

Contributors :

- Fafling - Fixing
- Benjamin Siskoo - Updated docs.

Corrections were applied to the following pages by the KTF tag, differences are marked in red.

Page (1) = Page 7 PDF	Page 120 = Page 136 PDF
Page (2) = Page 8 PDF	Page 121 = Page 137 PDF
Page (3) = Page 9 PDF	Page 123 = Page 139 PDF
Page 2 = Page 18 PDF	Page 125 = Page 141 PDF
Page 6 = Page 22 PDF	Page 126 = Page 142 PDF
Page 7 = Page 23 PDF	Page 127 = Page 143 PDF
Page 8 = Page 24 PDF	Page 128 = Page 144 PDF
Page 12 = Page 28 PDF	Page 129 = Page 145 PDF
Page 13 = Page 29 PDF	Page 130 = Page 146 PDF
Page 14 = Page 30 PDF	Page 134 = Page 150 PDF
Page 20 = Page 36 PDF	Page 142 = Page 158 PDF
Page 22 = Page 38 PDF	Page 147 = Page 163 PDF
Page 23 = Page 39 PDF	Page 148 = Page 164 PDF
Page 25 = Page 41 PDF	Page 150 = Page 166 PDF
Page 33 = Page 49 PDF	Page 153 = Page 169 PDF
Page 37 = Page 53 PDF	Page 158 = Page 174 PDF
Page 38 = Page 54 PDF	Page 159 = Page 175 PDF
Page 39 = Page 55 PDF	Page 164 - Page 180 PDF
Page 40 = Page 56 PDF	Page 165 - Page 181 PDF
Page 41 = Page 57 PDF	Page 166 - Page 182 PDF
Page 43 = Page 59 PDF	
Page 46 = Page 62 PDF	
Page 47 = Page 63 PDF	
Page 49 = Page 65 PDF	
Page 50 = Page 66 PDF	
Page 62 = Page 78 PDF	
Page 66 = Page 82 PDF	
Page 72 = Page 88 PDF	
Page 79 = Page 95 PDF	
Page 80 = Page 96 PDF	
Page 84 = Page 100 PDF	
Page 87 = Page 103 PDF	
Page 88 = Page 104 PDF	
Page 90 = Page 106 PDF	
Page 91 = Page 107 PDF	
Page 93 = Page 109 PDF	
Page 94 = Page 110 PDF	
Page 95 = Page 111 PDF	
Page 96 = Page 112 PDF	
Page 97 = Page 113 PDF	
Page 104 = Page 120 PDF	
Page 105 = Page 121 PDF	
Page 119 = Page 135 PDF	

## **General Notice**

When using this document, keep the following in mind:

1. This document is confidential. By accepting this document you acknowledge that you are bound by the terms set forth in the non-disclosure and confidentiality agreement signed separately and /in the possession of SEGA. If you have not signed such a non-disclosure agreement, please contact SEGA immediately and return this document to SEGA.
2. This document may include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new versions of the document. SEGA may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time.
3. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without SEGA'S written permission. Request for copies of this document and for technical information about SEGA products must be made to your authorized SEGA Technical Services representative.
4. No license is granted by implication or otherwise under any patents, copyrights, trademarks, or other intellectual property rights of SEGA Enterprises, Ltd., SEGA of America, Inc., or any third party.
5. Software, circuitry, and other examples described herein are meant merely to indicate the characteristics and performance of SEGA's products. SEGA assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples describe herein.
6. It is possible that this document may contain reference to, or information about, SEGA products (development hardware/software) or services that are not provided in countries other than Japan. Such references/information must not be construed to mean that SEGA intends to provide such SEGA products or services in countries other than Japan. Any reference of a SEGA licensed product/program in this document is not intended to state or simply that you can use only SEGA's licensed products/ programs. Any functionally equivalent hardware/software can be used instead.
7. SEGA will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's equipment, or programs according to this document.

NOTE: A reader's comment/correction form is provided with this document. Please address comments to :

SEGA of America, Inc., Developer Technical Support (att. Evelyn Merritt)  
150 Shoreline Drive, Redwood City, CA 94065

SEGA may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.



SEGA OF AMERICA, INC.  
Consumer Products Division

# VDP1 User's Manual

Doc. # ST-013-R3-061694

# **READER CORRECTION/COMMENT SHEET**

**Keep us updated!**

If you should come across any incorrect or outdated information while reading through the attached document, or come up with any questions or comments, please let us know so that we can make the required changes in subsequent revisions. Simply fill out all information below and return this form to the Developer Technical Support Manager at the address below. Please make more copies of this form if more space is needed. Thank you.

## **General Information:**

**Your Name** \_\_\_\_\_ **Phone** \_\_\_\_\_

## **Phone**

**Document number** ST-013-R3-061694 **Date** \_\_\_\_\_

Date \_\_\_\_\_

**Document name** VDP1 User's Manual

## **Corrections:**

## **Questions/comments:**

---

---

---

---

---

## Where to send your corrections:

Fax: (415) 802-1717  
Attn: Evelyn Merritt,  
Developer Technical Support

Mail: SEGA OF AMERICA  
Attn: Evelyn Merritt,  
Developer Technical Support  
150 Shoreline Dr.  
Redwood City, CA 94065

## **REFERENCES**

In translating/creating this document, certain technical words and/or phrases were interpreted with the assistance of the technical literature listed below.

1. *Dictionary of Science and Engineering, 350,000 words, 3rd Edition*  
Inter Press  
Tokyo, Japan  
1990
2. *Computer Dictionary*  
Kyoritsu Publishing Co., LTD.  
Tokyo, Japan  
1978
3. *IBM Dictionary of Computing*  
McGraw-Hill, Inc.  
New York, New York  
1994
4. SEGA SATURN TECHNICAL BULLETIN #15
5. SEGA SATURN TECHNICAL BULLETIN #22
6. SEGA SATURN TECHNICAL BULLETIN #23
7. SEGA SATURN TECHNICAL BULLETIN #25
8. SEGA SATURN TECHNICAL BULLETIN #SOA-2

## **Revision History**

Revision 7.1

December 6, 1993

- Minor corrections
- Total 155 pages

First Edition

February 20, 1994

- Added even/odd coordinate selection bit (EOS) to frame buffer change mode register (FBCR)
- Added high speed shrink (HSS) and pre-clipping disable (Pclp) to plot mode word (+04H) in the command table.
- Total 163 pages

# Introduction

This manual explains the functions of the VDP1 and how they are used. The VDP1 primarily defines draw data and performs drawing.

## Definitions

The terms used in this manual are defined as follows:

### VDP1

VDP1 is an integrated circuit (IC) that controls drawing. The VRAM and frame buffer (both DRAM) are connected to the VDP1. The CPU writes draw commands to VRAM via the VDP1. **The VDP1 reads draw commands from VRAM and writes (draws) draw data to the frame buffer, and later transfers this data to VDP2 for mixing and output to a display monitor.**

The frame buffer is a bit-mapped memory, and any data written to it has a 1-to-1 correspondence with the contents of the display screen. Therefore, the frame buffer is a map of the display screen. There are system registers in the VDP1 and screen display is controlled by setting values in these registers.

### Display

Televisions are commonly used as display devices. The standard used in Japan and in the United States is NTSC. PAL system TVs are used in Europe.

Hi-res is a high resolution mode that doubles the horizontal resolution. 31KC is a system that doubles the horizontal frequency of 15KC in order to raise the vertical resolution. Hi-vision (HDTV), or so-called "high-definition" TV, has twice the horizontal and vertical resolution of regular TV.

A TV display is comprised of fields and frames. A field is the time it takes a scanning line to scan one screen. In the NTSC system, this is 1/60th of a second and in the PAL system, 1/50th of a second. A frame is the time it takes from one change of the frame buffer to the next change. When the change mode of the frame buffer is in a one cycle mode, one frame is equal to one field. When the frame buffer is changed every two fields such as in single density interlace, one frame comprises two fields. When the frame buffer is changed only once a second in the manual mode, one frame comprises 60 fields in the NTSC system, and 50 fields in the PAL system.

In the rest of the document, for a PAL system, read "1/50th of a second" instead of "1/60th of a second", and "1/25th of a second" instead of "1/30th of a second".

Before each frame is drawn, the frame buffer can be automatically erased. Erase is performed by writing (filling) a code specified in a register to the frame buffer. This operation is referred to as erase / write.

## Access

Reading and writing of data by the CPU and VDP1 in VRAM, the system registers, and frame buffers are referred to as access. When two or more devices attempt access at the same time, the accesses are arbitrated, with writes normally given priority over reads.

When a DMA controller is used, access of VRAM can be performed continuously and rapidly by burst transfer. However, do not perform burst access of registers.

## Data

A bit is the smallest unit of data which is represented by "0" or "1." A combination of 8 bits is called a byte, and 16 bits (or 2 bytes) is called a word. When a byte is divided into its upper 4 bits and its lower 4 bits, each is referred to as a nibble.

## Commands

Commands include draw commands, clipping coordinate set commands, and local coordinate set commands. Clipping is the removal of graphics outside a set area. Local coordinates change the drawing position using the coordinates specified by the draw command as local coordinates.

Commands are defined in VRAM as command tables. In addition to the command table, there are the character pattern table, the Gouraud shading table, and the color lookup table.

Tables are defined by addresses with an 8H-byte boundary or a 20H-byte boundary, and are stored in VRAM. A boundary is the splitting of addresses at the 8H or 20H boundary so there is no remainder. **Registers have 2H-byte (one word) boundaries.**

The VDP1 fetches command tables successively and draws according to these commands. Fetch is the act of reading from VRAM in order to process commands.

## Parts

Objects drawn by using the draw command are called parts. Parts are divided into textured and non-textured parts. Textured parts are called sprites; non-textured parts include polygons, polylines, and lines.

Sprites are characters and the color codes of these characters are defined in VRAM as a character pattern table. Polygons are filled quadrangles, polylines are non-filled quadrangles, and lines are straight lines that connect two points.

The VDP1 processes draw commands in VRAM and draws parts in the frame buffer 1 pixel at a time. The bit depth of the data in the frame buffer is 8 or 16 bits per pixel (bpp).

## Color

The RGB code method and the color bank method, which uses palette codes and color banks are used to express color. The RGB code method specifies the luminance of each of the colors: red (R), green (G), and blue (B), in 5 bits. In the color bank method, the VDP2 color palette is selected from a color bank, and a palette code is used to select the desired color from this palette.

Sprite colors can be specified using the color lookup table method. The color is selected from among the 16 colors defined in the color lookup table. The color code can be specified using RGB or color bank codes.

## Color Calculation

Color calculation is a special function of the VDP1. Gouraud shading is an example of color calculation. The luminance of the RGB code of a particular part can be changed using the RGB codes of four points defined in the Gouraud shading table.

## Notations Used in this Manual

The notations used in this manual are as follows:

### Binary and Hexadecimal Numbers

Binary numbers are designated with a "B" at the end of the number—for example, 100B.  
Hexadecimal numbers are designated with an "H" at the end of the number—for example,  
00H, FFH.

### Units

1 Kbyte is equal to 1024 bytes. 1 Mbit is 1,048,576 bits.

### MSB, LSB

In the configuration of bytes and words, the left is the most significant bit (MSB) and the right is the least significant bit (LSB).

### Undefined Bits

Bits that are not defined in the command table or the system registers are indicated by a hyphen "-". Write 0B to undefined bits in the system registers. Write "0" or "1" to undefined bits in command tables. If an entire word is unused, fill with either "0" or "1". When a word is partially used, fill the remaining unused bits with "0".

### Addresses

The addresses shown in this manual are relative VDP1 addresses. VDP1 is at the absolute address 5C00000H of the system.

Add 5C00000H to a relative address to determine the absolute address. For example, the absolute address 000000H in VRAM is address 5C00000H, and the absolute address 180000H in the system registers becomes 5D80000H.

# Table of Contents

Introduction	
Definitions .....	(1)
Notations Used in this Manual .....	(4)
Contents .....	(5)
List of Figures .....	(8)
List of Tables.....	(10)
Chapter 1    Functions of the VDP1 .....	1
1.1     VDP1 .....	2
System Configuration .....	2
Functions of the VDP1 .....	3
Parts .....	4
Textured Parts .....	5
Non-Textured Parts .....	10
Color .....	12
1.2     Screen Modes.....	14
Screen Modes and Display Areas .....	14
Rotated Reading of Frame Buffer .....	15
Chapter 2    Address Map .....	17
2.1     Address Map .....	18
VRAM .....	19
Frame Buffer .....	20
System Registers .....	23
2.2     Tables in VRAM .....	24
Command Table .....	25
Color Lookup Table .....	26
Gouraud Shading Table .....	26
Character Pattern Table .....	26
Chapter 3    Processing Flow .....	27
3.1     Draw Procedure Flow .....	28
3.2     Command Table Flow .....	30
3.3     Table Referencing .....	31

Chapter 4	System Registers .....	33
4.1	TV Mode Selection Register .....	36
4.2	Frame Buffer Change Mode Register .....	38
4.3	Plot Trigger Register .....	45
4.4	Erase/Write .....	46
4.4.1	Erase/Write Data Register .....	46
4.4.2	Erase/Write Upper-Left Coordinate Register .....	47
4.4.3	Erase/Write Lower-Right Coordinate Register .....	47
4.5	Draw Forced Termination Register .....	51
4.6	Transfer End Status Register .....	52
4.7	Last Operation Command Address Register .....	54
4.8	Current Operation Command Address Register .....	55
4.9	Mode Status Register .....	57
Chapter 5	Tables .....	59
5.1	Character Pattern Tables .....	60
5.2	Color Lookup Tables .....	62
5.3	Gouraud Shading Table .....	64
5.4	Command Tables .....	66
Chapter 6	Command Tables .....	69
6.1	CMDCTRL (Control Words) .....	70
6.1.1	Commands .....	71
6.1.2	Jump Mode .....	72
6.1.3	Zoom Point .....	73
6.1.4	Character Read Direction .....	77
6.2	CMDLINK (Link Specification) .....	78
6.3	CMDPMOD (Draw Mode Word) .....	79
6.3.1	High Speed Shrink .....	81
6.3.2	Pre-Clipping Disable .....	83
6.3.3	User Clipping Enable .....	84
6.3.4	User Clipping Mode .....	84
6.3.5	Mesh Enable .....	85
6.3.6	End Code Disable .....	86
6.3.7	Transparent Pixel Disable .....	88
6.3.8	Color Mode .....	89
6.3.9	Color Calculation .....	93
6.3.10	MSB ON .....	97

6.4	CMDCOLR (Color Control Word) .....	98
	Color Bank .....	99
	Color Lookup Table .....	101
	Non-Textured Color .....	102
6.5	CMDSRCA (Character Address) .....	103
6.6	CMDSIZE (Character Size) .....	104
6.7	CMDXA~CMDYD (Vertex Coordinate Data) .....	105
6.8	CMDGRDA (Gouraud Shading Table) .....	106
<b>Chapter 7</b>	<b>Commands .....</b>	<b>107</b>
7.1	System Clipping Coordinate Set Command .....	110
	System Clipping .....	111
7.2	User Clipping Coordinate Set Command .....	112
	User Clipping .....	113
7.3	Local Coordinate Set Command .....	116
	Local Coordinates .....	117
7.4	Normal Sprite Draw Command .....	118
7.5	Scaled Sprite Draw Command.....	120
	Specification of Two Coordinate Vertices .....	120
	Specification of Fixed Point (Scaled Sprite Draw Command) .....	122
7.6	Distorted Sprite Draw Command .....	124
7.7	Polygon Draw Command .....	126
7.8	Polyline Draw Command .....	128
7.9	Line Draw Command .....	130
7.10	Draw End Command .....	132
<b>Chapter 8</b>	<b>Quick Reference .....</b>	<b>133</b>
8.1	System Registers.....	134
8.2	Tables .....	140
8.3	Command Tables .....	142
8.4	Commands .....	151
<b>Chapter 9</b>	<b>Precautions for Use .....</b>	<b>157</b>
<b>Index</b>	<b>.....</b>	<b>161</b>

# List of Figures

## (Chapter 1 Functions of the VDP1)

Figure 1.1 System Configuration .....	2
Figure 1.2 Normal Sprites .....	5
Figure 1.3 Scaled Sprites (Two Vertices Specified) .....	6
Figure 1.4 Scaled Sprites (Zoom Point Specified) .....	7
Figure 1.5 Distorted Sprites .....	8
Figure 1.6 Fill-In Processing .....	9
Figure 1.7 Polygons.....	10
Figure 1.8 Polylines .....	11
Figure 1.9 Line .....	11
Figure 1.10 Bit Configuration of Color Bank Method .....	12

## (Chapter 2 Address Map)

Figure 2.1 Address Map .....	18
Figure 2.2 Frame Buffer Plane .....	21

## (Chapter 3 Processing Flow)

Figure 3.1 Command Table Flow .....	30
Figure 3.2 Referencing of Tables .....	31

## (Chapter 4 System Registers)

Figure 4.1 Single Interlace and Double Interlace Display .....	43
Figure 4.2 Erase/Write Area .....	48
Figure 4.3 Last Operation Command and Current Operation Command Address .....	54

## (Chapter 5 Tables)

Figure 5.1 Examples of Character Pattern Tables .....	61
Figure 5.2 Color Lookup Table .....	62
Figure 5.3 Relationship between Tables in Lookup Table System .....	63
Figure 5.4 RGB Code Format.....	64
Figure 5.5 Command Table .....	66

## (Chapter 6 Command Tables)

Figure 6.1 Zoom Point .....	73
Figure 6.2 Drawing Area .....	75
Figure 6.3 Zoom Point and Drawing Area .....	76
Figure 6.4 Character Read Direction .....	77
Figure 6.5 High Speed Shrink .....	82
Figure 6.6 Pre-Clipping .....	83
Figure 6.7 Drawing Area .....	84
Figure 6.8 Mesh Processing .....	85
Figure 6.9 Mesh Processing of Lines and Polylines .....	85
Figure 6.10 (a) End Code Processing (1 of 2) .....	87
Figure 6.10 (b) End Code Processing (2 of 2) .....	87
Figure 6.11 Example of Drawing in Modes 0 and 1 .....	90
Figure 6.12 Example of Drawing in Modes 2, 3, and 4 .....	91
Figure 6.13 RGB Code Format .....	92
Figure 6.14 Example of Drawing in Mode 5 .....	92
Figure 6.15 Examples of Color Calculation .....	96
Figure 6.16 MSB ON .....	97
Figure 6.17 Color Bank .....	99
Figure 6.18 Color Lookup Table .....	101
Figure 6.19 CMDSIZE .....	104

## (Chapter 7 Commands)

Figure 7.1 System Clipping .....	111
Figure 7.2 User Clipping Settings .....	114
Figure 7.3 User Clipping .....	115

## (Chapter 8 Quick Reference)

Figure 8.1 Examples of Character Pattern Tables .....	140
Figure 8.2 Color Lookup Table .....	140
Figure 8.3 Command Table .....	142

# List of Tables

## (Chapter 1 Functions of the VDP1)

Table 1.1 Classification of Parts .....	4
Table 1.2 Screen Modes and Display Areas .....	14

## (Chapter 2 Address Map)

Table 2.1 System Registers .....	23
Table 2.2 Tables in VRAM .....	24

## (Chapter 4 System Registers)

Table 4.1 System Registers .....	34
Table 4.2 Screen Modes .....	37
Table 4.3 (a) Example of Use of Frame Buffer Change Mode (Fixed at VBE = 0) .....	41
Table 4.3 (b) Example of Use of Frame Buffer Change Mode (VBE Is Used) .....	42
Table 4.4 Number of Rasters and Number of Pixels .....	49
Table 4.5 Number of Pixels that Can Be Used in V-Blank Erase (in Non-Interlace Display) ....	50

## (Chapter 5 Tables)

Table 5.1 Size of Character Pattern Tables .....	60
Table 5.2 Gouraud Shading Table .....	64
Table 5.3 Relationship between Gouraud Shading Table Settings and Correction Values ...	65

## (Chapter 6 Command Tables)

Table 6.1 Commands .....	71
Table 6.2 Pixel Data .....	92
Table 6.3 CMDCOLR .....	98
Table 6.4 Example of Relationship of Defined Data and Draw Data to Color Bank .....	100
Table 6.5 Relationships between Settings and Drawn Pixels .....	104
Table 6.6 Correspondence between Commands and CMDXA~CMDYD .....	105

## (Chapter 7 Commands)

Table 7.1 Commands .....	109
--------------------------	-----

## (Chapter 8 Quick Reference)

Table 8.1 Screen Modes .....	134
Table 8.2 Gouraud Shading Table .....	141
Table 8.3 Commands .....	145
Table 8.4 CMDCOLR .....	149
Table 8.5 Relationships between Settings and Drawn Pixels .....	149
Table 8.6 Correspondence between Commands and CMDXA~CMDYD .....	150

# Chapter 1

## Functions of the VDP1

### Contents

1.1	VDP1 .....	2
	System Configuration .....	2
	Functions of the VDP1 .....	3
	Parts .....	4
	Textured Parts .....	5
	Non-textured Parts .....	10
	Color .....	12
1.2	Screen Modes.....	14
	Screen Modes and Display Areas .....	14
	Rotated Reading of Frame Buffer .....	15

## 1.1 VDP1

The VDP1 is a sprite drawing IC for the SEGA SATURN. Because the VDP1 uses a frame buffer, is much faster, and is characterized by an increased RAM capacity, it can display many more sprites (characters) compared to previous systems.

