpf_y* The vertical LPF coefficient code for luminance signals, range from "0" to "2", default is "0"

*vlpf_c* The vertical LPF coefficient code for chrominance signals, range from "0" to "2", default is "0"

*hlpf_y* The horizontal LPF coefficient code for luminance signals, range from "0" to "3", default is "0"

*hlpf_c* The horizontal LPF coefficient code for chrominance signals, range from "0" to "3", default is "0"

### Return value

**GDC_TRUE** Complete

### Error code

None

### Target GDC

MB86291 or later (MB86293 is excluded)

### Description

The mode (expression) of the low pass filter at the time of a video capture is set up. Coefficients in Table 6.18.12a are available for *vlpf_y* and *vlpf_*c. When a coefficient is "0", it outputs as it is, without covering a low pass filter. The initial value of *vlpf_y* and *vlpf_c* are "0".

**Table 6.18.12a Low Pass Filter Expression (for *vlpf_y* and *vlpf_c*)**

| Coefficient | Low Pass Filter Expression |
|---|---|
| 0 | y[i,j] = x[i,j] |
| 1 | y[i,j] = x[i,j-1]/4 + x[i,j]/2 + x[i,j+1]/4 |
| 2 | y[i,j] = x[i,j-1]*3/16 + x[i,j]*5/8 + x[I,j+1]*3/16 |

Notes:

x[i,j] : input pixel value at i-th column and j-th raster

y[i,j] : output pixel value at i-th column and j-th raster

Coefficients in Table 6.18.12b are available for *hlpf_y* and *hlpf_*c. When a coefficient is "0", it outputs as it is, without covering a low pass filter. The initial value of *hlpf_y* and *hlpf_c* are "0".

**Table 6.18.12b Low Pass Filter Expression (for hlpf_y and hlpf_c)**

| Coefficient | Low Pass Filter Expression |
|---|---|
| 0 | y[i,j] = x[i,j] |
| 1 | y[i,j] = x[i-1,j]/4 + x[i,j]/2 + x[i+1,j]/4 |
| 2 | y[i,j] = x[i-1,j]*3/16 + x[i,j]*5/8 + x[i+1,j]*3/16 |
| 3 | y[i,j] = x[i-2,j]*3/32 + x[i-1,j]/4 + x[i,j]*5/16+ x[I+1,j]/4 + x[i+2,j]*3/32 |

Notes:

x[i,j] : input pixel value at i-th column and j-th raster

y[i,j] : output pixel value at i-th column and j-th raster

### 6.18.13 GdcCapSetAttrVideo [Sets Various Modes of Video Capture]

**Interface**

GDC_BOOL **GdcCapSetAttrVideo** (GDC_ULONG *target*, GDC_ULONG *param*)

**Arguments**

*target* Video capture mode

| | |
|---|---|
| **GDC_CAP_VIDEO_SELECT** | Video format |
| **GDC_CAP_V_INTERPOLATION_MODE** | Interpolation mode in vertical direction |

*param* Parameter corresponding to *target* (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_ATTRIBUTE** | Invalid attribute has been specified |

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

It sets parameters concerning the input source in video capture.

(*1) Video capture mode (*target*) and parameter (*param*) corresponding to each video capture modes are shown below.

[Explanatory notes]

| Video capture mode | Description of video capture modes |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |

**GDC_CAP_VIDEO_SELECT** Video format

| | |
|---|---|
| **GDC_CAP_NTSC** | NTSC (default) |
| **GDC_CAP_PAL** | PAL |

**GDC_CAP_V_INTERPOLATION_MODE** Interpolation mode in vertical direction

| | |
|---|---|
| **GDC_ENABLE** | Interpolate (default) |
| **GDC_DISABLE** | No interpolate |

### 6.18.14 GdcCapStartVideoCapture [Starts Capturing Video Data]

**Interface**

GDC_BOOL **GdcCapStartVideoCapture** (void)

**Arguments**

None

**Return value**

**GDC_TRUE** Complete

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Starts capturing video data.

**Notes**

Call this command synchronized with the VSYNC interrupt.

Note that there is a possibility that the halt condition in an enlargement capture occurs when the interval between the VSYNC interrupt and the video capture starting is not within the regulation time in using the enlargement capture function with MB86295S.

Refer to the errata sheet of MB86295S for details.

### 6.18.15 GdcCapStopVideoCapture [Stops Capturing Video Data]

**Interface**

GDC_BOOL **GdcCapStopVideoCapture** (void)

**Arguments**

None

**Return value**

**GDC_TRUE** Complete

**Error code**

None

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Stops capturing video data.

**Notes**

Call this command synchronized with the VSYNC interrupt.

Note that there is a possibility that the halt condition in an enlargement capture occurs when the interval between the VSYNC interrupt and the video capture starting is not within the regulation time in using the enlargement capture function with MB86295S.

Refer to the errata sheet of MB86295S for details.

### 6.18.16 GdcCapSetCaptureArea [Sets Area of Capturing Video Data]

**Interface**

GDC_BOOL **GdcCapSetCaptureArea** (GDC_ULONG *src_x*, GDC_ULONG *src_y*,
GDC_ULONG *src_w*, GDC_ULONG *src_h*,
GDC_ULONG *dest_w*, GDC_ULONG *dest_h*)

**Arguments**

| | |
|---|---|
| *src_x* | x coordinate on the upper left of area cut out from input image |
| *src_y* | y coordinate on the upper left of area cut out from input image |
| *src_w* | Width of area cut out from input image, in pixels |
| *src_h* | Height of area cut out from input image, in pixels |
| *dest_w* | Width of image after enlargement or reduction, in pixels |
| *dest_h* | Height of image after enlargement or reduction, in pixels |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_ILLEGAL_CAPTURE_SCALE**

An illegal video capture scale was specified

**Target GDC**

MB86291 or later (MB86293 is excluded)

**Description**

Sets the range of the target image for capturing video data and the size of the image after enlargement or reduction.

Nothing is processed, **GDC_TRUE** is set to the return value, and it returns if any one of src_w, src_h, dest_w, and dest_h is "0".

Error code **GDC_ERR_ILLEGAL_CAPTURE_SCALE** is set, **GDC_FALSE** is set to the return value, and it returns if the enlargement function is set for MB86291/86292/86294.

Error code **GDC_ERR_ILLEGAL_CAPTURE_SCALE** is set, **GDC_FALSE** is set to the return value, and it returns if the combination of horizontal enlargement and vertical reduction or the combination of horizontal reduction and vertical enlargement is set for MB86295S.