### System Configuration

A VRAM (4-Mbit DRAM) and a two-plane frame buffer (2-Mbit DRAM per screen) are connected to the VDP1. The image data defined in VRAM by the CPU are output to the display device via the frame buffer.

Draw data is sent from the CPU to the VDP1 via the system control IC and is written in VRAM. Parts written in VRAM are drawn in the frame buffer in a 16- or 8-bit/pixel form. The frame buffer data that is drawn is displayed on the display device via the priority circuit in the VDP2. **The priority circuit prioritizes the scroll plane and the sprite plane.** The frame buffer has two screens, and draw and display are changed every frame.

The information that controls draws is set in the VDP1 system registers by the CPU via the system controller IC. The system registers control draws.

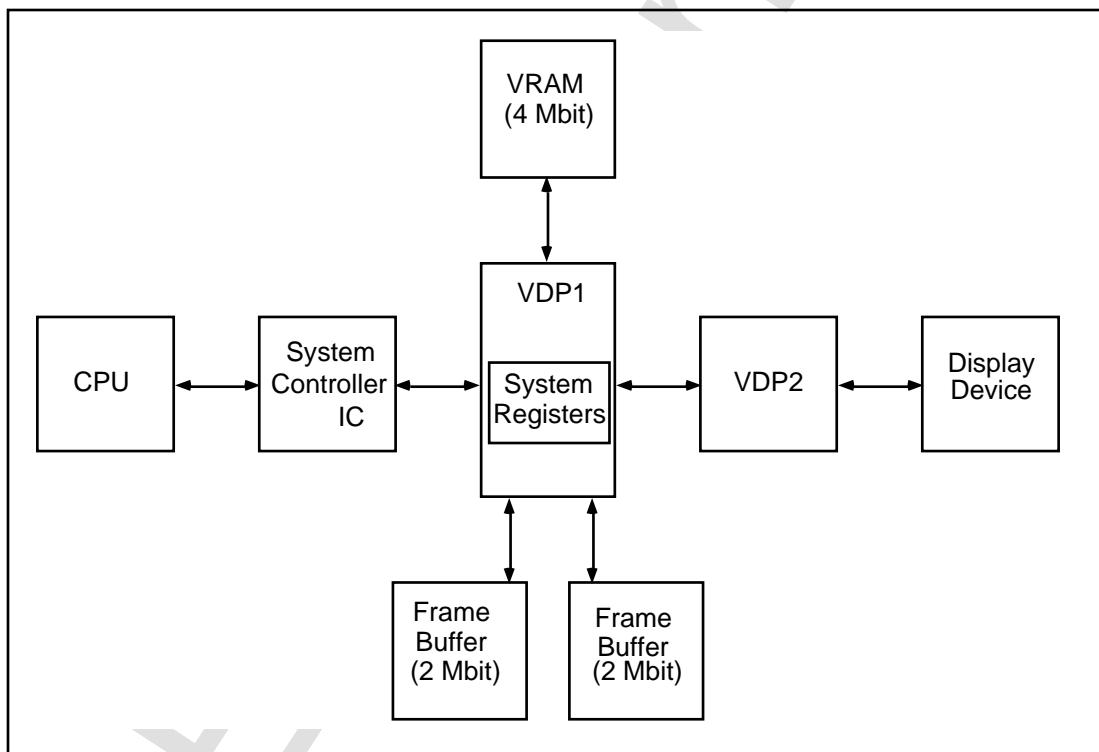


Figure 1.1 System Configuration



## **Functions of the VDP1**

The functions of the VDP1 include the drawing of parts (characters and lines), specification of colors, color calculation of Gouraud shading, specification of clipping coordinates and local coordinates, and control of display by the frame buffer. Parts, color, and coordinates are controlled by the command table in VRAM, and display of the frame buffer is controlled by the system registers.

### **Parts**

The following graphics can be drawn as parts.

- Normal sprites
- Scaled sprites (with zooming)
- Distorted sprites (includes rotation)
- Polygons (quadrangles)
- Polylines (quadrangles comprising four lines)
- Lines

Collectively, sprites are referred to as textured parts, and polygons, polylines, and lines are referred to as non-textured parts.

### **Color**

- The possible numbers of colors for each textured part are 16, 64, 128, 256, and 32,768.

### **Special Functions**

- When RGB codes are used, color calculation of half-luminance, half-transparency, Gouraud shading, and shadowing are possible.
- Mesh (tiling).

### **Coordinate Control**

- System and user clipping settings are possible.
- Local coordinates can be set during drawing.

### **Frame Buffer Display Control**

- Enlargement, reduction, and rotation of the entire frame buffer screen.
- Specification of the TV display mode.
- Specification of the frame buffer change mode, the change trigger, and the plot trigger.
- Specification of double interlace.
- Specification of fill area and fill data during erase/write.
- End status of transfer to the frame buffer as help information during program development.

## Parts

Parts are classified as shown in Table 1.1.

**Table 1.1 Classification of Parts**

Classification	Part name	Function	How defined
Parts	Textured parts	Normal sprites	Character, vertical, and horizontal inversion
		Scaled sprites	Character, vertical, and horizontal inversion, enlargement and reduction, stretching
		Distorted sprites	Character, vertical, and horizontal inversion, enlargement and reduction, stretching, rotation, twisting
Non-textured parts	Polygons	Filled quadrangles	4 vertices
	Polylines	Quadrangles	4 vertices
	Lines	Straight lines	Starting vertex and ending vertex



## Textured Parts

Textured parts are called sprites. Sprites draw character patterns. Character patterns define pixel data as character pattern tables in VRAM. The size of the pixel data is determined by the color mode and the size of the characters.

Sprites include normal sprites, scaled sprites, and distorted sprites. Normal sprites can be inverted vertically and horizontally; scaled sprites can be inverted vertically and horizontally, enlarged and reduced, and stretched; and distorted sprites can be inverted vertically and horizontally, enlarged and reduced, stretched, rotated, and twisted.

### Normal Sprites

The character pattern is drawn at a specified position. The coordinates of the upper-left vertex at which the character pattern is drawn are specified. The character pattern is drawn from the specified coordinates to the right in the X direction and down in the Y direction. When vertical and horizontal inversion are specified for the read-out direction of the character pattern, the right side of the defined character pattern is drawn from the left and the bottom side is drawn from the top.

Normal sprites cannot be rotated 90°. Specify the distorted sprite draw command to perform 90° rotation.

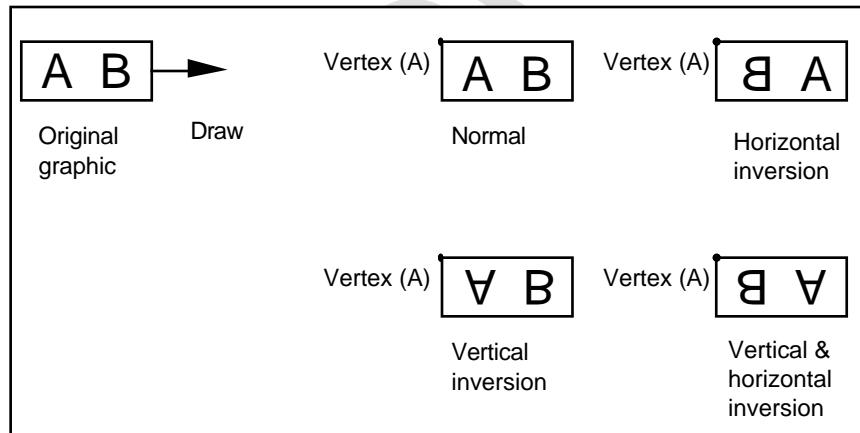


Figure 1.2 Normal Sprites

## Scaled Sprites

The character pattern is drawn enlarged or reduced at a specified position. There are two methods by which the position and scaling can be specified. The coordinates of the upper-left vertex and the lower-right vertex are specified in one method, while the zoom point coordinates and the horizontal and vertical display widths are specified in another.

Scaled sprites cannot be rotated 90°. Specify the distorted sprite draw command to perform 90° rotation.

### Specification of Coordinates of Two Vertices

In this method, the coordinates of the upper-left and the lower-right vertices are specified. The draw direction (direction of drawing a character) is determined by the positional relationship of the lower-right coordinate with respect to the upper-left coordinate. The character pattern is drawn from the upper-left coordinate toward the lower-right coordinate. When the value of X of the lower-right coordinate is smaller than the X value of the upper-left coordinate, the character pattern is inverted horizontally, and when the Y value of the lower-right coordinate is smaller than the Y value of the upper-left coordinate, the character pattern is inverted vertically.

As with normal sprites, inversion can be specified by the read direction. However, when the coordinates and the direction of readout are both inverted vertically, the inversions cancel each other out and the character remains unchanged.

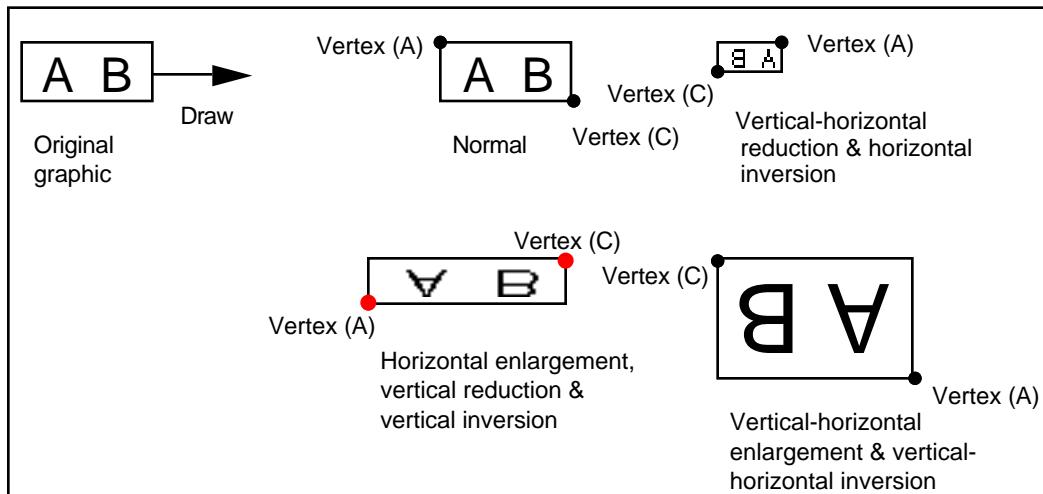
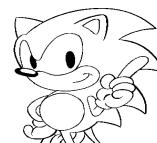


Figure 1.3 Scaled Sprites (Two Vertices Specified)



### Specification of Zoom Point

Scaled sprites can be drawn by specifying the zoom point and display width. The zoom point of the character pattern, the draw coordinates of the zoom point, and the display width at which the character pattern is drawn are specified. The zoom point specifies which point on the character pattern is used as a stationary point for drawing. The point is selected from the left side, center, and right side in the horizontal direction and from the top side, center, and bottom side in the vertical direction. The display width specifies the display width in the X direction and the Y direction. When vertical and horizontal inversions are specified for the character pattern readout direction, each character pattern is drawn inverted vertically and horizontally with reference to each zoom point. The draw area differs depending on vertical or horizontal inversion.

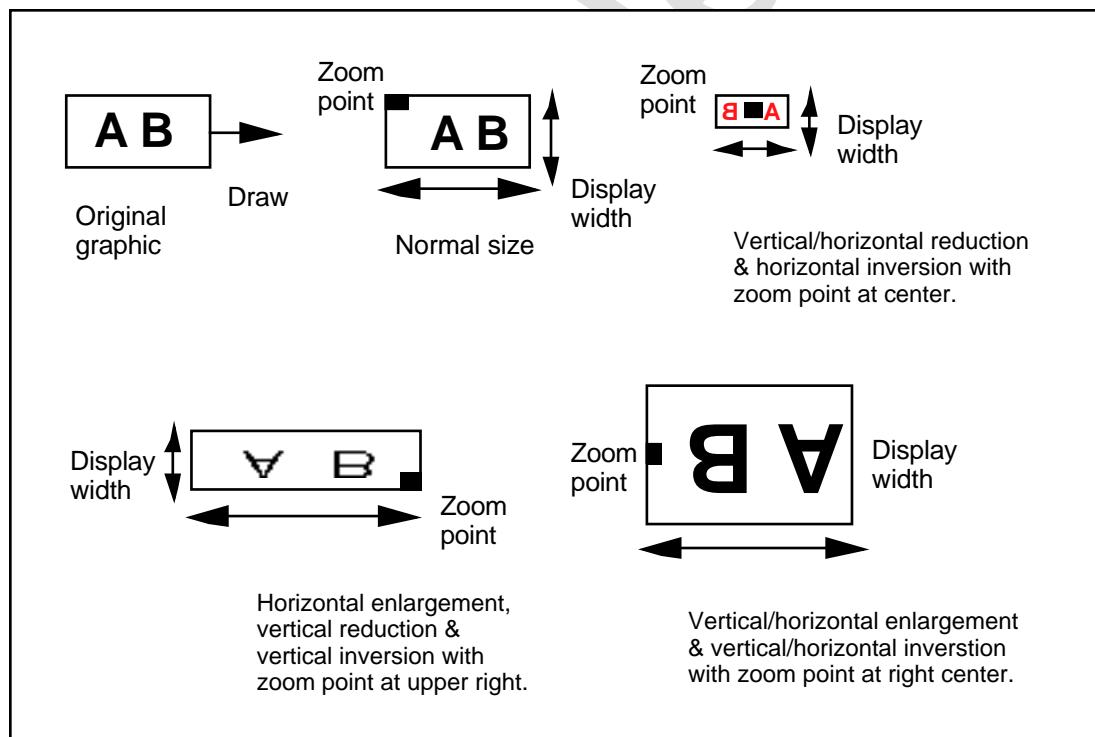


Figure 1.4 Scaled Sprites (Zoom Point Specified)

### Distorted Sprites

Draw a character pattern by specifying four coordinates. Specify the four draw coordinates corresponding to the four vertices of the character pattern. These four coordinates can assume any position relative to each other, and therefore the character pattern can be inverted, enlarged or reduced, rotated, or twisted, depending on how they are specified.

Distorted sprites are created by skewing the character pattern. At this time holes are filled to prevent the dropout of pixels. For this reason, there may be some pixels that are written twice, and therefore the results of half-transparent processing as well as other processing in color calculation cannot be guaranteed.

Because drawing is done with lines, part of the character pattern may result outside the graphic formed by linking the four vertices if it is twisted or concave.

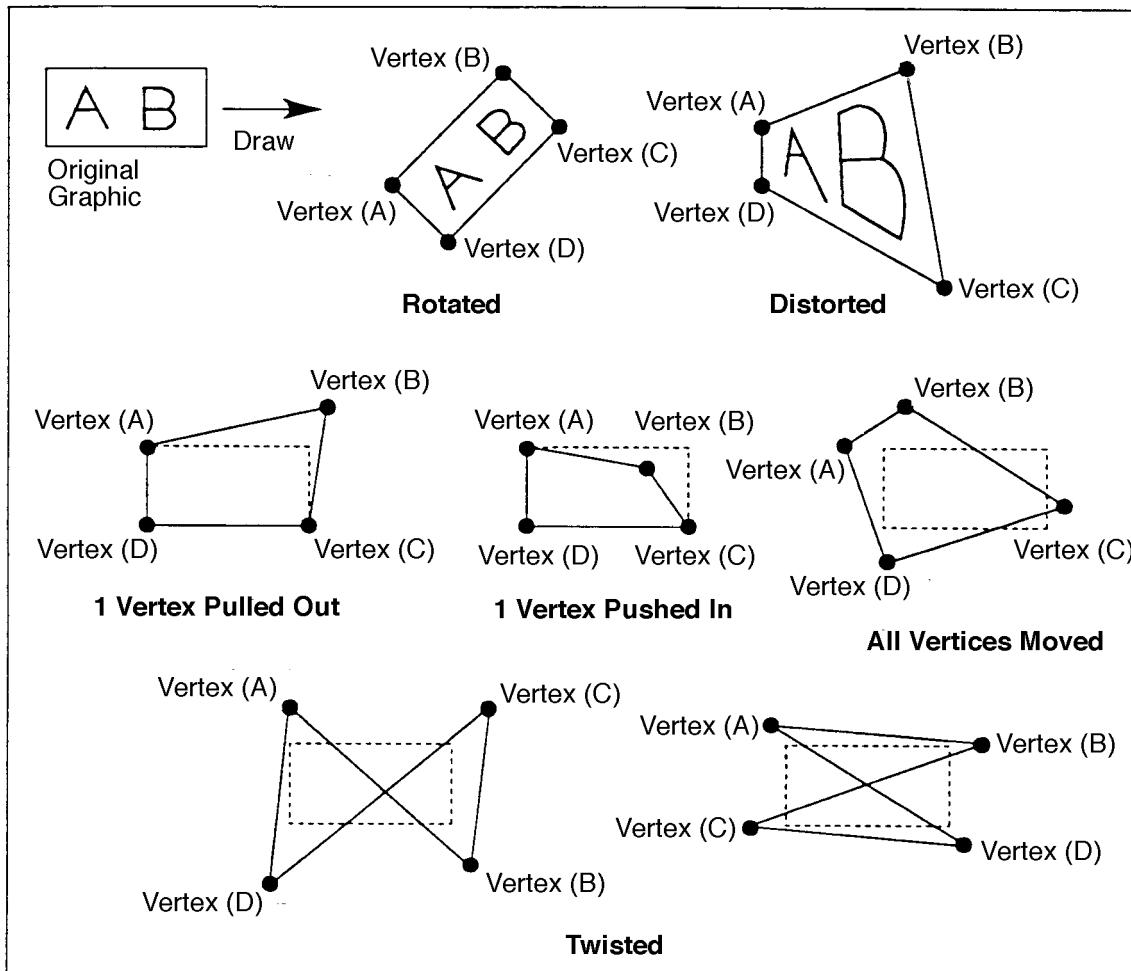
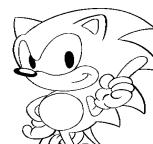


Figure 1.5 Distorted Sprites



## Anti-aliasing

Distorted sprites and polygons contain diagonal lines that may result in pixel drop-out (aliasing). When this happens, holes are anti-aliased as shown below.

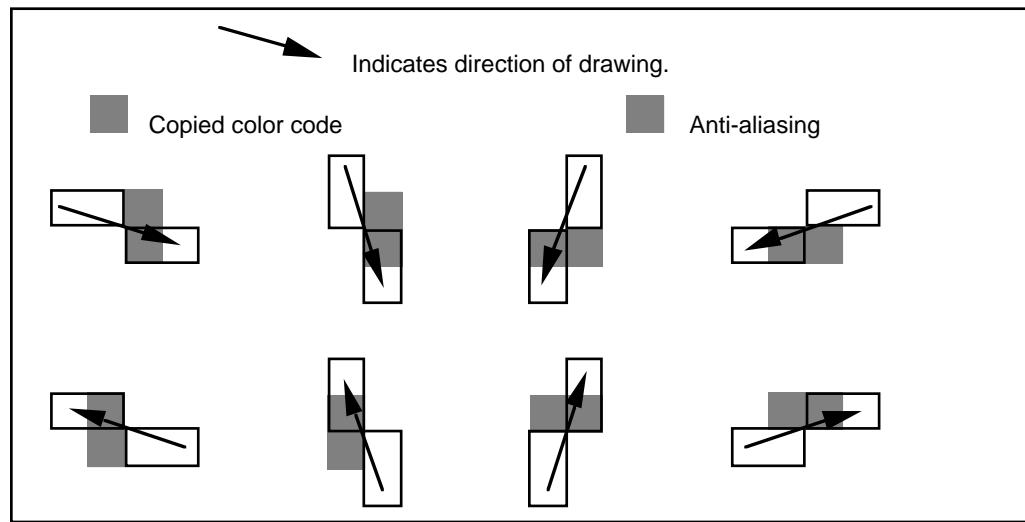


Figure 1.6 Anti-aliasing

## Non-Textured Parts

Non-textured parts are different from textured parts in that they do not require an original picture, and VRAM is not frequently accessed from the VDP1.

### Polygons

Draw a quadrangle by specifying four vertices and filling the enclosed area with a single color. The four vertices can be specified as desired. The color is specified as a non-textured color.

Polygons contain diagonal lines that may result in pixel dropout (aliasing). When this occurs, holes are anti-aliased. For this reason, some pixels may be written twice, and therefore the results of half-transparency processing as well as other color calculations cannot be guaranteed. Concave polygons can also be drawn. However, fills may extend outside of the polygon since lines are used to fill the polygon.

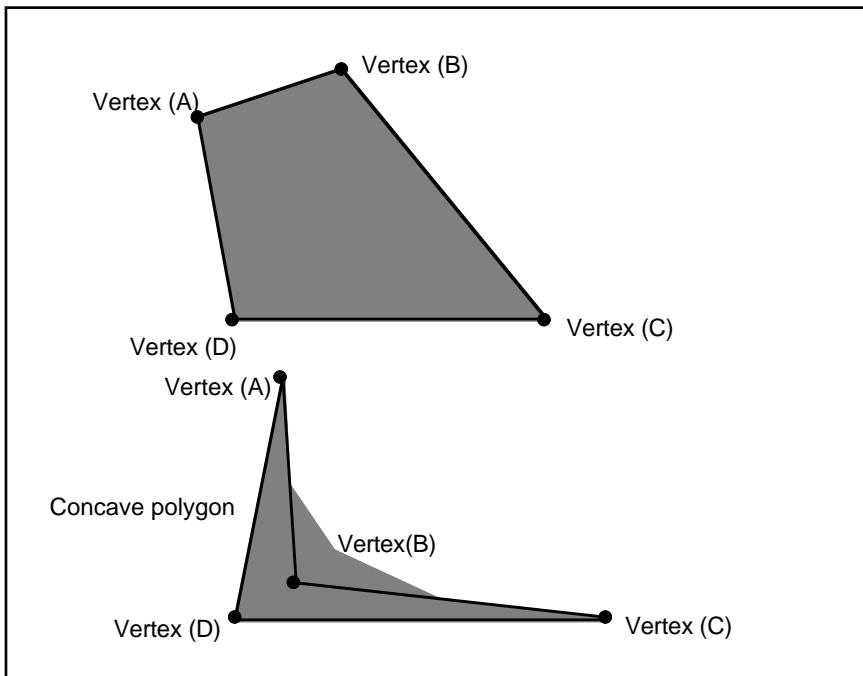


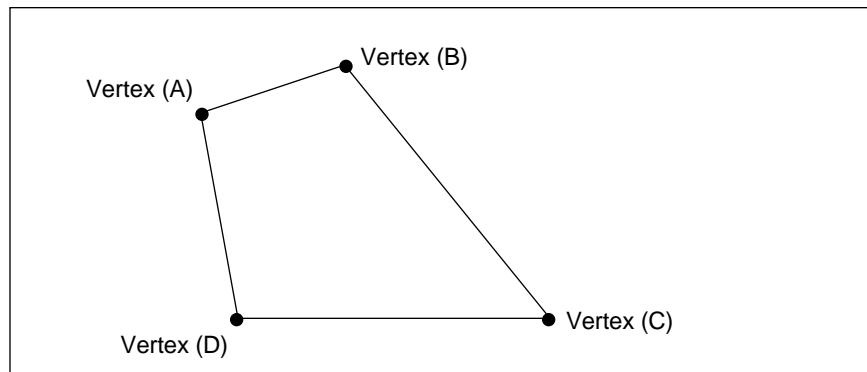
Figure 1.7 Polygons



### **Polyline**

A quadrangle is drawn by connecting four lines. Specify four vertices, and lines connecting the vertices are drawn in order. Unlike polygons, the area enclosed by the four vertices cannot be filled. The four vertices can be specified as desired.

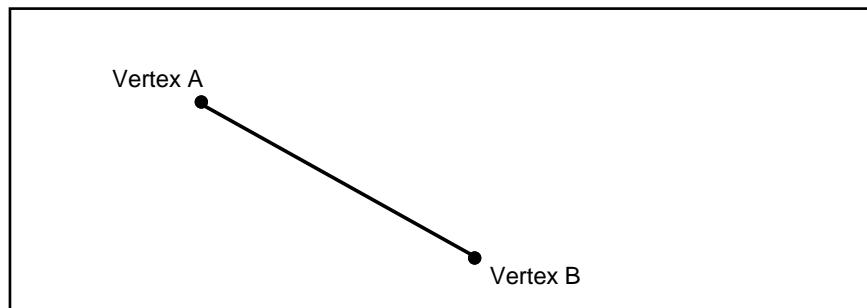
Specify the color as non-textured color. Because four lines are drawn, the pixels near the vertices are written twice. Therefore, the results of half-transparent processing and other color calculations cannot be guaranteed.



**Figure 1.8 Polyline**

### **Line**

A line is drawn in a single color between two specified coordinates. Specify the color as a non-textured color.



**Figure 1.9 Line**

## Color

The functions and special functions related to color are as follows.

- The number of colors per textured part unit is 16, 64, 128, 256, and 32,768.
- In the RGB mode, color calculation of half-luminance, half-transparency, Gouraud shading, and shadowing is possible.
- Mesh (tiling).

The color bank and RGB code methods are used to express the colors of pixels drawn (written) in the frame buffer. The color capability of the color bank method is 16, 64, 128, or 256 colors and the RGB code method is 32,768 colors.

### Color Bank Method

The color bank method combines the color bank with the palette codes for 16, 64, 128, and 256 colors, and references the colors stored in the color RAM of the VDP2. A 16 color palette code requires four bits of memory. The upper 12 bits of the color bank is added to the palette to give a total of 16 bits which is written to the frame buffer. Graphics that use 64, 128, or 256 colors (8-bit graphics) require a total of eight bits of memory. Only the lower 6 bits and 7 bits are significant for 64 and 128 color graphics, respectively, while 256 colors use all 8 bits. The upper 10, 9, and 8 bits of the color bank are added.

The data written by the color bank method are divided into function bits as shown in Figure 1.10 and processed in the VDP2. ([Refer to Sprite types, p. 200 of VDP2 User's Manual](#)).

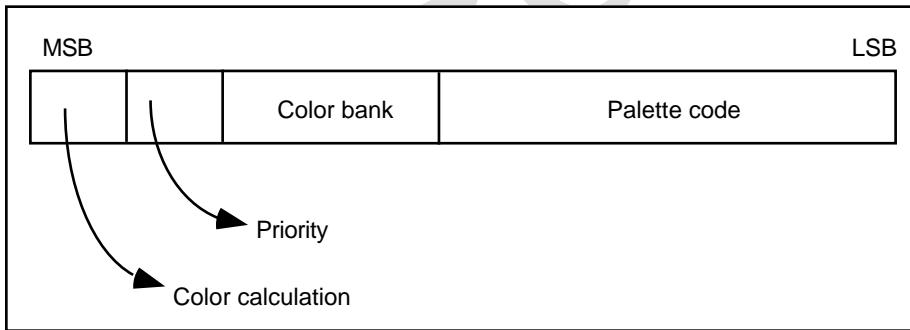
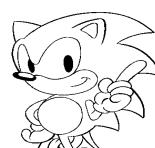


Figure 1.10 Bit Configuration of Color Bank Method

### RGB Code Method

The RGB code method represents colors using a 5-bit luminance for each of red, green, and blue (RGB) dot. The luminance of each of RGB is represented by 00H to 1FH. The closer the number is to 00H, less light is emitted from the RGB dots, the closer it is to 1FH, more light is emitted. Up to 32,768 colors can be formed by RGB codes. RGB codes are 16-bit data, and the most significant bit (MSB) is 1.



**Part Colors**

The color bank method, the color lookup table method, or the RGB code method can be used to set the color of sprites. Set a non-textured color for each non-textured part such as polygons, polylines, and lines.

**Color Lookup Table**

The color lookup table references the 4-bit graphic data to a 16-color lookup table and converts it to 16-bit data. Because the 16-bit data in the table are written as is to the frame buffer, there is no distinction between color bank codes and RGB codes for textured data.

When read to the VDP2, the code is handled as a color bank code when the MSB of the 16-bit data is "0" and as an RGB code when the MSB is "1."

**Non-textured Color**

Non-textured color is handled as pixel data without the use of a color bank or color lookup table.

**8 Bits/Pixel Frame Buffer**

When using high resolution or specifying a bit width of 8 bits / pixel for a rotated frame buffer, the graphics will be written to the frame buffer in 8 bits / pixel. When there are 8 bits / pixel, the lower 8 bits are written to the frame buffer when using a color lookup table or non-textured color. In either case, the lower 8 bits of the 16 bits are written to the frame buffer. This will be abbreviated as 8 bits / pixel (high resolution or rotation 8).

## 1.2 Screen Modes

### Screen Modes and Display Areas

Table 1.2 shows the screen modes and the coordinate values displayed in

each. **Table 1.2 Screen Modes and Frame Buffer Display Areas (without rotated display)**

Screen mode (DIE and TVM HDTV enable)	Display area size in pixels	X range		Y range
		Standard	High resolution	
Non-interlace Single interlace NTSC, PAL	320/640 H × 224 V	0≤X≤319	0≤X≤639	0≤Y≤223
	320/640 H × 240 V	0≤X≤319	0≤X≤639	0≤Y≤239
	320/640 H × 256 V	0≤X≤319	0≤X≤639	0≤Y≤255
	352/704 H × 224 V	0≤X≤351	0≤X≤703	0≤Y≤223
	352/704 H × 240 V	0≤X≤351	0≤X≤703	0≤Y≤239
	352/704 H × 256 V	0≤X≤351	0≤X≤703	0≤Y≤255
Double interlace NTSC, PAL	320/640 H × 224 V	0≤X≤319	0≤X≤639	0≤Y≤447
	320/640 H × 240 V	0≤X≤319	0≤X≤639	0≤Y≤479
	320/640 H × 256 V	0≤X≤319	0≤X≤639	0≤Y≤511
	352/704 H × 224 V	0≤X≤351	0≤X≤703	0≤Y≤447
	352/704 H × 240 V	0≤X≤351	0≤X≤703	0≤Y≤479
	352/704 H × 256 V	0≤X≤351	0≤X≤703	0≤Y≤511
Non-interlace 31KC, HDTV	320 H × 240 V	0≤X≤319	—	0≤Y≤239
	352 H × 240 V	0≤X≤351	—	0≤Y≤239

H = Horizontal

V = Vertical

- These are coordinate ranges and are not the sizes of the areas occupied in the frame buffer. For more information about the size of the frame buffer, see “Frame Buffer” in Section 2.1 Address Map.
- A high-resolution frame buffer has a bit depth of 8 bits/pixel, while standard mode frame buffer can be 8 or 16 bpp. Rotated display of a high-resolution frame buffer is not possible.
- Rotated display of the frame buffer is not possible with double interlace.
- A double interlaced display system may use graphic information from 2 consecutive frame buffers to make a frame with double vertical resolution (1/30 second).
- A single interlaced display system switches frame buffers every two fields and displays the same picture in both the odd and even display lines. In both cases, there are no gaps between the scanning lines.
- The resolution in single interlace and non-interlace are the same, and the method of writing to the frame buffer is the same.
- In 31KC and HDTV, a frame buffer pixel is displayed in 4-pixel units on screen, 2 vertical pixels and 2 horizontal pixels. That is, the sprite resolution is the same as 320 x 240 (31KC) and 352 x 240 (HDTV) in the standard mode, and the method of writing to the frame buffer is the same.
- For information about how to set the interlace, refer to TV Screen Mode, p. 12 of VDP2 User's Manual.

#### Notes

Field: The time it takes the scanning lines to scan one screen (1/60 second).

Frame: The time period during which one image is displayed. If single interlaced, two fields make one frame (1/30 second.)



## Rotated Reading of Frame Buffer

- By reading the frame buffer diagonally, the entire frame buffer plane can be displayed rotated.
- Display coordinates that exceed the frame buffer address range are handled as transparent (XX00H for 8-bit graphics and 0000H for 16-bit graphics).
- Even if rotated reading of the frame buffer is performed, the clipping area and erase/write area remain fixed with respect to the frame buffer plane. Therefore, the clipping area and erase/write area become inclined with respect to the display screen.
- To prevent dropout of any of the display screen when the frame buffer is displayed rotated, the coordinate range of the frame buffer must be made large. In this case the frame buffer is set to 8 bits/pixel and the screen to  $512 \times 512$  rather than  $512 \times 256$ . The number of colors that can be expressed at one time becomes fewer than 256.
- Rotation is prohibited when normal, high resolution, HDTV, or double interlace is set.
- The read start coordinates and read movement value for rotated reading from the frame buffer are received from the VDP2 (see "Rotation Read Out of the Frame Buffer", p. 159 of VDP2 User's Manual).
- Rotated reading of the frame buffer is only valid in rotation 16 and rotation 8, and is prohibited in all other cases. In the case of non-rotation, any rotation data received from the VDP2 are invalid and the parameters of the VDP1 become valid.

(Page 16 is blank in the original Japanese version.)

SEGA Confidential



# **Chapter 2**

## **Address Map**

### **Contents**

2.1	Address Map .....	18
	VRAM .....	19
	Frame Buffer .....	20
	System Registers .....	23
2.2	Tables in VRAM .....	24
	Command Table .....	25
	Color Lookup Table .....	26
	Gouraud Shading Table .....	26
	Character Pattern Table .....	26

## 2.1 Address Map

Figure 2.1 shows the address map for the VRAM, frame buffer, and system registers controlled by the VDP1.

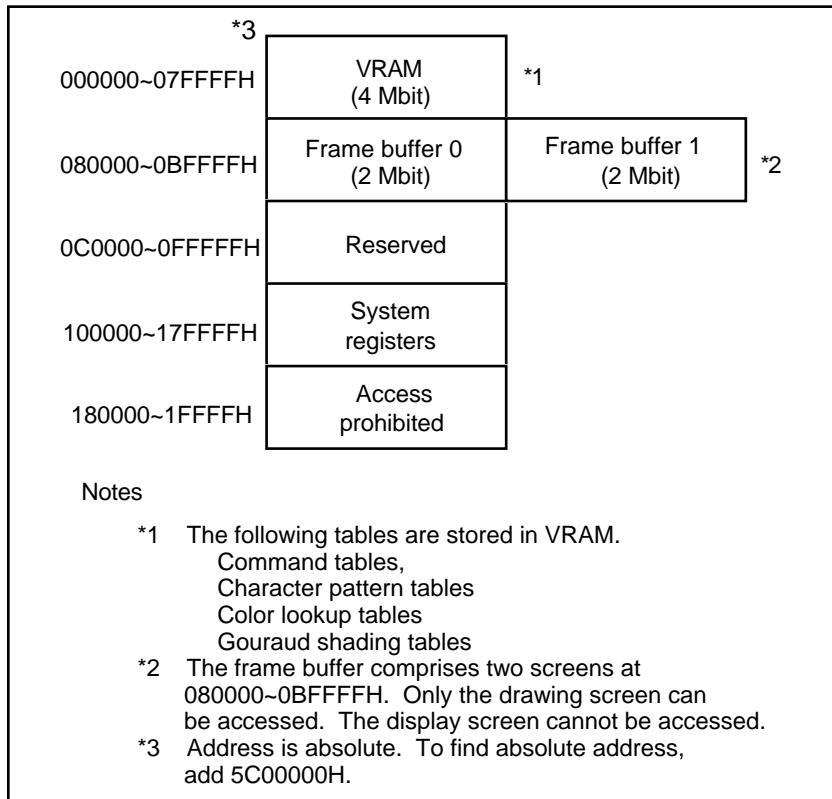


Figure 2.1 Address Map



## VRAM

- The VRAM is a 4-Mbit DRAM.
- The command table, sprite character pattern table, color lookup table, and Gouraud shading table are defined in VRAM. It doesn't matter where in VRAM the data of each are, or whether they are in a nested condition. They are referenced by specifying the address of each.
- Fetching of the command must be performed from the top (000000H) of VRAM.
- Fetching of VRAM is repeated in the following order:
  - Command table
  - Gouraud shading table (only when Gouraud shading is used)
  - Color lookup table (only when lookup table method is used)
  - Character pattern table (only when drawing sprites)
- When fetching of the command table goes beyond the end address (07FFFFH) of VRAM, fetching wraps to the top address (000000H) of VRAM.
- Byte access and word access are both possible from the CPU.
- Read-write access of the VRAM by the system controller IC, and parameter read and pattern read access by the VDP1, are performed after assigning an order of priority.
- The order of priority of access of the VRAM is always: system controller (system controller IC) > drawing.
- Because access is performed after assigning an order of priority, there may be more than 10 wait cycles according to that timing, depending on the operating clock of the CPU. The operating clock of the CPU is 28 MHz.
- Perform access from the CPU when drawing is not being performed in order to prevent interruption of drawing. To determine if drawing is being performed, poll the system registers, or use an interrupt signal. Access of the VRAM and frame buffer is performed in a short period of time using burst transfer.

## Frame Buffer

- The frame buffer comprises two 2-Mbit DRAM and is divided into two screens: a display frame buffer and a draw frame buffer.
- The function of the two buffers is changed immediately before the screen display period (DISP). After powering on or resetting, frame buffer 0 becomes the drawing frame buffer, and frame buffer 1 becomes the display frame buffer.
- Read-write access of the frame buffer by the system controller IC is performed only on the draw frame buffer. The display frame buffer becomes the rear bank, so it cannot be accessed.
- Drawing is performed in sync with the CPU operating clock. The CPU operating clock is 28 MHz, and the data for 1 pixel is drawn in sync with this.
- Read-write access of the frame buffer by the system controller IC and draw access by the VDP1 are performed after assigning an order of priority.
- The order of priority of access of the frame buffer is always: system controller IC > drawing
- When access from the system controller is performed during drawing, drawing is interrupted and must wait. Therefore this situation should be avoided as much as possible.
- Because drawing and access of the CPU are not performed together, perform control by using a method that uses a manual start for drawing start.
- Access can be performed using word access. **Byte access is only possible when the frame buffer bit depth is 8 bits/pixel. Do not use byte access when bit depth is 16 bpp.**
- The entire frame buffer can be displayed rotated by reading the frame buffer diagonally. Also, by skipping or repeating read addresses, the entire frame buffer plane can be enlarged or reduced. Rotation, enlargement, and reduction can be performed simultaneously.
- Rotation is specified from the VDP2. Refer to the VDP2 instruction manual for more information about the specification of rotation.