Set the width of the window and the height of the layer L1 as the value after enlargement on enlarging. Neither the width of the window nor the height of the layer L1 are set on reducing, and so set the values appropriately by the

**GdcDispSetLayerWindow** command.

The example of using this command is shown as follows.

[Example of using the **GdcCapSetCaptureArea** command]

```
/* Sets layer L1 parameter */
GdcDispDimension(GDC_DISP_LAYER_L1,GDC_ENABLE,
        GDC_16BPP_FORMAT, GDC_FLIPMODE_0,
        0x0, 0x0, 736, 480);
GdcCapSetWindowMode(GDC_CAP_YC_MODE, GDC_CAP_CAPTURE_MODE);

/* Sets video capture buffer (size=3 frame, 1frame=736*480*2[byte]) */
GdcCapSetVideoCaptureBuffer(0x96000,(0x96000+(736*(480*3-2)*2)),736*2/64);

/* Sets video capture data format */
GdcCapSetAttrMisc(GDC_CAP_CAPTURE_FORMAT , GDC_CAP_YCIN_YCOUT);

/* Sets video capture area (display resolution is SVGA) */
GdcCapSetCaptureArea(0, 0, 720, 480, 800, 600);

/* Starts video capture */
GdcCapStartVideoCapture();
```

### Notes

Before calling this command, specify **GDC_CAP_CAPTURE_FORMAT** for the first argument of the **GdcCapSetAttrMisc** command, specify the second argument appropriately, and set Input/Output and the writing format of the video capture data.

Do not use this command together with the following Driver Command. The following Driver Command are scheduled to be abolished in the Graphics Driver of the next version.

- **GdcCapSetImageArea** command
- **GdcCapSetVideoCaptureScale** command

### 6.18.17 GdcCapSetMaxHorizontalPixel [Sets Maximum, Horizontal Pixels of Input Images]

**Interface**

GDC_BOOL **GdcCapSetMaxHorizontalPixel** (GDC_ULONG *pixel*)

**Arguments**

| | |
|---|---|
| *pixel* | Specify the value by the number of horizontal pixels and the even number.<br>[For MB86294]<br>Specify the value in the range from "0" to "720" pixels. The default is "720" pixels.<br><br>[For MB86295S or later]<br>Specify the value in the range from "0" to "840" pixels. The default is "720" pixels. |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86294 or later

**Description**

Sets maximum, horizontal pixels of input images. Doesn't need to call this command if the number of horizontal pixels of input images is "720" or less.

Example: In the case of capturing the image of the resolution of 800x700 with MB86296S,

- The argument pixel of this command is set to "800".
- The argument of the **GdcCapSetMaxVerticalPixel** command is set to "700".

### 6.18.18 GdcCapSetMaxVerticalPixel [Sets Maximum, Vertical Pixels of Input Images]

**Interface**

GDC_BOOL **GdcCapSetMaxVerticalPixel** (GDC_ULONG *pixel*)

**Arguments**

| | |
|---|---|
| *pixel* | Number of vertical pixels<br>Specify the value in the range from "1" to "1023" pixels. The default is "625" pixels. |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86296S

**Description**

Sets maximum, vertical pixels of input images. Does not need to call this command if the number of vertical pixels of input images is "625" or less.

Example: In the case of capturing the image of the resolution of 800x700 with MB86296S

- The argument of the **GdcCapSetMaxHorizontalPixel** command is set to "800".
- The argument pixel of this command is set to "700".

**Notes**

Do not call this command because the function to set the number of maximum, vertical pixels of input images is not provided in MB86294 and MB86295S. Do not guarantee the operation if this command is called for MB86294 or MB86295S.

### 6.18.19 GdcCapSetRGBInputTiming [Sets Range of RGB Input]

**Interface**

GDC_BOOL **GdcCapSetRGBInputTiming** (GDC_ULONG *hcycle*,
GDC_ULONG *hstart*, GDC_ULONG *harea*,
GDC_ULONG *vstart*, GDC_ULONG *varea*)

**Arguments**

| | |
|---|---|
| *hcycle* | Horizontal cycle, in pixels |
| *hstart* | Start position of horizontal effective pixel, in pixels |
| *harea* | Horizontal effective pixel, in pixels |
| *vstart* | Start position of vertical effective pixel, in pixels |
| *varea* | Vertical effective pixel, in pixels |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86295S or later

**Description**

Sets the input range of RGB input in the video capture function.

### 6.18.20 GdcCapSetRGBInputSync [Sets RGB Input Synchronous Signal]

**Interface**

GDC_BOOL **GdcCapSetRGBInputSync** (GDC_ULONG *target*,
GDC_ULONG *param*)

**Arguments**

*target* Attribute of RGB input synchronous signal

| | |
|---|---|
| **GDC_CAP_RGB_VSYNC_POLARITY** | Edge detection mode of VSYNC |
| **GDC_CAP_RGB_HSYNC_POLARITY** | Edge detection mode of HSYNC |

*param* Parameter corresponding to *target* (*1)

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

| | |
|---|---|
| **GDC_ERR_INVALID_ATTRIBUTE** | Invalid attribute has been specified |

**Target GDC**

MB86295S or later

**Description**

Set the synchronized signal when RGB is input.

(*1) Attribute of RGB input synchronous signal (*target*) and parameter (*param*) corresponding to each attribute of RGB input synchronous signal are shown below.

[Explanatory notes]

| Attribute of RGB input synchronous signal | Description of attribute of RGB input synchronous signal |
|---|---|
| Parameter 1 that can be set | Description of parameter 1 |
| Parameter 2 that can be set | Description of parameter 2 |
| : | : |
| **GDC_CAP_RGB_VSYNC_POLARITY** | Edge detection mode of VSYNC. |
| **GDC_POSITIVE_EDGE** | Positive edge |
| **GDC_NEGATIVE_EDGE** | Negative edge (default) |

| | |
|---|---|
| **GDC_CAP_RGB_HSYNC_POLARITY** | Edge detection mode of HSYNC. |
| **GDC_POSITIVE_EDGE** | Positive edge |
| **GDC_NEGATIVE_EDGE** | Negative edge (default) |

### 6.18.21 GdcCapSetRGBMatrix [Sets RGB to YCbCr Transformation Matrix]

**Interface**

GDC_BOOL **GdcCapSetRGBMatrix** (const GDC_ULONG **matrix*)

**Arguments**

| | |
|---|---|
| *matrix* | Pointer to matrix values to be set |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86295S or later

**Description**

Specifies the conversion parameter to convert the color data from the RGB form to the YCbCr form in the video capture function by the matrix form.

Specify the pointer to array {m0, m1, ..., m11} corresponding to the following 3x4 matrix M for matrix, and express each element (from m0 to m11) by the fixed point form of eight digits in the fraction part.

$$M=\begin{pmatrix} m0 & m1 & m2 & m3 \\ m4 & m5 & m6 & m7 \\ m8 & m9 & m10 & m11 \end{pmatrix}$$

Y = m0*R + m1*G + m2*B + m3
Cb = m4*R + m5*G + m6*B + m7
Cr = m8*R + m9*G + m10*B + m11

### 6.18.22 GdcCapStartClock [Starts Video Capture Clock Supply]

#### Interface

GDC_BOOL **GdcCapStartClock** (void)

#### Arguments

None

#### Return value

| | |
|---|---|
| **GDC_TRUE** | Complete |

#### Error code

None

#### Target GDC

MB86296S

#### Description

Starts supplying the clock of the video capture.

### 6.18.23 GdcCapStopClock [Stops Video Capture Clock Supply]

**Interface**

GDC_BOOL **GdcCapStopClock** (void)

**Arguments**

None

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |

**Error code**

None

**Target GDC**

MB86296S

**Description**

Stops supplying the clock of the video capture.

# 6.19 I²C Control Commands

## 6.19.1 GdcI2CGetBusStatus [Gets I²C Bus Status]

### Interface

GDC_ULONG **GdcI2CGetBusStatus** (void)

### Arguments

None

### Return value

I²C bus status in the following format (value of BSR register):

| bit 31 | ... | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Don't care | | x | x | x | x | x | x | x | x |

'x': used bit

bit7: Detects START/STOP condition
0:STOP condition
1:START condition (The bus is in use)
bit6: Detects repeated START condition
0:Repeated START condition was not detected
1:START condition was detected again while bus is in use
bit5: Detects arbitration lost
0:Arbitration lost was not detected
1:Arbitration lost occurred during master transmission
bit4: Status of acknowledge
0:No acknowledge
1:Acknowledge
bit3: Status of data transfer
0:Receiving state
1:Transmitting state
bit2: Detects addressing
0:Addressing was not performed in a slave mode
1:Addressing was performed in a slave mode
bit1: Detects "General call address (00h)"
0:"General call address (00h)" was not received in a slave mode
1:"General call address (00h)" was received in a slave mode
bit0: Detects the first byte
0:Received data is not the first byte
1:Receiverd data is the first byte (address data)
All other bits: Don't care

**Figure 6.19.1 I²C Bus Status**

### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

### Description

Reads BSR (Bus Status Register) and returns I²C bus status.

### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.

It is no problem to access I²C registers in DMA transferring.

### 6.19.2 GdcI2CSetBusControl [Controls I²C Bus]

#### Interface

void **GdcI2CSetBusControl** (GDC_UCHAR *param*)

#### Arguments

*param* The parameter for I²C bus control (the following format)

| bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| x | x | x | x | x | x | x | x |

'x': used bit

bit7: Flag bit for request of bus error interruption
0:Clears a request of bus error interruption
1:Invalid
bit6: Permits bus error interruption
0:Prohibition of bus error interruption
1:Permission of bus error interruption
bit5: Generates START condition
0: Invalid
1:START condition is generated again at the time of master transmission
bit4: Selects master/slave mode
0:Becomes a slave mode after generation of STOP condition and completing transfer
1:Becomes a master mode, generates START condition and starts transfer
bit3: Permits acknowledge generation at the time of data reception
0:Acknowledge is not generated
1:Acknowledge is generated
bit2: Permits acknowledge generation when general call address (00h) is received
0:Acknowledge is not generated
1:Acknowledge is generated
bit1: Permits interruption
0:Prohibition of interrupt
1:Permission of interrupt
bit0: Flag bit for request of interruption for transfer end
0:Clears the flag
1: Invalid

**Figure 6.19.2 Parameter for I²C Bus Control**

#### Return value

None

### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

### Description

Controls I²C bus by writing BCR (Bus Control Register) to *param*.

### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.

It is no problem to access I²C registers in DMA transferring.

### 6.19.3 GdcI2CGetBusControlStatus [Gets I²C Bus Control Status]

#### Interface

GDC_ULONG **GdcI2CGetBusControlStatus** (void)

#### Arguments

None

#### Return value

I²C bus control status in the following format (value of BCR register):

| bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| x | x | Don't care | x | x | x | x | x |

'x': used bit

bit7: Flag bit for request of bus error interruption
0:A bus error was not detected
1:Invalid START condition or STOP condition was detected while data transfer
bit6: Permission state of bus error interruption
0:Prohibition of bus error interruption
1:Permission of bus error interruption
bit5: Read value of this bit is always "0"
bit4: Master/slave mode
0:Slave mode
1:Master mode
bit3: Permission state of acknowledge generation at the time of data reception
0:Acknowledge is not generated
1:Acknowledge is generated
bit2: Permission state of acknowledge generation when general call address (00h) is received
0:Acknowledge is not generated
1:Acknowledge is generated
bit1: Permission state of interruption
0:Prohibition of interrupt
1:Permission of interrupt
bit0: Flag bit for request of interruption for transfer end
0:The transfer is not ended
1:It is set when 1 byte transfer including the acknowledge bit is completed and it corresponds to the following conditions
- It is a bus master
- It is an addressed slave
- It received "General call address (00h)"

It was going to generate START condition while other systems by which arbitration lost occurred used the bus

**Figure 6.19.3 I²C Bus Control Status**

### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

### Description

Reads BCR (Bus Control Register) and returns I²C bus control status.

### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.

It is no problem to access I²C registers in DMA transferring.

### 6.19.4 GdcI2CSetClock [Sets I²C Clock]

#### Interface

void **GdcI2CSetClock** (GDC_UCHAR *param*)

#### Arguments

*param* Parameter for setup of I²C clock (*1) (the following format)

| bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | x | x | x | x | x | x | x |

'x': used bit

bit6: Selects standard-mode/high-speed-mode
0:Standard-mode
1:High-speed-mode
bit5: Permits I²C operation
0:Prohibition of operation
1:Permission of operation
bit4-0: Frequency of a transfer clock
From "00000" to "11111"
All other bits: 0

**Figure 6.19.4 I²C Clock Parameter**

(*1) For detail information about I²C clock, refer to the chapter of the "clock control register" of "I²C Interface specifications".

#### Return value

None

#### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

#### Description

Sets I²C clock setting parameter to CCR (Clock Control Register).

#### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.
It is no problem to access I²C registers in DMA transferring.