### Frame Buffer Plane

The coordinates of the frame buffer plane are as follows. Coordinates increase in value toward the lower-right.

$$X \quad -1024 \leq X \leq 1023$$

$$Y \quad -1024 \leq Y \leq 1023$$

Operation cannot be guaranteed when specified values exceed these values.

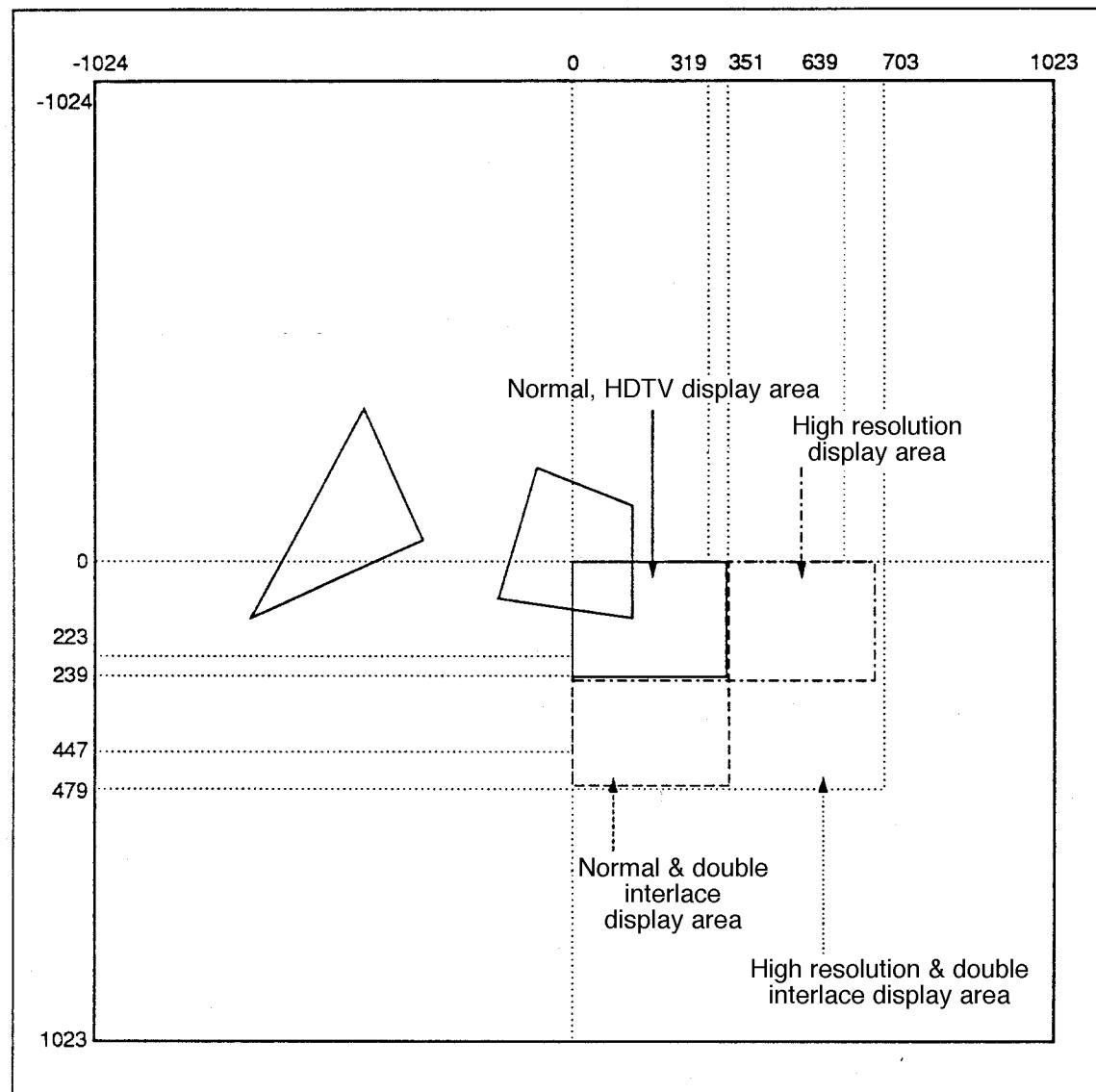


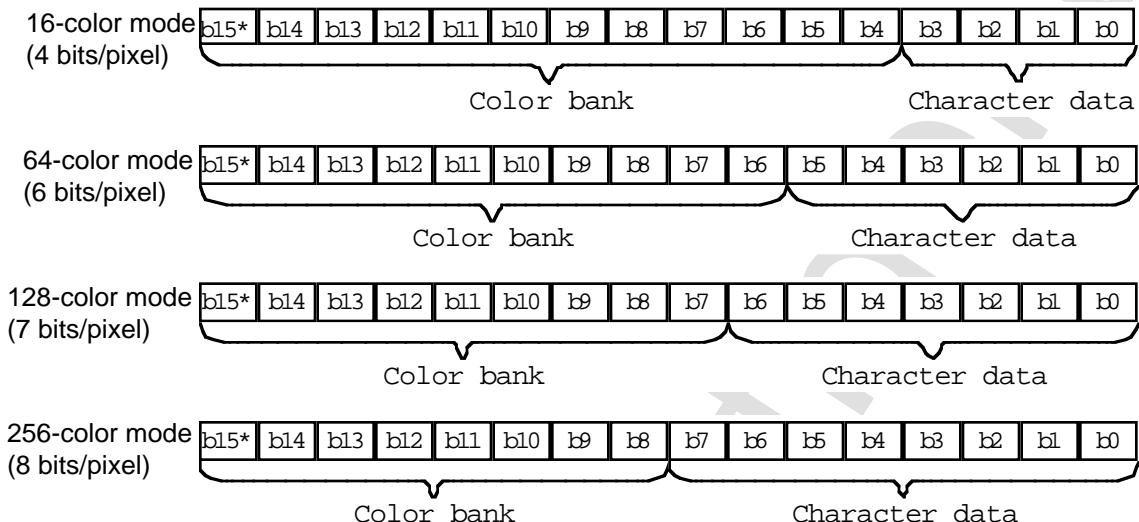
Figure 2.2 Frame Buffer Plane

As shown in Figure 2.2, parts can be positioned outside the display screen. However, nothing is written for parts that exceed the range of the frame buffer plane.

### Pixel Data in Frame Buffer

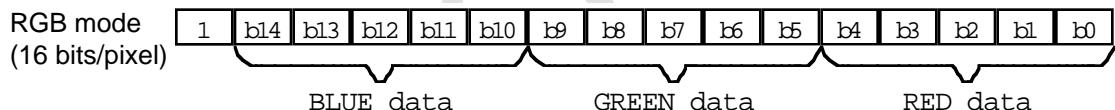
The pixel data in the frame is shown below. There is 16-bit and 8-bit data. RGB data, which does not go through the color RAM in the VDP2, is only 16-bit data. When all bits are 0, the dot is treated as a transparent dot by the VDP2. **Color RAM pixels containing the normal shadow palette code will be treated by VDP2 as transparent with optional half luminance on background** (see "The Normal Shadow" on p. 165).

- Bit configuration when data goes through color RAM (color RAM address)

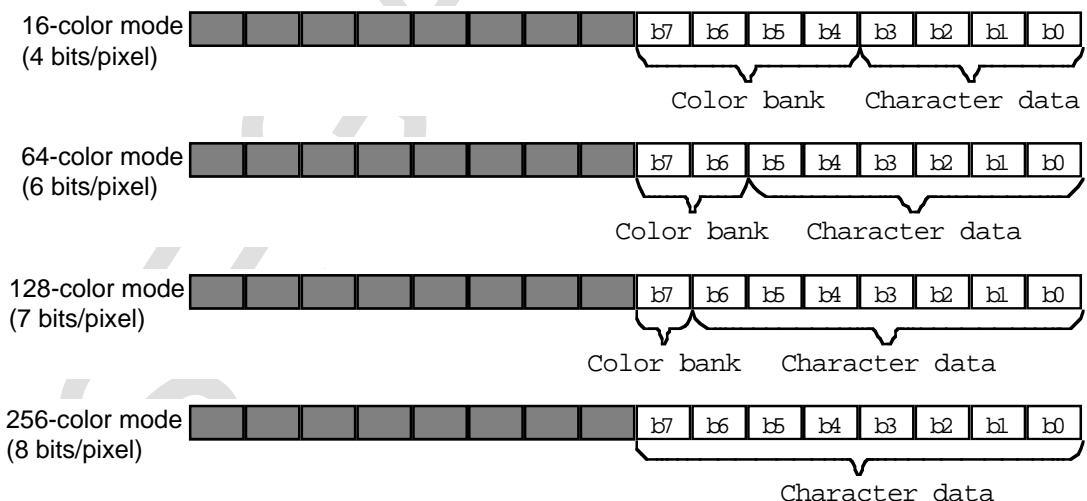


Note: b15 = 0 when RGB data are mixed; b15 can be either when not mixed.

- Bit configuration when data do not go through color RAM (RGB data)



- Bit configuration when pixel data is 8 bit (color RAM address)



## System Registers

- System registers are memory used for making system settings for the VDP1. They are housed inside the VDP1 separately from the VRAM and frame buffer.
- There are read-only registers and write-only registers.
- The write-only registers are used to control display of the frame buffer. They select the TV mode, specify change of display and the drawing trigger, and define the fill data and area for erase / write. Drawing can also be forcibly terminated.
- Read-only registers are used as help information during program development. They make it possible to know the address of the command table that underwent draw processing last.
- Read / write access from the CPU must be performed in word units.
- Do not use DMA burst transfer when accessing the system registers.
- Except for the plot trigger register (PTMR), the values in the write-only registers become undefined after powering on and after resetting, so be sure to set the values from the CPU. Undefined data is displayed from the frame buffer until a suitable value is set.
- Set the unused bits of write-only system registers to “0”.

**Table 2.1 System Registers**

Address	Name	Description	Access
100000H	TVMR	TV mode selection	Write-only (word)
100002H	FBCR	Frame buffer change mode	Write-only (word)
100004H	PTMR	Draw trigger	Write-only (word)
100006H	EWDR	Erase/write data	Write-only (word)
100008H	EWLR	Erase/write upper-left coordinate	Write-only (word)
10000AH	EWRR	Erase/write lower-right coordinate	Write-only (word)
10000CH	ENDR	Draw forced termination	Write-only (word)
100010H	EDSR	Transfer end status	Read-only (word)
100012H	LOPR	Last operation command address	Read-only (word)
100014H	COPR	Current operation command address	Read-only (word)
100016H	MODR	Mode status	Read-only (word)

## 2.2 Tables in VRAM

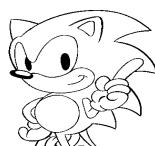
The command table, color lookup table, Gouraud shading table, and character pattern table are defined in VRAM. Table 2.2 shows the sizes and boundaries of the tables.

**Table 2.2 Tables in VRAM**

Name	Function		Commands	Size	Boundary
Command Table	Draw Commands	Textured Draw Command	Normal sprite draw command Scaled sprite draw command Distorted sprite draw command	1EH	20H
		Non-textures	Polygon draw command Polyline draw command Line draw command		
		Coordinate Set Commands	User clipping coordinate set command Clipping coordinate set commands System clipping coordinate set command Local coordinate set command		
	Drawing end command				
Color lookup table				20H	20H
Gouraud shading table				8H	8H
Character pattern table				optional*	20H

Note: \*Differs depending on the character size and color mode.

The VRAM is 4 Mbit (512 Kbyte), and it is addressed in byte units from 000000H to 07FFFFH. Table data cannot be written beyond 07FFFFH. Each table must be kept within the size of the VRAM.



## Command Table

- The command table is a table in VRAM where commands are defined, and in which the VDP1 reads commands, draws parts and processes clipping.
- The commands are as follows:

Draw commands

These are divided into texture drawing and non-texture drawing. Texture drawing includes normal sprite, scaled sprite, and distorted sprite draw commands. Non-texture drawing includes polygon, polyline, and line draw commands. Draw commands draw these parts.

Clipping coordinate set commands

Includes user clipping and system clipping coordinate set commands, and they set the draw area for parts.

Local coordinate set commands

**Specifies the offset of the parts draw coordinate as local coordinate.**

Draw end command

Terminates drawing.

- The size of the command table is 1EH (30) bytes, and its boundary is 20H (32) bytes.
- Each command is read to the VDP1 and is processed. This operation is referred to as fetching.
- Fetching of the command table is performed from the top address (000000H) of VRAM, and the next command table is fetched according to the specification of the jump mode.
- According to the command table specification, the color lookup table, Gouraud shading table, and character pattern table are referenced after the command table.

## **Color Lookup Table**

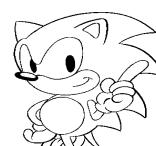
- This is a table in which 16-bit color codes for 16 colors are defined in VRAM.
- When a lookup table method is used for the color mode according to the texture draw command, the table is referenced as color data.
- The pixel data of character patterns is converted to color codes and written to the frame buffer.
- The size of the table is 20H (32) bytes, and its boundary is also 20H (32 ) bytes.
- The table is referenced according to the instruction at the lookup table address of the texture draw command.

## **Gouraud Shading Table**

- This is a table in VRAM in which 16-bit RGB codes for 4 points are defined.
- It is referenced when the processing of Gouraud shading by color calculation is instructed by the draw command for the part.
- The pixel data of the part undergoes processing for Gouraud shading and is written to the frame buffer.
- The size of the table is 8H bytes, and its boundary is also 8H bytes.
- It is referenced according to the instruction at the Gouraud shading table address of the draw command for the part.
- When Gouraud shading is performed, the pixel data of the part is limited to RGB code. When the pixel data of the part is a color bank code, the results cannot be guaranteed.

## **Character Pattern Table**

- This is a table in VRAM in which the pixel data for character patterns is defined.
- The table is referenced by the texture draw command.
- The pixel data is defined as 4, 8, or 16 bits / pixel, according to the specification of the color mode.
- In the color bank mode, a color bank is added to the pixel data of the character pattern; in the color lookup table mode it is converted in the color lookup table; in the RGB mode it is written, as is, to the frame buffer.
- The size of the table is determined by the size of the characters and the color mode. Its boundary is 20H (32) Bytes.
- The table is referenced according to the instruction at the character address of the texture draw command.



# **Chapter 3**

## **Processing Flow**

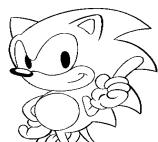
### **Contents**

3.1 Draw Procedure Flow .....	28
3.2 Command Table Flow .....	30
3.3 Table Referencing .....	31

### 3.1 Draw Procedure Flow

Draw procedure flow using the VDP1 is as follows.

- Step 1. Power on the system.
- Step 2. Set the necessary values in the system registers of the VDP1.
- Step 3. Write the necessary character pattern table to VRAM.
- Step 4. Write the necessary color lookup table to VRAM.
- Step 5. Write the necessary Gouraud shading table to VRAM.
- Step 6. Write the necessary command table to VRAM.
- Step 7. Drawing to the frame buffer starts automatically at the start of frame change, and the drawn frame buffer is displayed in the next frame change.
- Step 8. Repeat steps 3 through 6 as required.



## Table Access

The procedure VDP1 uses to access the table in VRAM and draw is as follows:

- Step 1. Controls drawing and display according to the instructions set in the system registers.
- Step 2. Fetches the command table at the top address of VRAM.
- Step 3. The fetched command table:
  - (1) Terminates drawing in the case of a draw end command (goes to step 9).
  - (2) Is ignored when jump mode is skipped; reading of the table is terminated and the table is not processed (goes to step 8).
  - (3) In cases other than (1) and (2), goes to step (4).
- Step 4. In the case of a clipping coordinate set command or a local coordinate set command, each is processed (goes to step 8).
- Step 5. In the case of a drawing command, the Gouraud shading table and color lookup table are read if specified.
- Step 6. In the case of a textured drawing, the character pattern table is read and is written to the frame buffer according to the specification. At this time, processing of the color mode, color calculation, inversion, enlargement and reduction, and rotation are performed.
- Step 7. In the case of a non-textured drawing, writing to the frame buffer is performed according to the specification.
- Step 8. The next command table is fetched according to the specification of the jump mode and processing of the command table is repeated (goes to step 3).
- Step 9. Drawing is repeated with the start of framing (goes to step 1).

## 3.2 Command Table Flow

Except for the draw end command, a jump mode to the next command table to be processed can be specified in other commands. Those jump modes include the following.

- Jump to a command table
- Skip to a command table
- Call a command table group (subroutine)
- Return (to main routine)

Figure 3.1 shows an example of the flow of a command using a jump mode.

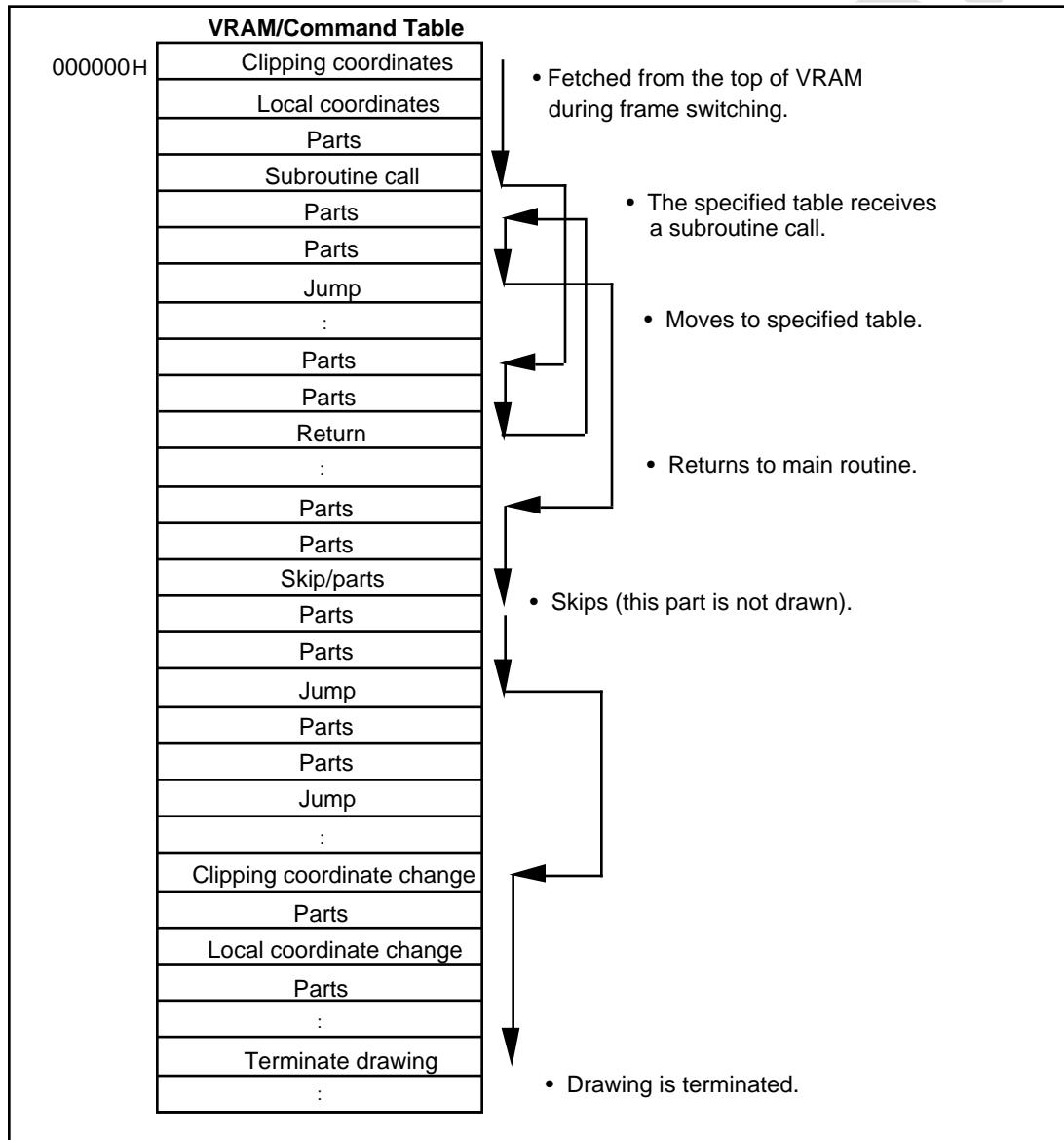


Figure 3.1 Command Table Flow



### 3.3 Table Referencing

Referencing of the tables stored in VRAM begins with the following command table.