### 6.19.5 GdcI2CGetClock [Gets I²C Clock Control Status]

#### Interface

GDC_ULONG **GdcI2CGetClock** (void)

#### Arguments

None

#### Return value

I²C clock (*1) control status in the following format (value of CCR register):

| bit 31 | ... | bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | ... | 1 | x | x | x | x | x | x | x |

'x': used bit

bit7: Not used (read value is always 1)
bit6: Selects standard-mode/high-speed-mode
0:Standard-mode
1:High-speed-mode
bit5: Permits I²C operation
0:Prohibition of operation
1:Permission of operation
bit4-0: Frequency of a transfer clock
From "00000" to "11111"
All other bits: 0

**Figure 6.19.5 I²C Clock Parameter**

(*1) For detail information about I²C clock, refer to the chapter of the "clock control register" of "I²C Interface specifications".

#### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

#### Description

Reads CCR (Clock Control Register) and returns I²C clock control status.

#### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.
It is no problem to access I²C registers in DMA transferring.

### 6.19.6 GdcI2CSetData [Sets Transfer Data]

#### Interface

void **GdcI2CSetData** (GDC_UCHAR *param*)

#### Arguments

*param* A data to be transferred

#### Return value

None

#### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

#### Description

Sets a data to be transferred by writing DAR (DAta Register) to *param*.

#### Notes

When accessing I²C registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access I²C registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.

It is no problem to access I²C registers in DMA transferring.

### 6.19.7 GdcI2CGetData [Gets Transfer Data]

#### Interface

GDC_ULONG **GdcI2CGetData** (void)

#### Arguments

None

#### Return value

Value of transfer data

#### Target GDC

MB86291S/86291AS/86292S/86294S/86295S/86296S

#### Description

Reads DAR (DAta Register) and returns value of transfer data.

#### Notes

When accessing $I^2C$ registers while local display list is being transferred, XRDY is not returned and the Graphics Controller hangs. Therefore, access $I^2C$ registers after confirming that local display list is not being transferred. The local display list transferring status can be acquired by the **GdcGetLocalDisplayListTransferStatus** command.

It is no problem to access $I^2C$ registers in DMA transferring.

# 6.20 Register Control Commands

## 6.20.1 GdcWriteHostRegister [Writes Host Interface Register]

### Interface

void **GdcWriteHostRegister** (GDC_ULONG *offset*, GDC_ULONG *data*)

### Arguments

*offset* Offset from the top area of the host interface register (byte unit)
Only the following macro can be used.
**GDC_REG_IOM** Offset of IOM register (0x000000A8)

*data* Data to be written

### Return value

None

### Target GDC

MB86295S or later

### Description

It writes the value to the host interface register.

### Notes

Call this command only from one task so that there will not cause a problem of simultaneous update if this command is called from two or more tasks.

Use this command only for the usage in which the value is written in the IOM register. Do not guarantee operation if used for other usages.

The terminal of MB86295S/86296S is shared with RGB input of the video capture, GPIO and Serial Interface. Therefore, it is not possible to use the functions sharing these terminals simultaneously.

[For RGB input function of the video capture]

```
#define GDC_USE_RGB_INPUT       0x00000020L
GDC_ULONG iom;
iom = GdcReadHostRegister(GDC_REG_IOM) | GDC_USE_RGB_INPUT;
GdcWriteHostRegister(GDC_REG_IOM, iom);
```

### 6.20.2 GdcReadHostRegister [Reads Host Interface Register]

#### Interface

GDC_ULONG **GdcReadHostRegister** (GDC_ULONG *offset*)

#### Arguments

| | |
|---|---|
| *offset* | Offset from the top area of the host interface register (byte unit)<br>Only the following macro can be used.<br>**GDC_REG_IOM** Offset of IOM register (0x000000A8) |

#### Return value

The value of host interface register

#### Target GDC

MB86295S or later

#### Description

It reads the value of the host interface register, and returns the value.

#### Notes

Call this command only from one task so that there will not cause the problem of simultaneous update if this command is called from two or more tasks.

Use this command only for the usage in which the value is read from the IOM register. Do not guarantee operation if used for other usages.

# 7 System Dependent Commands

The list of System Dependent Command is shown. Application designers need to create these commands.

## 7.1 System Dependent Commands

System Dependent Commands are shown in Table 7.1.

**Table 7.1 System Dependent Command list**

| No. | Command Name | Function |
|---|---|---|
| 1 | **GdcFlushDisplayList** | Transfers display list |
| 2 | **XGdcSwitchDLBuf** | Changes DL buffer |
| 3 | **GdcWait** | Waiting routine |

# 8 System Dependent Command Interface

This chapter explains the call interface and the content of processing of the System Dependent Commands.

The following descriptions are as follows.

**Interface**

Command interface

**Arguments**

Description of arguments

**Return value**

Return values and the description of them

**Error code**

Error code and error name when the command terminates abnormally (When the return value returns by **GDC_FALSE**).

This item is omitted if the command has no return value.

**Called by**

Command name of the calling

**Description**

Description of the command

**Notes**

Notes

This item omitted if there are no notes.

# 8.1 Command Interface

## 8.1.1 GdcFlushDisplayList [Transfers Display List]

### Interface

void **GdcFlushDisplayList** (GDC_ULONG **src*, GDC_ULONG *count*)

### Arguments

| | |
|---|---|
| *src* | Source address (DL buffer's address) |
| *count* | Transfer count, range from "1" to "4294967295" |

### Return value

None

### Called by

**XGdcFlush** command

**XGdcFlushEx** command

### Description

This command is to transfer a display list of the size specified by "*count*" started from the source address specified by "*src*". The "*src*" specifies the DL buffer address mapped to the host CPU address field. The unit of "*count*" is what specified by the **GdcSetDMAMode** command (32byte or 4byte). If the **GdcSetDMAMode** command is not applied since DMA is not used, this unit is set to 4byte. For the display list transfer, the following four methods are applicable. For each procedure, refer to the description [Display list transfer procedure] as follows:

- DMA transfer
- Transfer of local display list
- Transfer of local display list by BCU
- CPU transfer

[Display list transfer procedure]

*** DMA transfer**

This is a method of display list transfers utilizing the DMA controller of the host CPU (the Graphics Controller does not contain a DMA controller). The operation procedure of this case is shown as follows.

(Notes) Prior to call this command, DMA transfer mode must be appropriately set on both DMAC (DMA controller) and the Graphics Controller.

(1) Check DMA transfer enable/disable

Checks the appropriate operation mode check of the DMAC and wait till it will be ready to accept a new DMA transaction request.

(2) Check of the completion of drawing

- MB86290A: Wait until bit 1-0 of the CTR register is set to 0x0
- MB86291 or later: Wait until bit 1-0 of the GCTR register is set to 0x0

(3) Set DMA

According to the applied procedure for the DMA, set the following parameter.