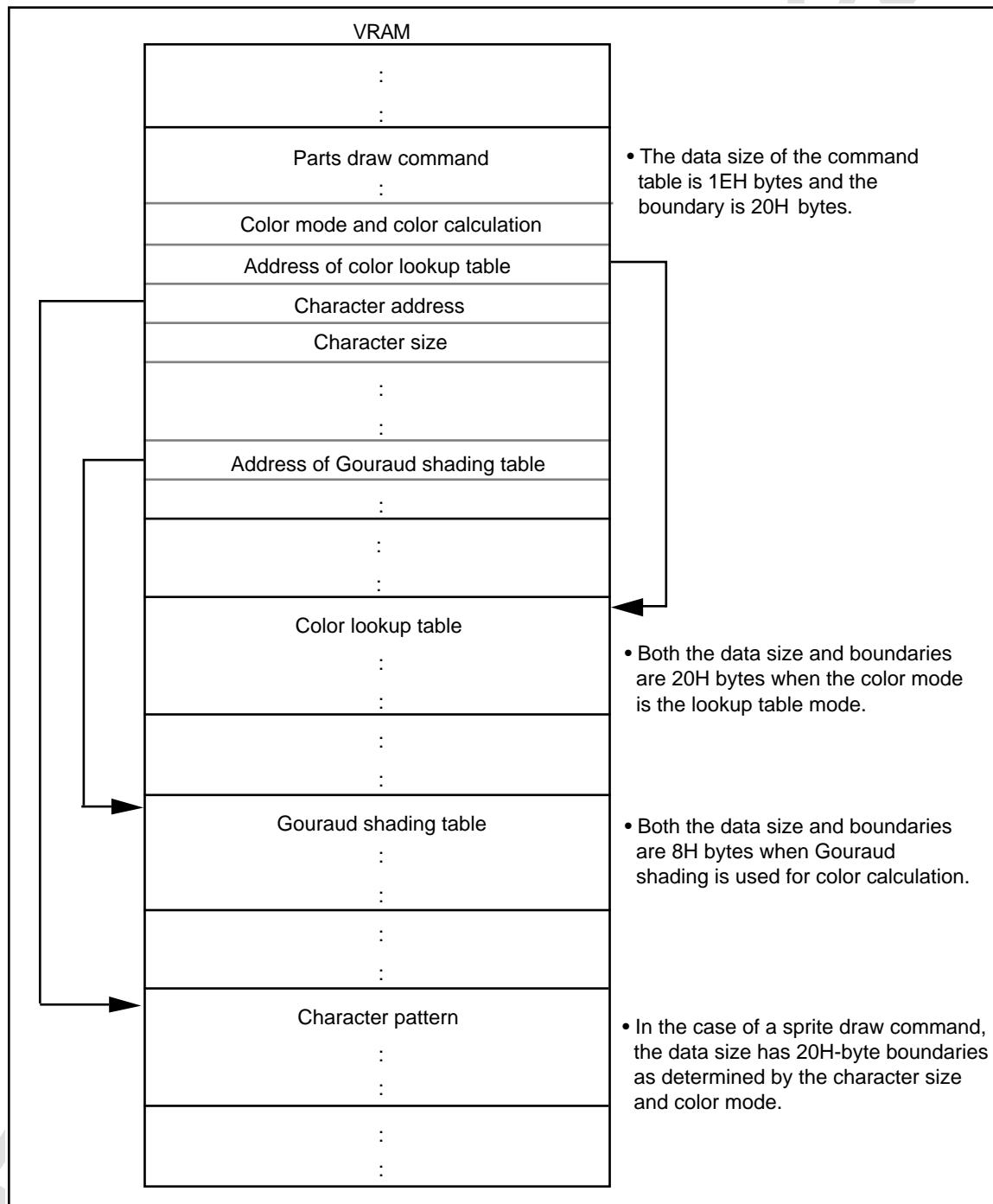


Figure 3.2 Referencing of Tables

(This page is blank in the original Japanese document.)

SEGA Confidential



# Chapter 4

## System Registers

### Contents

4.1	TV Mode Selection Register .....	36
4.2	Frame Buffer Change Mode Register .....	38
4.3	Plot Trigger Register .....	45
4.4	Erase/Write .....	46
	Erase/Write Data Register .....	46
	Erase/Write Upper-Left Coordinate Register .....	47
	Erase/Write Lower-Right Coordinate Register .....	47
4.5	Draw Forced Termination Register .....	51
4.6	Transfer End Status Register .....	52
4.7	Last Operation Command Address Register .....	54
4.8	Current Operation Command Address Register .....	55
4.9	Mode Status Register .....	57

The following table shows the functions of the systems registers.

**Table 4.1 System Registers**

Register Names	Abbrevi-ation	Address	Access	Function	Internal update period
TV mode selection	TVMR	100000H	Write-only (word)	Specifies TV display mode	PTM as occasion demands VBE is field
Frame buffer switch mode	FBCR	100002H	Write-only (word)	Controls frame buffer toggle and double interlace mode	FCM and FMT are set for each field, others are frame buffer SW timing
Plot trigger	PTMR	100004H	Write-only (word)	Controls start of drawing	01B write as the occasion demands, 00B and 10B write are frame buffer SW timing
Erase/write data	EWDR	100006H	Write-only (word)	Specifies fill data for frame buffer during erase/write	Frame buffer SW timing
Erase/write upper-left coordinate	EWLR	100008H	Write-only (word)	Specifies upper-left coordinate of area in frame buffer to fill during erase/write	Frame buffer SW timing
Erase/write lower-right coordinate	EWRR	10000AH	Write-only (word)	Specifies lower-right coordinate of area in frame buffer to fill during erase/write	Frame buffer SW timing
Plot abnormal end	ENDR	10000CH	Write-only (word)	Forces termination of drawing	As the occasion demands
Transfer end status	EDSR	100010H	Read-only (word)	Status of 'END' bit of current/previous frame	–
Last Operation Command Address	LOPR	100012H	Read-only (word)	Address of last accessed command table for previous frame	–
Current Operation Command Address	COPR	100014H	Read-only (word)	Address of last accessed command table being processed	–
Mode status	MODR	100016H	Read-only (word)	Displays setting of write-only register	–



## **System Register Settings Switch Timing**

The timing with which the system register settings become valid is as follows.

- Settings that change immediately
  - Plot trigger mode (PTM, when 01B is written)
  - TV mode selection (TVM)
- Settings that change with each field (1/60 second)
  - Frame buffer change mode (FCM)
  - Frame buffer change trigger (FCT)
- Settings that change with the switching of the frame buffer
  - Plot trigger mode (PTM, when 00B or 10B is written)
  - Even/odd coordinate selection bit (EOS)
  - Double-density interlace enable (DIE)
  - Double-density interlace draw line (DIL)
  - Erase/write data
  - Erase/write coordinates (upper-left, lower-right)
- Changes following the termination of display of one line after the V-blank IN interrupt.
  - Enables V-blank erase/write (VBE)

## 4.1 TV Mode Selection Register

The TV mode selection register (TVMR, TV mode register) enables V-blank erase and specifies the TV display mode. It is a 16-bit write-only register, and is at address 100000H. Its value becomes undefined after powering on or resetting; therefore the TV display mode must be set. The unused bits must be set to "0."

TVMR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100000H		0	0	0	0	0	0	0	0	0	0	0	0	VBE	TVM		

Write-only

### V-Blank Erase/Write Enable (VBE): bit 3

- When VBE = 1, erase / write is performed during V-blank. VBE = 1 can only be set when FCM = 1 and FCT = 1 are written. For more information, refer to section 4.2, "Frame Buffer Change Mode Register."
- When VBE is set to "0" or "1," TVM of the same value before making the setting must be set at the same time.
- The VBE setting must be set immediately after the V-blank IN interrupt. Access is prohibited from the first H-blank IN interrupt after the V-blank IN interrupt until the next H-blank IN interrupt.

### TV Mode Select (TVM): bits 2~0

- Specifies the TV display mode.
- For more information about the coordinate values of the displayed area, refer to "Screen Mode and Display Areas" in section 1.2.
- When TVM = 010B(2), the frame buffer has only 512 (H) x 256 (V) pixels, and therefore if the screen is greatly inclined, a transparent area occurs in the display screen.
- Since only non-interlace is possible when HDTV is specified (TVM = 100B(4)), and only non-interlace and single interlace are possible when rotated display is specified (TVM = 010B(2), 011B(3)), the frame buffer change mode register must be set to double-interlace disable (DIE bit = 0). For more information, refer to section 4.2, "Frame Buffer Change Mode Register."
- TVM settings must be performed from the second H-blank IN interrupt after the V-blank IN interrupt to the H-blank IN interrupt immediately after the V-blank OUT interrupt.



- The function of each bit is as follows.

Bit 2: HDTV enable bit

0 = NTSC, PAL

1 = HDTV, 31KC

Bit 1: Frame buffer rotation enable bit

0 = non-rotation

1 = rotation

Bit 0: Bit depth selection bit

0 = 16 bits/pixel

1 = 8 bits/pixel

**Table 4.2 Screen Modes**

TVM			Screen Mode		Bit width (bit/pixel)	Frame buffer screen size	Possible interlace (DIE)	VDP CLK (MHz)	
Bits 2	1	0	TV mode name	Sprite resolutions per field H x V				NTSC	PAL
0	0	0	Normal (NTSC, PAL)	320x224 320x240 <b>320x256</b> 352x224 352x240 <b>352x256</b>	16	512 H x 256 V	Single or double	26.8426 26.8426 - 28.6364 28.6364 -	26.6564 26.6564 <b>26.6564</b> 28.4375 28.4375 <b>28.4375</b>
0	0	1	High Resolution (NTSC, PAL, Hi-Res)	640x224 640x240 <b>640x256</b> 704x224 704x240 <b>704x256</b>	8	1024 H x 256 V	Single or double	26.8426 26.8426 - 28.6364 28.6364 -	26.6564 26.6564 <b>26.6564</b> 28.4375 28.4375 <b>28.4375</b>
0	1	0	Rotation 16 (NTSC, PAL Rotation)	320x224 320x240 <b>320x256</b> 352x224 352x240 <b>352x256</b>	16	512 H x 256 V	Single Only	26.8426 26.8426 - 28.6364 28.6364 -	26.6564 26.6564 <b>26.6564</b> 28.4375 28.4375 <b>28.4375</b>
0	1	1	Rotation 8 (NTSC, PAL Rotation)	320x224 320x240 <b>320x256</b> 352x224 352x240 <b>352x256</b>	8	512 H x 512 V	Single Only	26.8426 26.8426 - 28.6364 28.6364 -	26.6564 26.6564 <b>26.6564</b> 28.4375 28.4375 <b>28.4375</b>
1	0	0	HDTV (31KC, HDTV)	320x240 352x240	16	512 H x 256 V	No	26.8426 28.6364	26.6564 28.4375
All other			Setting prohibited (do not use)						

Sprite resolution per field depends on TVM and VDP2 settings (see "TV Screen Mode Register", p. 16 of VDP2 User's Manual) :

- If VDP2 and the frame buffer both are high res, or if both are not high res, then sprite and VDP2 horizontal resolutions are the same.

- If VDP2 is high res, and the frame buffer isn't high res, then sprite horizontal resolution is half of VDP2 horizontal resolution. Each frame buffer pixel is displayed on 2 horizontal pixels on screen.

- If VDP2 is not 480p (Dedicated/Exclusive/Special monitor mode), then sprite and VDP2 vertical resolution per field are the same.

- If VDP2 is 480p, then sprite vertical resolution is half of VDP2 vertical resolution. Each frame buffer pixel is displayed on 2 vertical pixels on screen.

See table on Page 164 (page 180 PDF) for possible combinations of VDP2 horizontal resolution and VDP1 TV mode.

## 4.2 Frame Buffer Change Mode Register

The frame buffer change mode register (FBCR, frame buffer change register) controls drawing and display change of the frame buffer, as well as double interlace drawing. It is a 16-bit read-only register at address 100002H. Its value becomes undefined after powering on or resetting, and therefore the change mode must be set. Unused bits must be set to "0."

FBCR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100002H		0	0	0	0	0	0	0	0	0	0	0	EOS	DIE	DIL	FCM	FCT

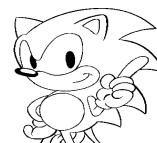
Write-only

**Frame Buffer Change Mode (FCM): bit 1**

**Frame Buffer Change Trigger (FCT): bit 0**

- Normally the two frame buffers are used as a front screen and a back screen. The front screen is displayed and the back screen is drawn. Also, the front screen and back screen are toggled after the time required to change one frame has elapsed. The front screen that was displayed becomes the back screen to be drawn, and the back screen that was being drawn becomes the front screen to be displayed.
- Only that part of the frame buffer being drawn can be accessed. The side being displayed cannot be accessed.
- The number of characters that can be drawn in one field is limited. Therefore, in order to draw more characters, the manual mode must be set for frame buffer erasure and change to be controlled by the CPU.
- When selected data is written from the CPU to FBCR, the values of FCM and FCT are updated at the time of a field change in the VDP1.
- When in the manual mode, the FCT bit setting is only valid for the next field. The contents of FCT are updated internally at the time of frame change.
- Make the FCM and FCT settings immediately after the V-blank OUT interrupt. Access is prohibited from the first H-blank IN interrupt after the V-blank OUT interrupt until the next H-blank IN interrupt.

VBE	FCM	FCT	Change mode	Change time
0	0	0	1-cycle mode	Change every 1/60 second
0	0	1	Setting prohibited	—
0	1	0	Manual mode (erase)	Erase in next field
0	1	1	Manual mode (change)	Change in next field
1	0	0	Setting prohibited	—
1	0	1	Setting prohibited	—
1	1	0	Setting prohibited	—
1	1	1	Manual mode (erase and change)	Erase by V-blank and change in next field



### 1-Cycle Mode

- This is the normal mode.
- One frame changes automatically every 1/60 second.
- Set the value of the VBE, FCM, and FCT bits to "0".

### Erase (Manual Mode)

- Erase / write for one frame is controlled from the CPU.
- Manual erase/write of the frame buffer is specified by writing "0" to the VBE and FCT registers and "1" to the FCM register. Erase / write of the display frame buffer is performed in the next specified field, and the display frame buffer is cleared.
- Changing of the drawing and display frame buffers is not performed. Because the display frame buffer is cleared when erase is performed, it is necessary to specify change in the next field when display is performed for which erase is specified and to perform change of drawing and display.
- This mode is used when there are two or more fields in one frame, as in double interlace. Also, when changing from manual mode to the 1-cycle mode, this erase is used in the field before changing.
- Erase is used when it is known that frame change will be performed two fields in advance. Use erase and change when it is unclear up until the prior field that frame change will be performed.

### Change (Manual Mode)

- The changing of one frame is controlled by the CPU.
- Change is specified by writing "0" to the VBE register and "1" to the FCM and FCT registers. The drawing and display are changed in the next specified field.
- Because erase / write is not performed, it is necessary to specify erase in the prior field in which change is specified and to erase / write the frame buffer.
- Until change is specified, characters can be drawn in the back screen.
- This change is normally used when switching from the 1-cycle mode to the manual mode. This change is also used when changing the frame buffer in manual mode.

## Erase & Change (Manual Mode)

- By writing "1" to the VBE register, V-blank erase is specified; and by writing "1" to the FCM and FCT registers at the same time, switching of the frame buffer after V-blank is specified.
- This mode is used when frame change is unclear up until the prior V-blank, as when the frame buffer cannot be changed by the end of processing by the CPU.
- Once VBE has been set to "1," erase / write is performed automatically even in the next V-blank, and therefore VBE must be set to "0" before the next V-blank after frame change.
- Place the erase & change setting immediately after the V-blank IN interrupt. When set at another time, the change may be performed without being able to erase.
- If the timing for frame change is known in advance in normal or high resolution screen modes, specify erase then change and perform erase/write during the display period. In this case, erase/write of the entire display screen is possible.

- 1) Set VBE to "0" and FCM and FCT to "1."
- 2) Wait for the end of processing by the CPU without selecting the TV mode or setting the FB change mode.
- 3) When processing by the CPU ends by the H-blank IN interrupt time (224th line in 224-line display and 240th line in 240-line display) immediately before V-blank, then VBE, FCM, and FCT are set to "1" (erase & change). The erase & change setting should be placed immediately after the V-blank IN interrupt.
- 4) V-blank erase is started after completion of the V-blank IN interrupt.
- 5) At the end of V-blank, erase / write is interrupted and the frame is changed.
- 6) If erase / write is not completed, erase / write non-erased areas with polygons.
- 7) **Return VBE to "0" after the V-blank OUT interrupt to prevent V-blank erase to occur automatically in the next V-blank.**
- 8) Return to 2).



**Example**

Table 4.3(a) shows the frame buffer change mode being used.

**Table 4.3(a) Example of Use of Frame Buffer Change Mode (Fixed at VBE = 0)**

Setting <sup>1</sup>		Frame buffer 0 <sup>2</sup>	Frame buffer 1 <sup>2</sup>	Frame buffer change mode	Change time
FCM	FCT				
0	0	Draw	Display and erase/write	1-cycle mode	60 frames/sec
		Display and erase/write	Draw		
		Draw	Display and Erase/write		
		Display and erase/write	Draw		
1	1	Draw	Display and erase/write	Manual mode (change) <sup>3</sup>	20 frames/sec
		Display	Draw		
1	0	Display	Draw	Manual mode (erase) <sup>4</sup>	
1	1	Display and erase/write	Draw	Manual mode (change) <sup>4</sup>	
		Draw	Display	Manual mode (erase) <sup>5</sup>	60 frames/sec
1	0	Draw	Display		
0	0	Draw	Display and erase/write		
		Display and erase/write	Draw		
		Draw	Display and erase/write		

**Notes:**

<sup>1</sup>Value written to register immediately after the V-blank OUT interrupt.

<sup>2</sup>Changes from the first of the field.

<sup>3</sup>Changes from the 1-cycle mode to the manual mode with change.

<sup>4</sup>Be sure to continue to specify erase then change.

<sup>5</sup>Specify erase in the field immediately before changing to the 1-cycle mode.