- Transfer count (the value specified in "*count*").
- Source address (the address specified in "*src*").
- Destination address (display list FIFO of the Graphics Controller).

(4) Set transfer count (the Graphics Controller side)

Sets DMA transfer count (the value specified in "*count*") to DTC (DMA Transfer Count) register.

(5) Start DMA transaction

Performs start up operation according to DMA used

(6) Issue the DMA request (the Graphics Controller side)

Sets "1" to DRQ (DMA ReQuest) register.

(7) Wait for the completion of DMA transfer if necessary (When DL buffer number is "1")

[Remark]

When the unit of transfer count is 32byte and the total byte size of the display list is not a multiple of 32byte, insert the NOP command until it reaches the multiple in 32byte by the **XGdcFlush** command or the **XGdcFlushEx** command.

[Example]

```
/* Top address of the host interface register field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define HOSTBASE               0x########

/* Top address of drawing control register field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define DRAWBASE       0x########

#define WRITE_DTC(i)    ( *(GDC_ULONG*)(HOSTBASE+0x00) = (i) )
#define WRITE_DRQ(i)    ( *(GDC_ULONG*)(HOSTBASE+0x18) = (i) )

#ifdef GDC_MB86290A /* for MB86290A */
#define FIFO_ADDRESS    (DRAWBASE+0x4a0)
#define DRAW_STATUS     (DRAWBASE+0x400)
#else /* for MB86291 or later */
#define FIFO_ADDRESS    (DRAWBASE+0x8400)
#define DRAW_STATUS     (DRAWBASE+0x8000)
#endif
#define READ_DRAW_REG(reg)      *(volatile unsigned long*)(reg)
#define GDC_BUSY()              ((READ_DRAW_REG(DRAW_STATUS) & 0x3));

void GdcFlushDisplayList(GDC_ULONG *src, GDC_ULONG count){
   /* Polling for DMA ready DMA */
   /* Please create DMA_BUSY function according to the environment of use */
   while( DMA_BUSY() );

   /* Polling of drawing end */
   while( GDC_BUSY() );

   /* Sets transfer count */
   /* Please create SET_DMA_COUNT function according to the environment of use */
   SET_DMA_COUNT(CHANNEL0, count);

   /* Sets source address */
   /* Please create SET_DMA_SRC function according to the environment of use */
   SET_DMA_SRC(CHANNEL0, src);

   /* Sets destination address */
   /* Please create SET_DMA_DEST function according to the environment of use */
   SET_DMA_DEST(CHANNEL0, FIFO_ADDRESS);

   /* Sets transfer count (Graphics Controller) */
   WRITE_DTC(count);

   /* Trigger of DMA transaction */
   /* Please create DMA_START function according to the environment of use */
   DMA_START();

   /* Issue of external DMA request */
   WRITE_DRQ(1);

#ifdef SINGLE_DL_BUFFER
   /* Wait for the next DL buffer write to be ready */
   while( DMA_BUSY() );
#endif
}
```

*** Transfer of local display list**

This is a method of the display list transfers utilizing the bus master function of the Graphics Controller.

Transfer count is 4byte unit.

In this case, the DL buffer must be located to the graphics memory of the Graphics Controller. And the source address "*src*" must be converted to the local address of the Graphics Controller. The operation procedure of this case is shown as follows:

(1) Check transfer enable/disable

Check the status of LSTA (display List transfer STAtus) register and wait until it will be "0".

(2) Check of the completion of drawing

- MB86290A: Wait until bit 1-0 of the CTR register is set to 0x0
- MB86291 or later: Wait until bit 1-0 of the GCTR register is set to 0x0

(3) Set source address

Set the source address to LSA (display List Source Address) register. The transferring source address is a value calculated by the following expressions:

("*src*" value) – (top address of the graphics memory field)

(4) Set transfer count

Set the transfer count ("*count*" value) to LCO (display List COunt) register.

(5) Start the transaction

Set "1" to LREQ (display List transfer REQuest) register.

(6) Wait for the completion of transfer if necessary (When DL buffer number is "1")

Same as (1).

[Example]

```
/* Top address of host interface register field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define HOSTBASE              0x########

/* Top address of drawing control register field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define DRAWBASE       0x########

/* Top address of the graphics memory field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define MB86290_BASE    0x########

#define READ_LSTA()     *((volatile GDC_ULONG*)(HOSTBASE+0x10))
#define WRITE_LSA(i)    ( *((GDC_ULONG*)(HOSTBASE+0x40)) = (i) )
#define WRITE_LCO(i)    ( *((GDC_ULONG*)(HOSTBASE+0x44)) = (i) )
#define WRITE_LREQ(i)   ( *((GDC_ULONG*)(HOSTBASE+0x48)) = (i) )

#ifdef GDC_MB86290A /* for MB86290A */
#define DRAW_STATUS     (DRAWBASE+0x400)
#else /* for MB86291 or later */
#define DRAW_STATUS     (DRAWBASE+0x8000)
#endif
#define READ_DRAW_REG(reg)      *(volatile GDC_ULONG*)(reg)
#define GDC_BUSY()              ((READ_DRAW_REG(DRAW_STATUS) & 0x3));

void GdcFlushDisplayList(GDC_ULONG *src, GDC_ULONG count){
   GDC_ULONG      src_local;

   /* Polling of transfer ready */
   while( READ_LSTA() );

   /* Polling of drawing end */
   while( GDC_BUSY() );

   /* Source address set */
   src_local = (GDC_ULONG)src – MB86290_BASE;
   WRITE_LSA(src_local);

   /* Transfer count set */
   WRITE_LCO(count);

   /* Trigger */
   WRITE_LREQ(1);

#ifdef SINGLE_DL_BUFFER
   /* Wait for next the DL buffer write to be ready */
   while( READ_LSTA() );
#endif
}
```

***Transfer of local display list by BCU**

This is a method of the display list transfers utilizing the BCU of the Graphics Controller. Only for MB86295 or later can be used. Transfer count can be specified arbitrarily.

In this case, the DL buffer must be located to the graphics memory of the Graphics Controller. And the source address "src" must be converted to the local address of the Graphics Controller. The operation procedure of this case is shown as follows:

(1) Check transfer enable/disable

In first-time transmission, this procedure is unnecessary.

In transmission of the 2nd henceforth, the BST register is read, and it waits until bit 23-0 of the BST register is set to "0".

(2) Check of the completion of drawing

Wait until bit 1-0 of the GCTR register is set to "0x0"

(3) Set source address

Set the source address to BSA register. The transferring source address is a value calculated by the following expressions:

("*src*" value) – (top address of the graphics memory field)

(4) Set destination address

Set the destination address to BDA register.

(5) Set transmission mode

Set the transmission mode (internal transfer) to BSR register.

(6) Set transfer count

The following parameters are set to BCR register.

- Transfer unit (this example is 4byte)
- Source address is increment
- Destination address is not increment
- Transfer count ("*count*" value)

(7) Start the transaction

Set "1" to BER register.

(8) Wait for the completion of the transfer (in case of single DL buffer mode)

Same as (1).

[Example]

```
/* Please set a suitable value to the following ########
   according to the environment of use*/
#define HOSTBASE                        0x########
#define DRAWBASE                        0x########
#define MB86290_BASE                    0x########
#define BST_REG                         (HOSTBASE + 0x8014L)
#define BSA_REG                         (HOSTBASE + 0x8000L)
#define BDA_REG                         (HOSTBASE + 0x8004L)
#define BCR_REG                         (HOSTBASE + 0x8008L)
#define BSR_REG                         (HOSTBASE + 0x800cL)
#define BER_REG                         (HOSTBASE + 0x8010L)
#define WRITE_HOST_REGISTER(reg,data)       *(unsigned long*)(reg) = (unsigned long)data
#define READ_HOST_REGISTER(reg)   *(volatile GDC_ULONG*)(reg)
#define READ_DRAW_REGISTER(reg)  *(volatile GDC_ULONG*)(reg)
#define GDC_BUSY()              ((READ_DRAW_REGISTER(DRAWBASE+0x8000) & 0x3))

void GdcFlushDisplayList(GDC_ULONG *src, GDC_ULONG count){
    GDC_ULONG    temp;
    GDC_ULONG    src_local;
    static int transfer = 0;

    if(transfer==1){
        /* Waits during transfer */
        temp = READ_HOST_REGISTER(BST_REG) & 0x00ffffff;
        while(temp!=0x0){
            temp = READ_HOST_REGISTER(BST_REG) & 0x00ffffff;
        }
        transfer = 0;
    }
    /* Polling of drawing end */
    while( GDC_BUSY() );

    /* Source address set */
    src_local = (GDC_ULONG)src - MB86290_BASE;
    WRITE_HOST_REGISTER(BSA_REG, src_local);

    /* Destination address set */
    WRITE_HOST_REGISTER(BDA_REG, 0x01ff8400);

    /* Transmission mode set */
    WRITE_HOST_REGISTER(BSR_REG, 0x00000002);

    /* Transfer count set */
    WRITE_HOST_REGISTER(BCR_REG, 0x21000000 | count);

    /* Trigger */
    WRITE_HOST_REGISTER(BER_REG, 0x1);
    transfer = 1;

#ifdef SINGLE_DL_BUFFER
    /* Waits during transfer */
    temp = READ_HOST_REGISTER(BST_REG) & 0x00ffffff;
    while(temp!=0x0){
        temp = READ_HOST_REGISTER(BST_REG) & 0x00ffffff;
    }
#endif
}
```

***CPU transfer**

This is a method of writing the display list directly in the Graphics Controller's display list FIFO in the program. The unit of transferring is 4byte.

The processing procedure of this command when CPU transferring is used is shown below.

Repeat (1) through (4) for the times specified by "*count*".

(1) Acquire the display list FIFO status.
Calls the **GdcGetFIFOStatus** command and acquire the display list FIFO status information.

(2) Check the display list FIFO status.
Checks the empty entries of the display list FIFO from the above status information. If FIFO is full, keep repeating (1) and (2) till open entries will be available.

(3) Transfer 4byte of data from the source address to the display list FIFO.

(4) Post increment (+4) source address.

[Example]

```
/* Top address of drawing control register field */
/* Please set a suitable value to the following ########
  according to the environment of use*/
#define DRAWBASE        0x########
#define FIFO_FULL       0x2

#ifdef GDC_MB86290A
/* for MB86290A */
#define WRITE_FIFO(i)   ( *(volatile GDC_ULONG*)(DRAWBASE+0x4a0) = (i) )
#else
/* for MB86291 or later */
#define WRITE_FIFO(i)   ( *(volatile GDC_ULONG*)(DRAWBASE+0x8400) = (i) )
#endif

void GdcFlushDisplayList(GDC_ULONG *src, GDC_ULONG count){
   int  i;

   for(i = 0; i < count; i++){
        /* If FIFO is full, wait until open entry will be available */
        while(GdcGetFIFOStatus() & FIFO_FULL);

        /* Transfers data to the FIFO */
        WRITE_FIFO(*src++);
   }

}
```

### 8.1.2 XGdcSwitchDLBuf [Changes DL Buffer]

**Interface**

GDC_BOOL **XGdcSwitchDLBuf** (GDC_CTX **drvctx*)

**Arguments**

| | |
|---|---|
| *drvctx* | Pointer to context |

**Return value**

| | |
|---|---|
| **GDC_TRUE** | Complete |
| **GDC_FALSE** | Incomplete |

**Error code**

**GDC_ERR_SWITCH_DL_BUF_FAILED**

The **XGdcSwitchDLBuf** command has finished abnormally

**Called by**

The command for drawing control (The command which generates a display list)

**Description**

When the display list to be generated is larger than the availability of the current DL buffer, the command for drawing control calls this command, in order to change the current DL buffer to another DL buffer. The operations that this command should perform are to search applicable DL buffer and to change the current DL buffer to the applicable DL buffer (by using the **XGdcSetDLBuf** command). When each other DL buffer is in the inapplicable state (for example, the waiting display list for transmission is stored in DL buffer), wait until the DL buffer is in the applicable state then change the DL buffer. For a certain reason, changing of the DL buffer is impossible and when interrupting the processing, set **GDC_ERR_SWITCH_DLBUF_FAILED** to the error code (by using the **XGdcSetErrCode** command) and return **GDC_FALSE**. Thereby, the command for drawing control of a calling interrupts the processing.

**Notes**

- When the generated display list needs to be saved, it is necessary to mount so that applicable DL buffer may not be chosen with this command.
- By calling this command, a display list is outputted ranging over two or more DL buffers. In this case, since two or more DL buffers must be transmitted to the Graphics Controller in application programs, it is necessary to mount the mechanism that which DL buffer should be transmitted to the Graphics Controller in which turn is detected from

application programs. Moreover, the display list generated ranging over two or more DL buffers must be transmitted continuously to the Graphics Controller.

- When this command ends abnormally, application programs need to discard the display list generated in the middle.

### 8.1.3 GdcWait [Waiting Routine]

**Interface**

void **GdcWait** (GDC_ULONG *msec*)

**Arguments**

| | |
|---|---|
| *msec* | Waiting time (millisecond) |

**Return value**

None

**Called by**

**GdcInitialize** command

**Description**

Return after the time progress specified by *msec*.