Table 4.3 (b) Example of Use of Frame Buffer Change Mode (VBE Is Used)

Write Timing	Value written to register during write-enabled period			FB0	FB1	FB Change Mode
	VBE	FCM	FCT			
CPU processing	0	0	0	Draw	Display+Erase/Write	1 cycle mode
	0	0	0	Display+Erase/Write	Draw	60 frames/second
	1	1	1	Draw	Display+Erase/Write	Manual (change)
	1	1	1	Display	Draw	Changes from 1 cycle mode to manual mode with "change"
	1	1	1	Display	Draw	Manual (erase & change)
	0	0	0	Draw	Display	Cancel V-Blank erase
	1	1	1	Draw	Display	Desired frames/second
	1	1	1	Draw	Display	Manual (erase & change)
	0	0	0	Display	Draw	Cancel V-Blank erase
	1	0	0	Draw	Display	Manual (erase)
Write immediately after interrupt	0	0	0	Draw	Display+Erase/Write	1 cycle mode
	0	0	0	Display+Erase/Write	Draw	60 frames/second
Avoid access disable without waiting for interrupt and write	0	0	0	Display	Draw	Manual (change)
	0	0	0	Draw	Display	Cancel V-Blank erase

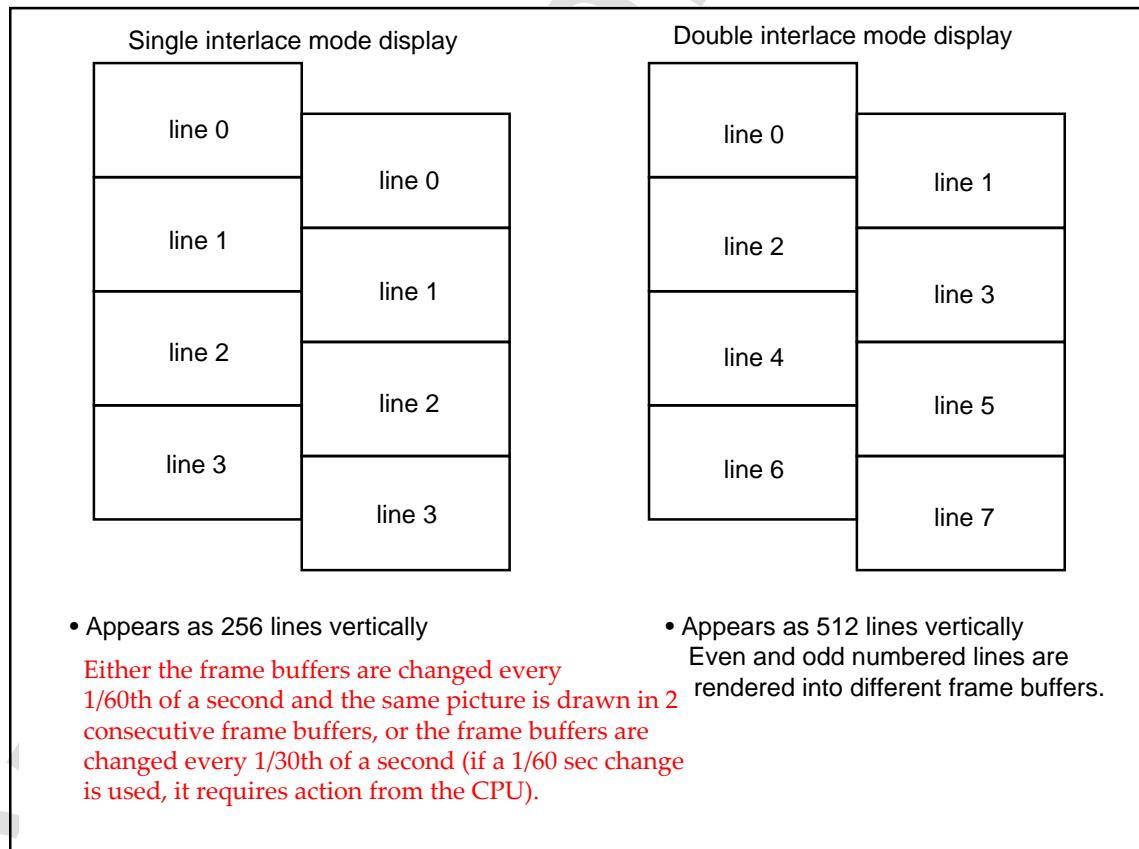
Note: //// Indicates write enabled



**Double Interlace Enable (DIE): bit 3****Double Interlace Draw Line (DIL): bit 2**

- In single interlace, the same picture is displayed in the even fields and the odd fields. In double interlace, however, the vertical resolution is doubled by displaying different pictures (each draws only even lines and odd lines, respectively) in the even fields and the odd fields.
- Double interlace is enabled by DIE = 1.
- The contents of the first screen displayed after changing DIE cannot be guaranteed.
- In double interlace, even and odd fields are rendered in different frame buffers, and therefore FCM = FCT = 0 (1-cycle mode) must be set.**

DIE	DIL	Interlace mode	Plot after next frame change
0	0	Non-interlace/single interlace	Plot both even and odd lines
0	1	Setting prohibited	—
1	0	Double interlace	Plot even-numbered lines only
1	1	Double interlace	Plot odd-numbered lines only

**Figure 4.1 Single Interlace and Double Interlace Display**

See table on Page 165 (page 181 PDF) for interaction between VDP1 and VDP2 interlace settings.

#### **Even/Odd Coordinate Select Bit (EOS): bit 4**

- When “1” is specified for high speed shrink (HSS) with a scaled or distorted sprite, this bit is enabled. When HSS = 1 is specified, lines drawn with a magnification ratio of less than 1 are drawn by sampling only the even or odd pixels of the original picture data. This bit specifies whether even or odd coordinates are sampled.
- When EOS = 0, only pixels at even coordinates are sampled. When EOS = 1, only pixels at odd coordinates are sampled.
- If HSS = 0, this bit is not referenced.
- Refer to “High Speed Shrink” under “6.3 CMDPMOD (Draw Mode Word)” for more information.

EOS	Even/odd coordinate select bit
0	Samples only pixels at even coordinates
1	Samples only pixels at odd coordinates



## 4.3 Plot Trigger Register

The plot trigger mode register (PTMR) controls the start of drawing. It is a 16-bit write-only register at address 100004H. Its value is reset to 00B after powering on or resetting. Set unused bits to “0”.

PTMR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100004H		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PTM
Write-only																	

### Plot Trigger Mode (PTM): bits 1, 0

- Controls the start of drawing.
- When the plot trigger mode bits are 10B, drawing begins automatically at the start of the frame.
- When the plot trigger mode bits are made 01B, drawing begins when the register is written. When 01B is written to the plot trigger mode bits during drawing, drawing begins from the top of the command table.
- When the plot trigger mode bits are 00B, drawing does not start even at the start of the frame. An idling condition is set.
- The value of the bits is reset to 00B after powering on or resetting.

PTM		Drawing mode
Bit 1	0	
0	0	Idle at frame change
0	1	Starts drawing when 01B is written
1	0	Starts drawing automatically with frame change
1	1	Setting prohibited (do not set)

- When the plot trigger mode bits are rewritten from 01B to 00B, drawing becomes valid from the next frame. However, when the plot trigger bits are rewritten from 00B or 10B to 01B, drawing becomes valid at that point and drawing is started even if drawing is automatically started by 10B.
- When the table is not rewritten the same drawing is performed, and therefore the results of color calculation of half-transparency change. Use the following procedure to change only the draw start mode without drawing.
  - 1) Change the plot trigger mode bits from 10B to 00B.
  - 2) Change the plot trigger mode bits from 00B to 01B in the next frame.

## 4.4 Erase/Write

Before rendering data into a frame buffer, VDP1 can erase the contents of that frame buffer when it is displayed. This erasure is referred to as erase/write, and it specifies the fill area to be erased and the fill data to be written to that area. The three registers related to erase/write are the erase/write data register, the erase/write left coordinate register and the erase/write right coordinate register.

### Erase/Write Data Register

The erase/write data register (EWDR) specifies the fill data during erase/write. It is a 16-bit write-only register at address 100006H. Its value becomes undefined after powering on or resetting, and therefore the fill data must be set.

EWDR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Erase/write data																	
Erase/write data for even X coordinates   Erase/write data for odd X coordinates																	
16 bits/pixel   8 bits/pixel   Write-only																	

### Erase/Write Data: bits 15~0

VDP1 can erase automatically the display frame buffer while writing parts to the draw frame buffer. At the time of this erase/write, the frame buffer is filled with the 16-bit data set in the erase/write register. Erase/write is performed 2 pixels at a time when the frame buffer depth is 8 bits/pixel. The area of erase/write is set by the erase/write left coordinate register and the erase/write right coordinate register.



## Erase/Write Upper-Left Coordinate Register

The erase / write left register (EWLR) sets the upper-left coordinate of the erase / write area. It is a 16-bit write-only register at address 100008H. Its value becomes undefined after powering on or resetting, and therefore the coordinates must be set. Set unused bits to "0."

EWLR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100008H		0	Upper-left coordinate X1						Upper-left coordinate Y1								
Write-only																	

**Erase/Write Upper-Left Coordinate X1:** bits 14~9

**Erase/Write Upper-Left Coordinate Y1:** bits 8~0

## Erase/Write Lower-Right Coordinate Register

The erase / write right coordinate register (EWRR) sets the lower-right coordinate of the erase / write area. It is a 16-bit write-only register at address 10000AH. Its value becomes undefined after powering on or resetting, and therefore the coordinates must be set. Set unused bits to "0".

EWRR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10000AH		Lower-right coordinate X3						Lower-right coordinate Y3									
Write-only																	

**Erase/Write Lower-Right Coordinate X3:** bits 15~9

**Erase/Write Lower-Right Coordinate Y3:** bits 8~0

- These registers specify the fill area in the frame buffer during erase / write.
- The X coordinate is set in 8-pixel units when there are 16 bits / pixel (normal, rotation 16, HDTV) and in 16-pixel units when there are 8 bits / pixel (high resolution, rotation 8). When the value of the system register is 1, it becomes 8 or 16. The actual lower-right X coordinate takes a value that is 8 or 16 times the register setting and from which 1 is subtracted.
- The Y coordinate is set in one-line units. Because the register setting for the Y coordinate is doubled during double interlace, the actual coordinate value should be set to one half. For example, when the setting is 223, the coordinate becomes 447.
- The actual X coordinate is expressed by the following equations.

16 bits / pixel

$$\text{Upper-left coordinate X1} = \text{register setting} \times 8$$

$$\text{Lower-right coordinate X3} = \text{register setting} \times 8 - 1$$

8 bits / pixel

$$\text{Upper-left coordinate X1} = \text{register setting} \times 16$$

$$\text{Lower-right coordinate X3} = \text{register setting} \times 16 - 1$$

Therefore the values that the upper-left coordinate X1 and the lower-right coordinate X3 can take are as shown in the following table.

Value set in register	16 bits/pixel		8 bits/pixel			
			High Resolution		Rotated 8	
	Upper left coordinate X1	Lower right coordinate X3	Upper left coordinate X1	Lower right coordinate X3	Upper left coordinate X1	Lower right coordinate X3
0	0	Setting prohibited	0	Setting prohibited	0	Setting prohibited
1	8	7	16	15	16	15
2	16	15	32	31	32	31
:	:	:	:	:	:	:
31	248	247	496	495	496	495
32	256	255	512	511	Setting prohibited	511
33	264	263	528	527	Setting prohibited	Setting prohibited
:	:	:	:	:	:	:
40	320	319	640	639	:	:
:	:	:	:	:	:	:
43	344	343	688	687	:	:
44	352	351	704	703	:	:
:	:	:	:	:	:	:
62	496	495	992	991	:	:
63	504	503	1008	1007	:	:
64	Setting prohibited	511	Setting prohibited	1023	:	:
over 65	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited

- The coordinates for erase / write are not checked, and therefore the registers must be set in the CPU in advance so that  $X1 < X3$  and  $Y1 \leq Y3$ .
- Set the erase / write range in each TV mode within their respective memory map ranges.

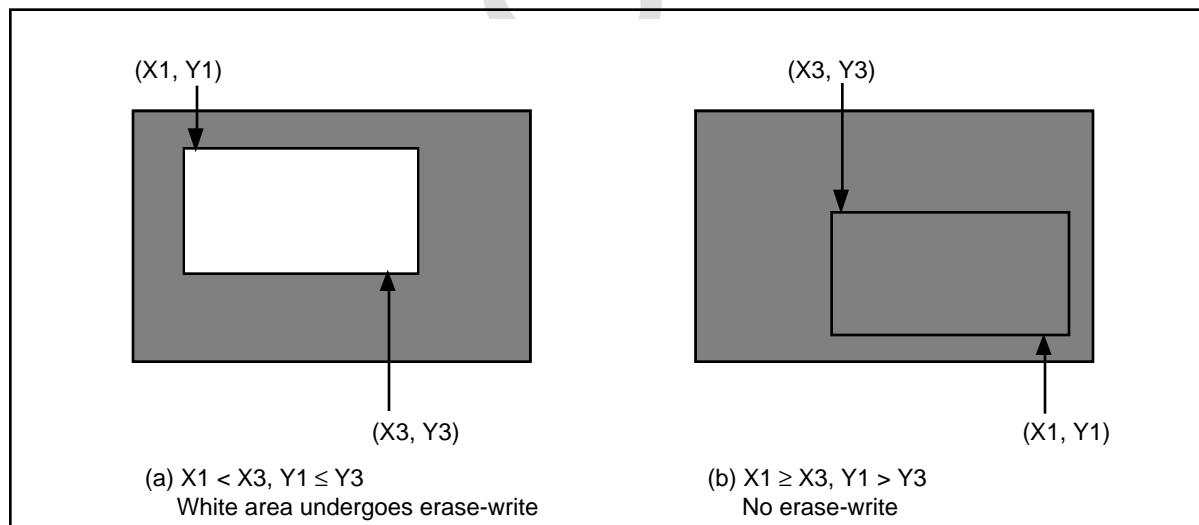


Figure 4.2 Erase/Write Area



- If the setting is  $X1 \geq X3$  or  $Y1 > Y3$ , then erase / write is performed for 1 dot in the normal or high-resolution mode and for 8 dots in the case of rotation or HDTV. In these cases, erase / write is performed under the assumption that the area  $(X1, Y1)$  is set to  $(X3 = X1 + 1, Y3 = Y1)$ .
  - When  $VBE = 0$  in the normal or high-resolution mode, erase / write is performed beyond the frame buffer during the display period. When setting the  $X3$  coordinate beyond the display screen, after the end of the effective data (fall of HTIM), erase / write is continued for 4 pixels in the normal mode and for 8 pixels in the high-resolution mode, and erase / write is not performed beyond the display screen other than in those areas.
  - When the erase / write area is set within the display screen area, no erase / write is performed outside the display area.
  - Erase / write is not affected by clipping.
  - Erase / write cannot be performed on the frame buffer during display when in the HDTV mode, when the frame buffer is enlarged or reduced, or when display is rotated. Because erase / write is performed in the vertical blanking period (V-BLANK), there is not enough time to erase / write the entire screen. To perform erase / write on the entire screen, the non-erased areas must be filled with polygons at the start of drawing.
  - **The number of pixels required for V-blank erase in a 16 bpp frame buffer is expressed by  $(X3 - X1) \times (Y3 - Y1 + 1) \times 8$**   
If this is within the number of pixels that can be used in V-blank erase, then erase / write is completed.
  - **The number of pixels that can be used in V-blank erase in a 16 bpp frame buffer is given by**  $\{(number\ of\ pixels\ in\ 1\ raster) - 200\}$   
 $\times \{(number\ of\ rasters\ in\ 1\ field) - (number\ of\ display\ rasters)\}$
- The respective values are shown in Tables 4.4 and 4.5.

**Table 4.4 Number of Rasters and Number of Pixels**

Screen mode	Number of horizontal pixels	Number of pixels in 1 raster	Number of rasters in 1 field
NTSC	320	1708	263
PAL	352	1820	313
31KC	—	852	525
HDTV	—	848	562

**Table 4.5 Number of Pixels that Can Be Used in V-Blank Erase in a 16 bpp frame buffer (in Non-Interlace Display)**

Screen mode	Resolution (horizontal x vertical)	Number of pixels that can be used
NTSC	320 x 224	58812
	320 x 240	34684
	352 x 224	63180
	352 x 240	37260
PAL	320 x 224	134212
	320 x 240	110084
	320 x 256	85956
	352 x 224	144180
	352 x 240	118260
	352 x 256	92340
31KC	320 x 480	29340
HDTV	352 x 480	53136



## 4.5 Draw Forced Termination Register

The draw forced termination register (ENDR) forces termination of drawing. It is a write-only 16-bit register at address 10000CH.

ENDR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10000CH		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Write-only

### Draw Forced termination: draw end

- Forces termination of the drawing currently being processed.
- Forces termination of the drawing in the frame within approximately 30 clock cycles after the data is written to the register.
- Specify 0000H for the write data.
- Interrupted drawing cannot be resumed.
- When the amount of data drawn is large and cannot be drawn in one frame, a pseudo draw continuation is used to divide the data into two parts and draw it. This draw forced termination is used to terminate drawing during this pseudo draw continuation. For more information, refer to section 4.8 “Current Operation Command Address Register.”

## 4.6 Transfer End Status Register

The transfer end status register (EDSR) indicates the end status of the prior frame processing. It is a 16-bit read-only register at address 100010H. Set unused bits to "0."

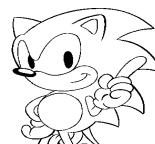
EDSR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100010H		0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEF	BEF
Read-only																	

### Current End Bit Fetch Status (CEF): bit 1

This register indicates whether or not the end bit (draw forced termination command) has been fetched from the command table in the frame currently being drawn. When it is "0," the end bit indicates a non-fetched status; when it is "1," it indicates that the end bit has been fetched and that drawing is terminated.

CEF	End bit fetch status
0	The end bit in current frame has not been fetched.
1	The end bit in current frame has been fetched and plotting is ended.

- The VDP1 successively fetches the following command tables in VRAM and draws them in the frame buffer. When the draw end command (when end bit is 1) is fetched, the drawing of one frame is terminated. CEF is set to "1" at this time.
- When data are transferred from the CPU to VRAM while in a draw end (CEF = 1) status, VRAM can be accessed without the overhead for stopping drawing and without causing the CPU to wait.
- When the draw end command is fetched, the VDP1 sets CEF to "1" and generates an interrupt signal.
- There are two methods of judging termination of drawing: one confirms the fetch status of the end bit with CEF (polling) and the other uses the interrupt signal.
- When there is no draw end command in VRAM, or when there is one and it is defined by the jump mode such that it cannot be fetched, this bit remains "0."
- This bit is reset to "0" when the frame buffers are changed or when drawing is started.
- If fetch of the draw terminate command matches when the frame buffer changes, CEF and BEF might not become "1."



### **Before End Bit Fetch Status (BEF): bit 0**

This register indicates whether or not the end bit (draw terminate command) has been fetched from the command table in the previous frame. When it is "0," it indicates that the end bit has not been fetched; when "1," it indicates that the end bit has been fetched and that drawing is terminated.

BEF	End bit fetch status
0	The end bit in previous frame has not been fetched.
1	The end bit in previous frame has been fetched and drawing is terminated.

- The VDP1 successively fetches the following command tables in VRAM and draws them in the frame buffer. When the draw terminate command (when end bit is 1) is fetched, the drawing of one frame is terminated. If there are many commands, or if there are many pixels to be drawn because of enlargement, drawing may not be terminated in one frame. This is referred to as "transfer-over." This bit indicates a transfer-over status.
- When transfer-over has occurred, it is necessary to reduce the drawing commands or to reduce the pixels drawn.
- When there is no draw terminate command in VRAM, or when there is one and it is defined by a jump mode such that it cannot be fetched, this bit remains at "0."
- This bit is written with the value of the CEF value when the frame buffer is changed or at the start of drawing, and is maintained until the next frame buffer change.

## 4.7 Last Operation Command Address Register

The last operation command address register (LOPR) indicates the command table address processed at the end of the previous frame. It is a 16-bit read-only register at address 100012H.

LOPR bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100012H	Last operation command address/8H												0	0		
Read-only																

### Last Operation Command Address: bits 15~0

- When the frame buffer is changed, the value resulting from dividing the address of the command table read to the VDP1 from the VRAM by 8H is written to this register.
- This register is updated when the frame buffer is changed, so it is possible to know the address of the command table last processed in the previous frame.
- Because the boundary of the table address is 20H bytes, the lower 2 bits of the register are fixed at 00B.