After initializing setting of the Graphics Controller in the **GdcInitialize** command, this command is called in order to consume time required for until stable of operation.

Even if it consumes more than the time specified by *msec*, there is no problem.

# 9 Appendix

## 9.1 Display List Size that Driver Command Generates

Table 9.1 shows the size (number of word (*1)) of the display list that each Driver Command generates.

(*1) One word is 4 bytes.

**Table 9.1 Display List Size List**

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Drawing Frame Setting Commands | XGdcDrawDimension | 6 | |
| | XGdcSetZPrecision | 2 | |
| | XGdcBufferCreateZ | 2 | |
| | XGdcBufferCreateC | 2 | |
| | XGdcBufferClearZ | 19 | |
| | XGdcBufferClearC | 3 | |
| | XGdcDrawClipFrame | 5 | |
| | XGdcSetAlphaMapBase | 2 | |
| Drawing Attribute Setting Commands | XGdcColor | 2 | |
| | XGdcBackColor | 2 | |
| | XGdcClipMode | 2 | |
| | XGdcSetAttrLine | 2 to 4 | It differs depending on the specified drawing attribute |
| | XGdcSetAttrSurf | 2 to 4 | It differs depending on the specified drawing attribute |
| | XGdcSetAttrTexture | 2 | |
| | XGdcSetAttrBlt | 2 | |
| | XGdcSetAlpha | 2 | |
| | XGdcSetLinePattern | 2 | |
| | XGdcSetTextureBorder | 2 | |
| | XGdcSetRop | • MB86290A/86291/86292<br>5<br>• MB86293 of later<br>14 | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Attribute Setting Commands for Object Coordinate System | **XGdcGeoSetAttrLine** | 2 | |
| | **XGdcGeoSetAttrSurf** | 2 | |
| | **XGdcGeoLoadMatrix[f]** | 19 | |
| | **XGdcGeoNdcDcViewportCoef[f]** | 7 | |
| | **XGdcGeoNdcDcDepthCoef[f]** | 5 | |
| | **XGdcGeoViewVolumeXYClip[f]** | 7 | |
| | **XGdcGeoViewVolumeZClip[f]** | 5 | |
| | **XGdcGeoViewVolumeWminClip[f]** | 4 | |
| | **XGdcGeoSetLogOutBase** | 2 | |
| | **XGdcGeoSetLogOutMode** | 0 | The display list is not generated so that only the value of the context may change |
| | **XGdcGeoShadowXY** | 2 | |
| | **XGdcGeoOverlapZ** | • When the precision of z value is 16-bit<br>8<br>• When the precision of z value is 8-bit<br>2 | |
| | **XGdcGeoShadowColor** | 2 | |
| | **XGdcGeoShadowBackColor** | 2 | |
| | **XGdcGeoBorderColor** | 2 | |
| | **XGdcGeoBorderBackColor** | 2 | |
| | **XGdcGeoSetupMode** | 0 | The display list is not generated so that only the value of the context may change |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Device Coordinate System | Fast line drawing<br>Fast polyline drawing | 5(Per 1 line) | |
| | Fast triangle drawing<br>Fast triangle fan drawing<br>Fast triangle strip drawing | 5(Per 1 triangle) | |
| | Polygon drawing | 5(Per 1 polygon) | |
| | 2D point drawing | 3*Number of vertices | |
| | 3D point drawing | 4*Number of vertices | |
| | 2D line drawing<br>2D polyline drawing | 6(Per 1 line) | |
| | 3D line drawing<br>3D polyline drawing | 10(Per 1 line) | |
| | 2D flat shading triangle drawing<br>2D flat shading triangle fan drawing<br>2D flat shading triangle strip drawing | 10(Per 1 triangle) | |
| | 2D Gouraud shading triangle drawing<br>2D Gouraud shading triangle fan drawing<br>2D Gouraud shading triangle strip drawing | 20(Per 1 triangle) | |
| | Flat shading triangle drawing with 2D texture<br>Flat shading triangle fan drawing with 2D texture<br>Flat shading triangle strip drawing with 2D texture | 30(Per 1 triangle) | |
| | Gouraud shading triangle drawing with 2D texture<br>Gouraud shading triangle fan drawing with 2D texture<br>Gouraud shading triangle strip drawing with 2D texture | 40(Per 1 triangle) | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Device Coordinate System | 2D line drawing<br>2D polyline drawing | 6(Per 1 line) | |
| | 3D line drawing<br>3D polyline drawing | 10(Per 1 line) | |
| | 2D flat shading triangle drawing<br>2D flat shading triangle fan drawing<br>2D flat shading triangle strip drawing | 10(Per 1 triangle) | |
| | 2D Gouraud shading triangle drawing<br>2D Gouraud shading triangle fan drawing<br>2D Gouraud shading triangle strip drawing | 20(Per 1 triangle) | |
| | Flat shading triangle drawing with 2D texture<br>Flat shading triangle fan drawing with 2D texture<br>Flat shading triangle strip drawing with 2D texture | 30(Per 1 triangle) | |
| | Gouraud shading triangle drawing with 2D texture<br>Gouraud shading triangle fan drawing with 2D texture<br>Gouraud shading triangle strip drawing with 2D texture | 40(Per 1 triangle) | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Device Coordinate System | 3D flat shading triangle drawing<br>3D flat shading triangle fan drawing<br>3D flat shading triangle strip drawing | 14(Per 1 triangle) | |
| | 3D Gouraud shading triangle drawing<br>3D Gouraud shading triangle fan drawing<br>3D Gouraud shading triangle strip drawing | 24(Per 1 triangle) | |
| | Flat shading triangle drawing with 3D texture<br>Flat shading triangle fan drawing with 3D texture<br>Flat shading triangle strip drawing with 3D texture | 34(Per 1 triangle) | |
| | Gouraud shading triangle drawing with 3D texture<br>Gouraud shading triangle fan drawing with 3D texture<br>Gouraud shading triangle strip drawing with 3D texture | 44(Per 1 triangle) | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Objects Coordinate System | **XGdcGeoTexCoord2D[N/Nf]** | 0 | The display list that shows the vertex color is generated in **XGdcGeoDrawVertex2D[f/i]** command or **XGdcGeoDrawVertex3D[f/i]** command |
| | **XGdcVertexColor[32/3f]** | 0 | The display list that shows the vertex color is generated in **XGdcGeoDrawVertex2D[f/i]** command or **XGdcGeoDrawVertex3D[f/i]** command |
| | 2D point drawing (GDC_SFLOAT type) | 4+3*Number of vertices | |
| | 2D point drawing (GDC_FIXED32 type) | 4+3*Number of vertices | |
| | 2D point drawing (GDC_LONG type) | 4+2*Number of vertices | |
| | 3D point drawing (GDC_SFLOAT type) | 4+4*Number of vertices | |
| | 3D point drawing (GDC_FIXED32 type) | 4+4*Number of vertices | |
| | 3D point drawing (GDC_LONG type) | 4+3*Number of vertices | |
| | 2D line drawing (GDC_SFLOAT type) | 4+3*Number of vertices | |
| | 2D line drawing (GDC_FIXED32 type) | 4+3*Number of vertices | |
| | 2D line drawing (GDC_LONG type) | 4+2*Number of vertices | |
| | 3D line drawing (GDC_SFLOAT type) | 4+4*Number of vertices | |
| | 3D line drawing (GDC_FIXED32 type) | 4+4*Number of vertices | |
| | 3D line drawing (GDC_LONG type) | 4+3*Number of vertices | |
| | 2D flat shading triangle drawing (GDC_SFLOAT type) | 4+3*Number of vertices | |
| | 2D flat shading triangle drawing (GDC_FIXED32 type) | 4+3*Number of vertices | |
| | 2D flat shading triangle drawing (GDC_LONG type) | 4+2*Number of vertices | |
| | 3D flat shading triangle drawing (GDC_SFLOAT type) | 4+4*Number of vertices | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Objects Coordinate System | 3D flat shading triangle drawing (GDC_FIXED32 type) | 4+4*Number of vertices | |
| | 3D flat shading triangle drawing (GDC_LONG type) | 4+3*Number of vertices | |
| | 2D Gouraud shading triangle drawing (GDC_SFLOAT type) | 4+6*Number of vertices | |
| | 2D Gouraud shading triangle drawing (GDC_FIXED32 type) | 4+4*Number of vertices | |
| | 2D Gouraud shading triangle drawing (GDC_LONG type) | 4+3*Number of vertices | |
| | 3D Gouraud shading triangle drawing (GDC_SFLOAT type) | 4+7*Number of vertices | |
| | 3D Gouraud shading triangle drawing (GDC_FIXED32 type) | 4+5*Number of vertices | |
| | 3D Gouraud shading triangle drawing (GDC_LONG type) | 4+4*Number of vertices | |
| | Flat shading triangle drawing with 2D texture (GDC_SFLOAT type) | 4+5*Number of vertices | |
| | Flat shading triangle drawing with 2D texture (GDC_FIXED32 type) | 4+5*Number of vertices | |
| | Flat shading triangle drawing with 2D texture (GDC_LONG type) | 4+4*Number of vertices | |
| | Flat shading triangle drawing with 3D texture (GDC_SFLOAT type) | 4+6*Number of vertices | |
| | Flat shading triangle drawing with 3D texture (GDC_FIXED32 type) | 4+6*Number of vertices | |
| | Flat shading triangle drawing with 3D texture (GDC_LONG type) | 4+5*Number of vertices | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Primitive Drawing Commands for Objects Coordinate System | Gouraud shading triangle drawing with 2D texture (GDC_SFLOAT type) | 4+8*Number of vertices | |
| | Gouraud shading triangle drawing with 2D texture (GDC_FIXED32 type) | 4+6*Number of vertices | |
| | Gouraud shading triangle drawing with 2D texture (GDC_LONG type) | 4+5*Number of vertices | |
| | Gouraud shading triangle drawing with 3D texture (GDC_SFLOAT type) | 4+9*Number of vertices | |
| | Gouraud shading triangle drawing with 3D texture (GDC_FIXED32 type) | 4+7*Number of vertices | |
| | Gouraud shading triangle drawing with 3D texture (GDC_LONG type) | 4+6*Number of vertices | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Texture Image Management Control Commands | **XGdcTextureMemoryMode** | 2 | |
| | **XGdcTextureLoadInt8** | 3+(number of pixels for width*height)/4 | |
| | **XGdcTextureLoadInt16** | 3+(number of pixels for width*height)/2 | |
| | **XGdcTextureLoadExt8** | • When the texture data is less than 65536 words<br>21+(number of pixels for width*height + 3)/4<br>• When the texture data is more than 65536 words<br>22+(number of pixels for width*height + 3)/4 | |
| | **XGdcTextureLoadExt16** | • When the texture data is less than 65536 words<br>21+(number of pixels for width*height + 1)/2<br>• When the texture data is more than 65536 words<br>22+(number of pixels for width*height + 1)/2 | |
| | **XGdcTextureLoadExt16Fast** | 20+(number of pixels for width*height*2) | |
| | **XGdcTextureDimension** | 7 | |
| | **XGdcBltTexture** | 6 | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Binary Pattern Drawing Commands | **XGdcBitPatternDraw** | • MB86290A<br>3+(number of pixels for width/8*height)/4<br>• MB86291 or later<br>5+(number of pixels for width/8*height)/4 | |
| | **XGdcBitPatternDrawByte** | • MB86290A<br>3+(number of pixels for width/8*height)/4<br>• MB86291 or later<br>5+(number of pixels for width/8*height)/4 | It differs according to coordinates at the drawing destination |
| | **XGdcBitPatternMode** | 2 | |
| BLT Commnads | **XGdcBltCopy** | 4 | |
| | **XGdcBltCopyAlt** | 8 | |
| | **XGdcBltCopyAltSync** | 9 | |
| | **XGdcBltDraw8** | • When the bitmap data are less than 65536 words<br>7+(number of pixels for width*height+3)/4<br>• When the bitmap data is more than 65536 words<br>8+(number of pixels for width*height+3)/4 | |
| | **XGdcBltDraw16** | • When the bitmap data are less than 65536 words<br>7+(number of pixels for width*height+1)/2<br>• When the bitmap data is more than 65536 words<br>8+(number of pixels for width*height+1)/2 | |
| | **XGdcBltFill** | 3 | |
| | **XGdcBltColorTransparent** | 2 | |
| | **XGdcBltCopyAltAlpha** | 8 | |

(Continue)

**Table 9.1 Display List Size List** (Continued)

| Type | Driver Command or Drawing Process | Display List Size [Unit: word] | Supplementation |
|---|---|---|---|
| Execution Control Commands | **XGdcVerticalSync** | 1 | |
| | **XGdcInterrupt** | 1 | |
| Drawing Attribute Restore Command | **XGdcRestoreAttr** | • When **GDC_RESTORE_COMMON** is specified<br>31<br>• When **GDC_RESTORE_GEOMETRY** is specified<br>36<br>• When **GDC_RESTORE_EXTEND** is specified<br>40<br>• When **GDC_RESTORE_ALL** is specified<br>107 | |