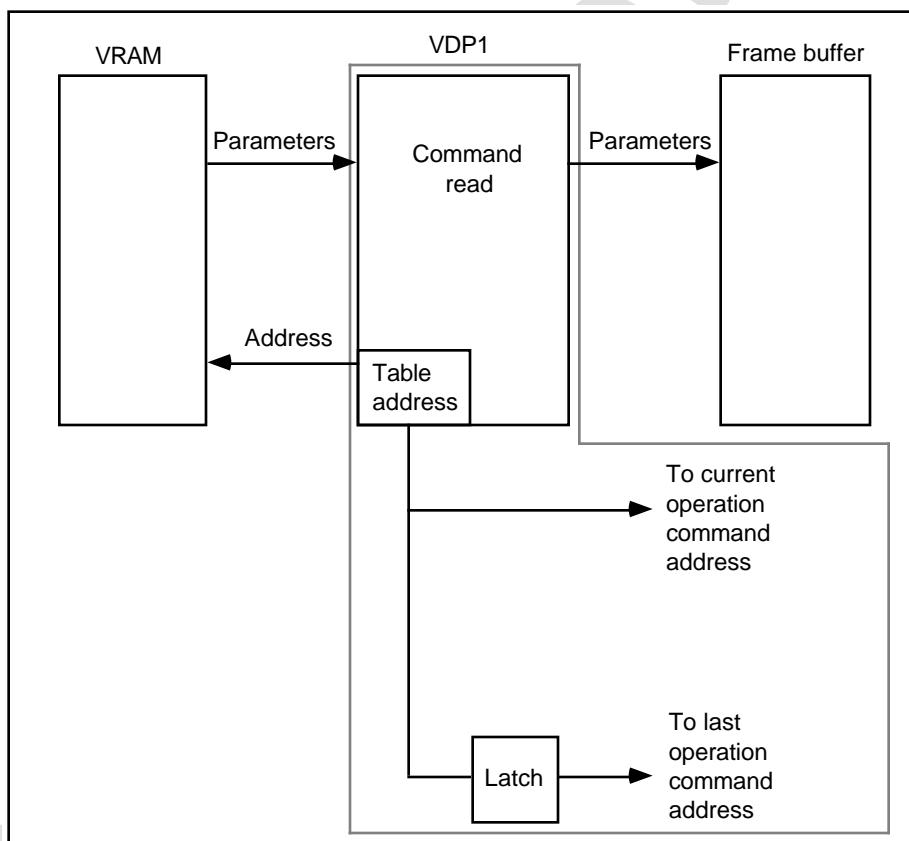


Figure 4.3 Last Operation Command and Current Operation Command Address



## 4.8 Current Operation Command Address Register

The current operation command address register (COPR) indicates the address of the command table being processed. It is a 16-bit read-only register at address 100014H.

COPR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100014H		Current operation command address/8H														0	0
Read-only																	

### Current Operation Command Address: bits 15~0

- The value resulting from dividing the command table address, which has received the parameter currently being processed, by 8H is written to this register.
- It is possible to know the address of the command table currently being processed. The address value is continually updated during command processing.
- When the draw end command is fetched and drawing is abnormally ended, the value of the address updated at that time is retained as is until drawing is started by a frame change or plot trigger.
- When the draw end command is fetched, this is the address (divided by 8H) of the draw end command table. In the case of draw forced termination, this is the address (divided by 8H) of the abnormally ended command table.
- Because the boundary of the table address is 20H bytes, the lower 2 bits of the register are fixed at 00B.

### Pseudo Draw Continuation

- When the amount of data drawn is large and cannot be drawn in one frame, drawing is terminated part way through and pseudo draw continuation is used to divide the data into two parts and to draw it.
- Pseudo draw continuation is performed using the following procedure. Drawing is terminated, at which time the top command is written so that it jumps to the address of the table whose processing was terminated at that time, and the plot trigger mode is set to start drawing ( $PTM = 01B$ ) when it is written.
- As much drawing is performed as possible in the time of one frame, at which point drawing is terminated (interrupted) by the CPU. Next, the CPU writes the jump destination in the top command in VRAM to the command table address at the time of termination, and drawing is immediately started by the plot trigger mode ( $PTM = 01B$ ). This is how pseudo draw continuation is done. However, drawing cannot be continued when the command table address is in the subroutine at the time of a forced termination. In this case, the drawing times necessary to transfer the respective pixel data to the frame buffer are set shorter than the time to the end of the frames.
- When color calculation of half-transparent is being performed at the time of a forced termination of drawing, it is possible that dots may occur for which half-transparent processing is performed twice (color calculation is performed twice) when drawing is continued using the above method. A forced termination of drawing must be performed when color calculation of half-transparency is not performed.



## 4.9 Mode Status Register

The mode register (MODR) indicates the setting of the write-only register. It is a 16-bit read-only register at address 100016H.

MODR	bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100016H		VER	—	—	—	PTM1	EOS	DIE	DIL	FCM	VBE	—	—	—	—	TVM
Read-only																

Because the registers at addresses from 100000H to 10000CH are write-only, they cannot be read to confirm the settings. The settings in write-only registers can be confirmed by this register. It is mainly used as help information during program development. However, because these values are the actual system register settings, they may be different from the values taken in as internal signals.

### **Version Number (VER): bits 15~12**

Indicates the version number of VDP1. The value is "1" (0001B).

### **Plot trigger Mode (PTM1): bit 8**

Setting of bit 1 of the plot trigger register (PTMR: 100004H).

### **Even/Odd Coordinate Select Bit (EOS): bit 7**

Setting of bit 4 of the frame buffer change mode register (FBCR: 100002H).

### **Double Interlace Enable Bit (DIE): bit 6**

Setting of bit 3 of the frame buffer change mode register (FBCR: 100002H).

### **Double Interlace Draw Line (DIL): bit 5**

Setting of bit 2 of the frame buffer change mode register (FBCR: 100002H).

### **Frame Buffer Change Mode Bit (FCM): bit 4**

Setting of bit 1 of the frame buffer change mode register (FBCR: 100002H).

**V-Blank Erase/Write Enable Bit (VBE): bit 3**

Setting of bit 3 of the TV mode selection register (TV<sup>MR</sup>: 100000H).

**TV Mode Selection Bits (TVM): bit 2~0**

Setting of bits 2 through 0 of the TV mode selection register (TV<sup>MR</sup>: 100000H).



## Chapter 5

### Tables

#### Contents

5.1	Character Pattern Tables .....	60
5.2	Color Lookup Tables .....	62
5.3	Gouraud Shading Table .....	64
5.4	Command Tables .....	66

## 5.1 Character Pattern Tables

A character pattern is data that becomes the basis for a sprite drawn by a texture draw command.

Define character patterns continuously to VRAM as tables. The stored data are referenced at the character address of the command table, and the data size is determined by the size of the character pattern and the color mode of the sprite.

### Character Pattern Table Addresses

Define character pattern boundaries with a 20H (32)-byte boundary. However, character patterns starting from address 00000H in VRAM cannot be defined. Character patterns are stored in a 20H-byte boundary, so that part of the 20H bytes not filled becomes free space. The address of VRAM is 7FFFFH. Do not define character patterns beyond address 80000H.

### Table Size

Depending on the color mode, 1 pixel of the character pattern becomes 4-, 8- or 16-bit data. The character size can be specified from 8 pixels to 504 pixels horizontally in 8-pixel units and from 1 pixel to 255 pixels vertically in 1-pixel units.

For example, in order to represent a character pattern 8 horizontal pixels by 3 vertical pixels, 0CH (12) bytes is required for 4 bits/pixel, 18H (24) bytes is required for 8 bits/pixel; and 30H (48) bytes is required for 16 bits/pixel. The table of a character pattern requires 4H bytes when the character pattern is 8 horizontal pixels by 1 vertical pixel and 4 bits/pixel, and the maximum data of a character pattern is 3EC10H (257,040) bytes when the character pattern is 504 horizontal pixels by 255 vertical pixels and 16 bits/pixel.

**Table 5.1 Size of Character Pattern Tables**

Item	Minimum	Maximum
Character size	8 horizontal pixels x 1 vertical pixel	504 horizontal pixels x 255 vertical pixels
Color mode	4 bits/pixel	16 bits/pixel
Character pattern	4H bytes *	3EC10H (257,040) bytes

\*Even a 4H-byte character requires 20H bytes in VRAM, so use caution.



### Examples of Character Pattern Tables

The examples of character patterns in Figure 5.1 are shown with a character size of 8 horizontal pixels by 3 vertical pixels:

---

For 4 bits/pixel, 0CH (12) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+0	+1	+2	+3	+4	+5	+6	+7
+04H	+4	+5	+6	+7				
+08H	+8	+9	+A	+B				

Value is relative address from character pattern address

For 8 bits/pixel, 18H (24) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+0	+1	+2	+3	+4	+5	+6	+7
+08H	+8	+9	+A	+B	+C	+D	+E	+F
+10H	+10	+11	+12	+13	+14	+15	+16	+17

For 16 bits/pixel, 30H (48) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+0	+1	+2	+3	+4	+5	+6	+7
+10H	+10	+11	+12	+13	+14	+15	+16	+17
+20H	+20	+21	+22	+23	+24	+25	+26	+27

---

Figure 5.1 Examples of Character Pattern Tables

## 5.2 Color Lookup Tables

The color lookup table is used to specify the color of the pixels of the character pattern in the lookup table mode. The color lookup table defines the respective color codes of 16 colors in VRAM as 16-bit data. In the lookup table mode, character patterns are defined in the character pattern table in 4 bits/pixel, and 1 color of the 16 colors defined as 4-bit data in the color lookup table is selected. The 16 bits of the color code of the selected color are written to the frame buffer, as is as the color code of the pixel.

The size of the color lookup table is 20H (32) bytes. The table should be written from the boundary addresses of 20H-byte units in VRAM. However, 00000H cannot be defined. VRAM occupies up to address 7FFFFH. Do not define color lookup tables beyond address 80000H.

If the frame buffer can use RGB codes (see Sprite color mode bit, p. 207 of VDP2 User's Manual), RGB codes MSB must be set to 1 and color bank code MSB must be set to 0. If the frame buffer is all in palette format, set MSB according to its use by VDP2 (see Sprite types, p. 200 of VDP2 User's Manual).

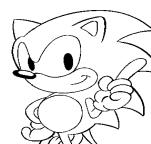
+00H	16-bit data	(color code of 0H)
+02H	16-bit data	(color code of 1H)
+04H	16-bit data	(color code of 2H)
:	:	
⋮	⋮	
+1CH	16-bit data	(color code of EH)
+1EH	16-bit data	(color code of FH)

Figure 5.2 Color Lookup Table

### Lookup Table Mode

The color mode is set to the lookup table mode by the sprite draw command. According to this specification, the character patterns stored in VRAM with 4 bits/pixel are converted to color codes by referencing the specified color lookup table, and are written to the frame buffer.

The storage address and size of the character pattern and the storage address in the color lookup table are specified by the sprite draw command.



## Character Patterns

In the lookup table mode a character pattern is 4 bits / pixel and is stored in VRAM.

## Command Tables

The address of the color lookup table referenced by the sprite is specified in the color lookup table address (top address + 06H) of the command table. The specified value is address / 8H. Since the color lookup table is stored in boundaries of 20H-byte units, the lower two bits become 00B.

The relationship between the command table, color lookup table and character pattern table is shown below.

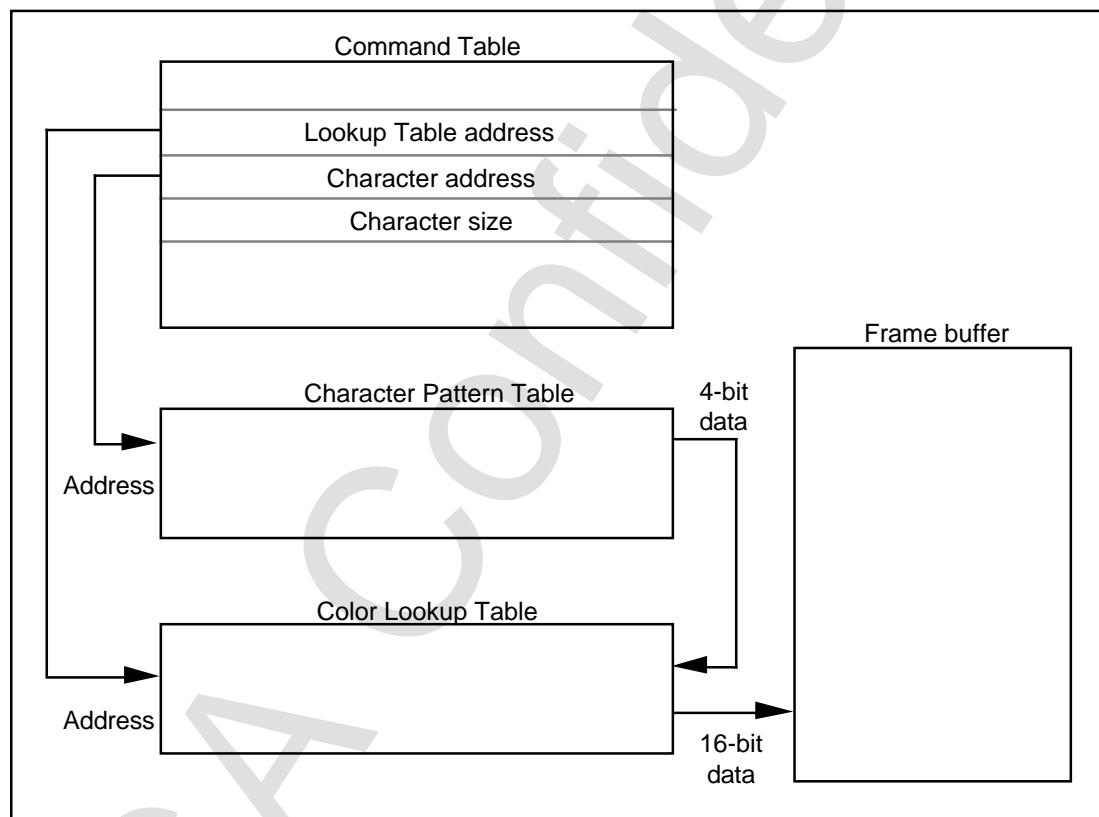


Figure 5.3 Relationship between Tables in Lookup Table System

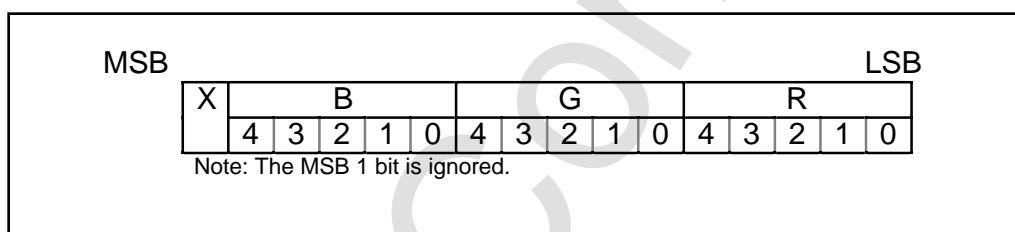
## 5.3 Gouraud Shading Table

This table specifies RGB data for four points when processing Gouraud shading for parts. The data for each of the four points is 16 bit, so 8 bytes are required for one table. The table is positioned where an 8H-byte boundary address begins, but do not write from 00000H to 0001FH. VRAM occupies up to address 7FFFFH. Do not define Gouraud shading tables beyond 80000H.

The table defines RGB data for vertices (A), (B), (C), and (D), in that order. In the case of lines, only vertices (A) and (B) are valid and correspond to the start and end of the line. In the case of sprites, vertices (A), (B), (C), and (D) correspond to the upper-left, upper-right, lower-right, and lower-left. This table is referenced when Gouraud shading processing is specified.

**Table 5.2 Gouraud Shading Table**

Table address	Corresponding Vertices	
	Sprites, polygons, polylines	Lines
Table top address	Vertex (A)	Line start point
Table top address + 2	Vertex (B)	Line end point
Table top address + 4	Vertex (C)	Ignored
Table top address + 6	Vertex (D)	Ignored



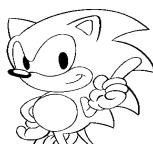
**Figure 5.4** RGB Code Format

# Gouraud Shading

Gouraud shading can be performed on parts drawn in RGB code. Gouraud shading specifies the amount of change in the luminance of each of R, G, and B, which are changed in RGB code parts in a Gouraud shading table. It is only effective on RGB color codes. The color cannot be guaranteed when Gouraud shading is specified for color bank color codes.

## Gouraud Shading Specification

Gouraud shading is specified with color calculation bits. The color calculation bits are at bits 2~0 of the draw mode word at the top address + 04H of the command table. When Gouraud shading is specified, the address of the Gouraud shading table is specified. The top address/8H of the Gouraud shading table is positioned at the top address + 1CH of the command table.



## Gouraud Shading Processing

This specifies the data for the amount of change in R, G, and B for the four points (two points in the case of lines) of the part in the Gouraud shading table. The data interpolated for each of R, G, and B between the four points are added to the original color of the part. Because each of the values of R, G, and B takes the values 00H to 1FH, the result of subtracting 10H from the complementary RGB data is added to the original color of the part. For example, if the value of RGB is 10H, the original color is left as is; if the value is 00H, the original color becomes -10H; and if the value is 1FH, then the original color becomes +0FH. If the value after color calculation becomes less than 00H, then 00H is used; if it is larger than 1FH, then 1FH is used.

The relationship between Gouraud shading table settings and correction values is shown as follows.

**Table 5.3 Relationship between Gouraud Shading Table Settings and Correction Values**

Table setting	Correction for original data	Table setting	Correction for original data
00H	-10H	10H	0
01H	-0FH	11H	+01H
02H	-0EH	12H	+02H
03H	-0DH	13H	+03H
04H	-0CH	14H	+04H
05H	-0BH	15H	+05H
06H	-0AH	16H	+06H
07H	-09H	17H	+07H
08H	-08H	18H	+08H
09H	-07H	19H	+09H
0AH	-06H	1AH	+0AH
0BH	-05H	1BH	+0BH
0CH	-04H	1CH	+0CH
0DH	-03H	1DH	+0DH
0EH	-02H	1EH	+0EH
0FH	-01H	1FH	+0FH

Real Gouraud shading changes only the luminance, but in this system it changes each of R, G, and B, and therefore in some cases the hue also changes. To avoid changing the hue, define the same value for each RGB for one point defined in the Gouraud shading table. By this means, white Gouraud shading is applied.

Gouraud shading is performed on non-textured colors in the case of lines, polylines, and polygons and on colors referenced by the character pattern data or the color lookup table in the case of sprites.

## 5.4 Command Tables

Command tables comprise 1EH (30) bytes. Because command tables are fetched every 20H (32) bytes, they should be defined with 20H boundaries. The 2 bytes following command tables are dummy bytes, and are skipped when the command table is fetched.

Command tables stored in VRAM are fetched from the top address (00000H) every frame. A command table must always be stored at address 00000H to 0001EH. Drawing operation cannot be guaranteed when other than a command table (color lookup table, Gouraud shading table, or character pattern table) is stored there. VRAM occupies up to 7FFFFH. Do not define Gouraud shading tables beyond address 80000H.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
CMDCTRL +00H	END	JP			ZP				0	0	Dir	Comm								
CMDLINK +02H	LINK specification/8H														0	0				
CMDPMOD +04H	MON	0	0	HSS	Pclp	Clip	Cmod	Mesh	ECD	SPD	Color mode			Color calculation bit						
CMDCOLR +06H	Color bank, color lookup table/8H (LSB is set to 00), non-textured color																			
CMDSRCRA +08H	Character address/8H														0	0				
CMDSIZE +0AH	0	0	Character size X/8						Character size Y											
CMDXA +0CH	Code extension						Point (A) X coordinate (XA)*													
CMDYA +0EH	Code extension						Point (A) Y coordinate (YA)													
CMDXB +10H	Code extension						Point (B) X coordinate (XB)													
CMDYB +12H	Code extension						Point (B) Y coordinate (YB)													
CMDXC +14H	Code extension						Point (C) X coordinate (XC)													
CMDYC +16H	Code extension						Point (C) Y coordinate (YC)													
CMDXD +18H	Code extension						Point (D) X coordinate (XD)													
CMDYD +1AH	Code extension						Point (D) Y coordinate (YD)													
CMDGRDA +1CH	Gouraud Shading Table/8H																			
+1EH	(Dummy) Skipped during table fetch																			
+20H	Succeeding table																			
:																				
+40H	Succeeding table																			
:																				
+60H															:					

\* Note: The top bit of the vertex coordinate is a sign bit. A negative value is indicated by a complement of 2. Extend the sign to the upper 5 bits.

Figure 5.5 Command Table

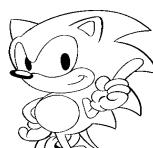


The order in which sprites and other parts are drawn is determined by how they are placed in the VRAM of this command table.

Drawn parts processed first are the farthest from the view point, and parts processed last are the closest to the view point.

(Page 68 is blank in the original Japanese document.)

SEGA Confidential



# Chapter 6

## Command Tables

### Contents

6.1	CMDCTRL (Control Words) .....	70
	Commands .....	71
	Jump Mode .....	72
	Zoom Point .....	73
	Character Read Direction .....	77
6.2	CMDLINK (Link Specification) .....	78
6.3	CMDPMOD (Draw Mode Word) .....	79
	High Speed Shrink .....	81
	Pre-Clipping Disable .....	83
	User Clipping Enable .....	84
	User Clipping Mode .....	84
	Mesh Enable .....	85
	End Code Disable .....	86
	Transparent Pixel Disable .....	88
	Color Mode .....	89
	Color Calculation.....	93
	MSB ON .....	97
6.4	CMDCOLR (Color Control Word) .....	98
	Color Bank .....	99
	Color Lookup Table .....	101
	Non-textured Color .....	102
6.5	CMDSRCA (Character Address).....	103
6.6	CMDSIZE (Character Size).....	104
6.7	CMDXA~CMDYD (Vertex Coordinate Data) .....	105
6.8	CMDGRDA (Gouraud Shading Table) .....	106

**Note:** Refer to “5.4 Command Tables” for more information on command tables.

## 6.1 CMDCTRL (Control Words)

CMDCTRL specifies commands and also controls command tables and specifies the inversion and zoom point of sprites. CMDCTRL is 16 bits at the top address + 00H of the command table, and its bit configuration is as follows. Set unused bits to "0."

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL +00H	END		JP		ZP			0	0		Dir		Comm			

### End Bit (END): bit 15

Indicates the draw terminate command. If there is no draw terminate command, the command selection bit becomes valid.

### Jump Select (JP): bits 14~12

Indicates the method by which the next command table to be read (fetched) is specified.

### Zoom Point (ZP): bits 11~8

Indicates the zoom point in the case of a scaled sprite draw command. The zoom point specifies the reference point for inversion and enlargement or reduction.

### Character Read Direction (Dir): bits 5, 4

Indicates the read direction from the character pattern table in the case of a texture draw command.

### Command Select (Comm): bits 3~0

Indicates the function of the command. When the end bit is a terminate command, command select is disabled.



## Commands

Commands are determined by the end bit (END, bit 15) of CMDCTRL and the command selection bit (Comm, bits 3~0). The commands set by the end bit and the command selection bit are shown in Table 6.1. The content of the command table in VRAM is determined by the command.

**Table 6.1 Commands**

END	Comm				Function	Commands
Bit 15	3	2	1	0		
0	0	0	0	0	Draw commands	Textured draw command Normal sprite draw command
		0	1	0		Scaled sprite draw command
		1	0	0		Distorted sprite draw command
		1	0	0		Non-textured draw command Polygon draw command
		0	1	0		Polyline draw command
	1	1	0	0	Register set commands	Line draw command
		0	0	0		Set command for user clipping coordinates
		0	0	1		Set command for system clipping coordinates
		1	0	0		Local coordinate set command
		1	0	0	Draw end command	
All other codes					Setting prohibited (do not use)	

## Jump Mode

The jump mode specifies the command table to be processed next and how to jump to the table. When the jump mode specifies the address and jumps to it, the address of the command table (prior to jumping) is set to a CMDLINK (top address + 02H of command table, 16 bits).

### Jump Mode (JP): top address + 00H of command table, bits 14~12

Specifies the jump mode for jumping to the next command table to be processed. When the jump mode is skipped (bit 14 = 1), reading of the command table is terminated there and the table is not processed. There is 1 level of nesting by a jump call. Do not use jump calls in subroutines.

JP			Jump mode	Processing
Bit 14	13	12		
0	0	0	Jump next	Automatically jumps to next table (address +20H) after this table is processed (CMDLINK is ignored).
0	0	1	Jump assign	Jumps to CMDLINK table after this table is processed.
0	1	0	Jump call	CMDLINK table receives subroutine call after this table is processed.
0	1	1	Jump return	Returns to main routine after this table is processed (CMDLINK is ignored).
1	0	0	Skip next	Jumps to next table (address +20H) without processing this table (CMDLINK is ignored).
1	0	1	Skip assign	Jumps to CMDLINK table without processing this table.
1	1	0	Skip call	CMDLINK table receives subroutine call without processing this table.
1	1	1	Skip return	Returns to main routine without processing this table (CMDLINK is ignored).

When jump assign, jump call, skip assign, or skip call is specified, the address of the table to be processed next is specified by CMDLINK.

When jump next, jump return, skip next, or skip return is specified, CMDLINK is ignored.



## Zoom Point

Zoom point specifies the zoom point of the character when a scaled sprite is drawn enlarged or reduced. It functions only with scaled sprites.

### Zoom Point (ZP): top address + 00H of command table, bits 11~8

Zoom point specifies the fixed point of the character when a scaled sprite is drawn enlarged or reduced. The relationship between the value of the zoom point and the fixed point is as follows. When the zoom point (ZP) is "0," there is no zoom point, and drawing is performed by specifying the upper-left coordinates and the lower-right coordinates.

Fix the zoom point to 0H for sprites other than scaled sprites.

ZP				Code	Zoom point
Bit 11	10	9	8		
0	0	0	0	0H	Specifies two coordinates
0	1	0	1	5H	Upper-left
0	1	1	0	6H	Upper-center
0	1	1	1	7H	Upper-right
1	0	0	1	9H	Center-left
1	0	1	0	AH	Center-center
1	0	1	1	BH	Center-right
1	1	0	1	DH	Lower-left
1	1	1	0	EH	Lower-center
1	1	1	1	FH	Lower-right
Other than above				Setting prohibited (do not set)	

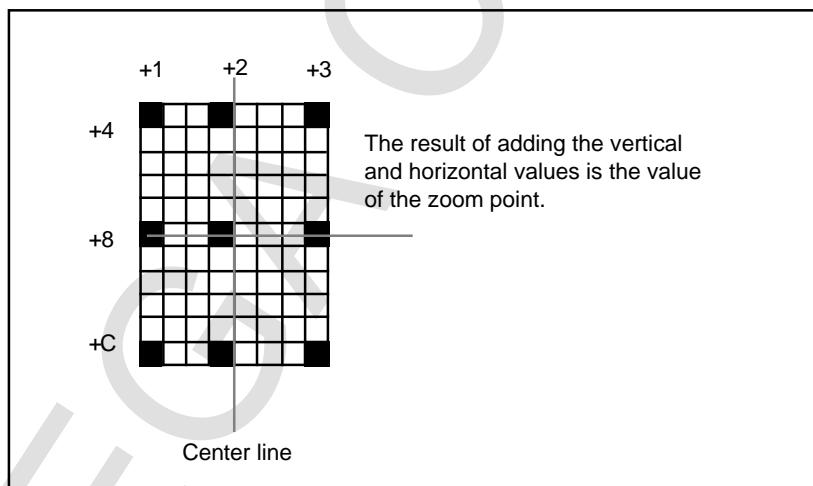


Figure 6.1 Zoom Point

When a value other than "0" is specified for the zoom point (ZP), the drawing position and drawing size of the sprite are determined by the zoom point (ZP), zoom point coordinates, and display width. The zoom point of a character whose zoom point is specified is drawn at the zoom point coordinates (CMDXA, CMDYA). The drawing size is determined by the display width (CMDXB, CMDYB).

When reduced, pixels are pulled out, and therefore the specified zoom point may disappear. Do not specify values other than those that have been established as the zoom point. Drawing cannot be guaranteed when values other than those that have been established are specified.

Because the character size in the X direction is a multiple of 8, the zoom point will not be exactly in the center. When the length in the Y direction is an even value, the zoom point will not be exactly in the center. When the length is an even number, the coordinates for the left side, right side, top, and bottom are calculated from point (A) and the display width, and therefore the position of the zoom point may shift.

The direction of drawing of the character is specified by the read direction of the character. A negative value cannot be specified for the display width. Drawing cannot be guaranteed when a negative value is specified for the display width.

### **Zoom Point Specification**

When ZP is other than 0H, the zoom point coordinates are specified by CMDXA and CMDYA and the display width is specified by CMDXB and CMDYB.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDXA +0CH	Code extension															Zoom point, X coordinate (XA)
CMDYA +0EH	Code extension															Zoom point, Y coordinate (YA)
CMDXB +10H	Code extension															Display, X width (XB)
CMDYB +12H	Code extension															Display, Y width (YB)

### **Specification of Coordinates for Two Points**

When ZP is 0H, coordinates for the upper-left and coordinates for the lower-right can be specified. Vertex (A) is specified by CMDXA and CMDYA, and vertex (C) is specified by CMDXC and CMDYC.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDXA +0CH	Code extension															Vertex (A), X coordinate (XA)
CMDYA +0EH	Code extension															Vertex (A), Y coordinate (YA)
+10H	:															:
+12H	:															:
CMDXC +14H	Code extension															Vertex (C), X coordinate (XC)
CMDYC +16H	Code extension															Vertex (C), Y coordinate (YC)



### Drawing Area

The zoom point coordinates are specified by CMDXA and CMDYA. The zoom point specified by the Zoom Point (ZP) is drawn at the zoom point coordinates. The zoom point becomes the reference point when zooming the character.

The display width is specified by CMDXB and CMDYB. The display width becomes the size drawn.

The area in which the character is drawn is determined by the zoom point bit, zoom point coordinates, and the display width. When the zoom point coordinate is (XA, YA) and the display width is (XB, YB), the drawing area is as shown in Figure 6.2.

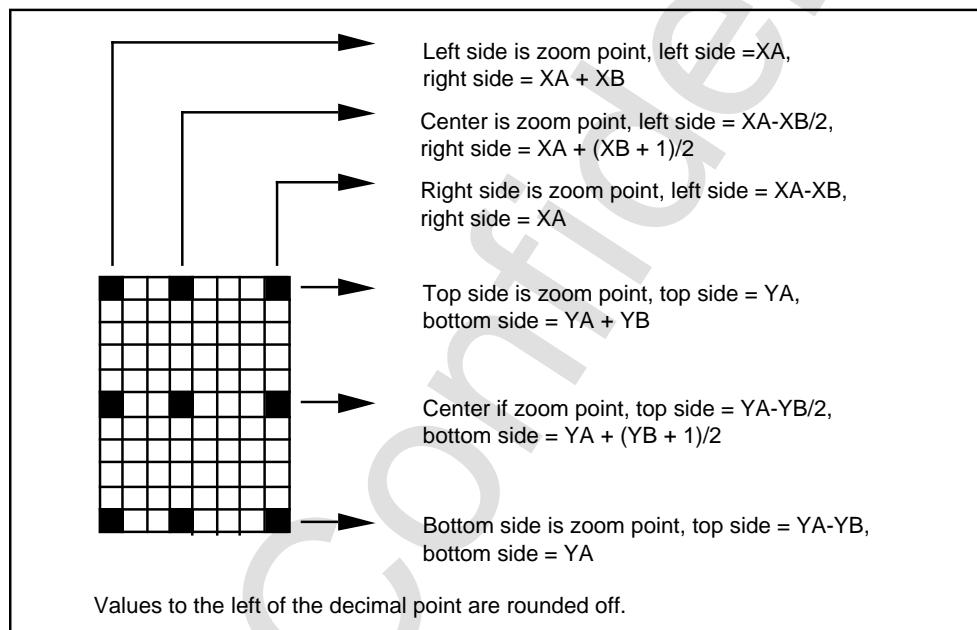


Figure 6.2 Drawing Area

### Zoom Point and Drawing Area

When the zoom point coordinates are (100, 50) and the display widths are (40, 30), the coordinates for each of the vertices are as follows. A sprite for which vertical or horizontal inversion is not specified is assumed. In this case, vertex (A) is (100, 50) and vertex (C) is (140, 80) when the zoom point (ZP) is 0H.

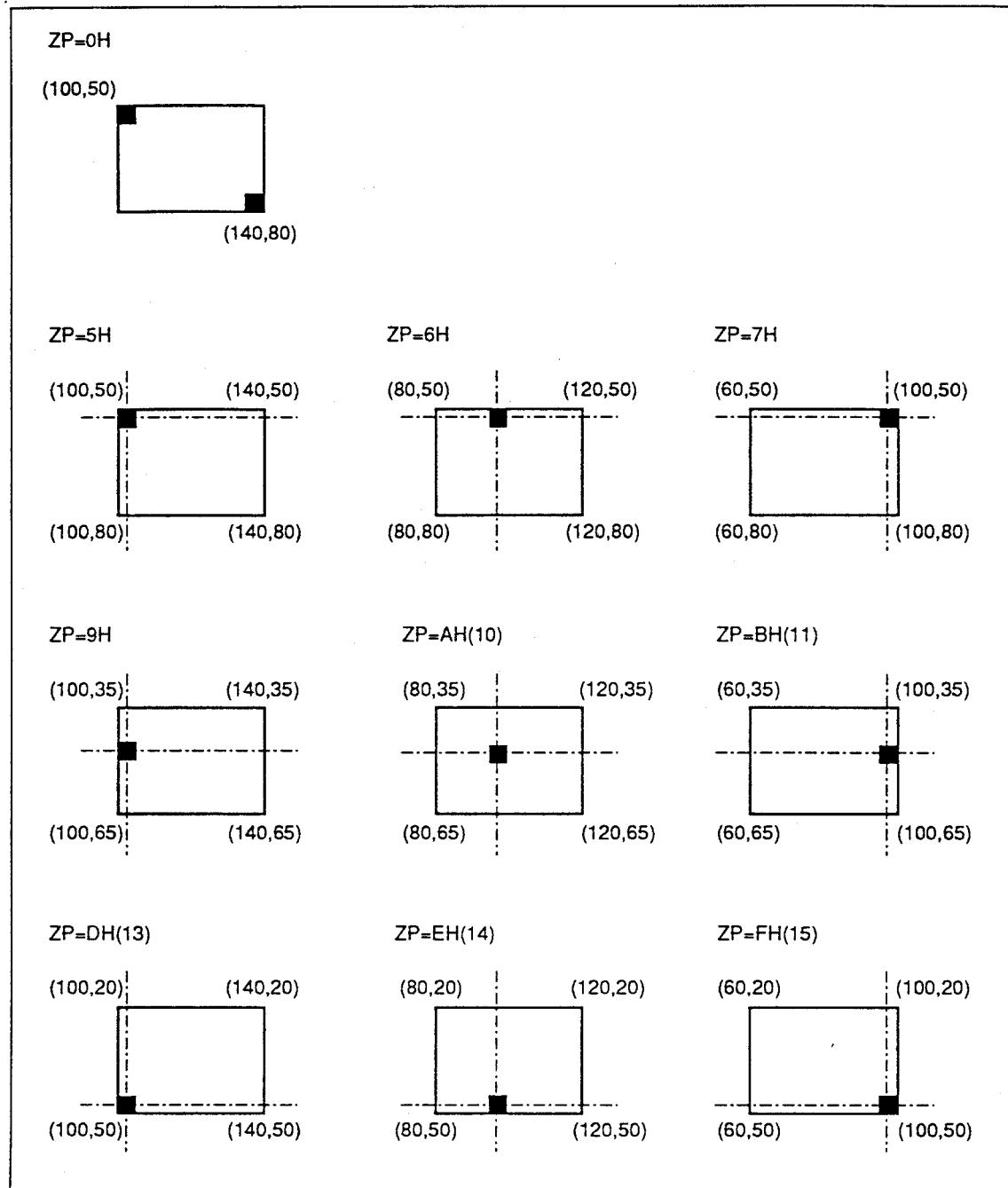


Figure 6.3 Zoom Point and Drawing Area



## Character Read Direction

The read direction of the character can be specified. This specification makes it possible to invert the character vertically and horizontally.

**Character Read Direction (Dir) Bits:** Command table start address + 00H, bits 5, 4

These bits specify the read direction of the character pattern. Vertical inversion, horizontal inversion, or simultaneous vertical and horizontal inversion can be specified.

When bit 5 is "0," the vertical (Y) direction is drawn as is, without inversion. When bit 5 is "1," the character pattern is inverted vertically.

When bit 4 is "0," the horizontal (X) axis is drawn as is, without inversion. When bit 4 is "1", then the character pattern is inverted horizontally.

When bits 4 and 5 are both "0", then the character pattern is drawn as is with no inversion. When bits 4 and 5 are both "1", then the character pattern is inverted both vertically and horizontally. Fix the character read direction to 00B for characters other than sprites.

Dir		Inversion processing
Y	X	
0	0	Not inverted
0	1	Inverted horizontally
1	0	Inverted vertically
1	1	Inverted vertically and horizontally

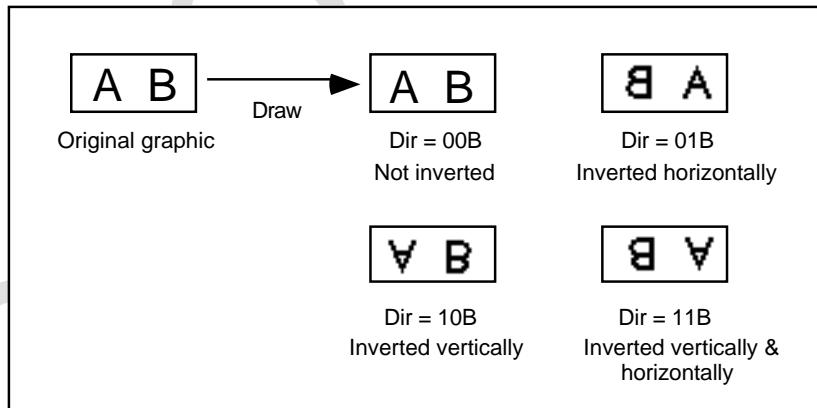


Figure 6.4 Character Read Direction

## 6.2 CMDLINK (Link Specification)

CMDLINK specifies the address of the command table to be processed next when assign or call is specified in the jump mode. The specification is made with the 16 bits at the top address + 02H of the command table.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDLINK +02H															0	0

### CMDLINK: Top Address + 02H of Command Table

When the command table being processed is completed, processing moves to the command table at the address of CMDLINK if the jump mode is set to assign, and if it is set to call, the command table at the CMDLINK address is called in a subroutine and processing moves to the next command table to be processed.

A value resulting from dividing the address in VRAM by 8H is specified with 16 bits as the address. Because command tables are stored in boundaries of 20H (32)-byte units, the lower 2 bits of CMDLINK become 00H.

When the jump mode specifies the next table or specifies return to the main routine, this CMDLINK is ignored.



### 6.3 CMDPMOD (Draw Mode Word)

CMDPMOD enables or disables clipping, specifies mesh processing and the transparent code, specifies the color mode, and controls color calculation and shade processing. It occupies the 2 bytes from the top address + 04H of the command table, and its bit configuration is as follows. Set unused bits to “0.”

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDPMOD +04H	MON	0	0	HSS	Pclip	Clip	Cmod	Mesh	ECD	SPD	Color mode	Color calculation bits				

#### High Speed Shrink (HSS): bit 12

Specifies whether speed or precision is given priority when scaled or distorted sprites are reduced and drawn.

#### Pre-clipping Disable (Pclip): bit 11

Specifies whether coordinate calculation, which judges whether clipping is required or not, is disabled or not.

#### User Clipping Enable Bit (Clip): bit 10

Specifies whether or not the part is drawn according to the user clipping coordinates already set in the case of a part draw command.

#### Clipping Mode Bit (Cmod): bit 9

Specifies whether clipping is performed inside or outside the user clipping coordinates when user clipping is enabled (Clip = 1).

#### Mesh Enable Bit (Mesh): bit 8

Specifies whether or not mesh processing is performed in the case of a part draw command.

#### End Code Disable (ECD): bit 7

Specifies whether or not the end code of the character pattern is disabled.

#### Transparent Pixel Disable (SPD): bit 6

Specifies whether or not the transparent pixel of the character pattern is disabled.

#### Color Mode Bits: bits 5~3

Specifies the number of colors in which the sprite is drawn and how they are expressed.

**Color Calculation Bits: bits 2~0**

Specifies color calculation of Gouraud shading, shadow, half-luminance, and half-transparent.

**MSB On Bit (MON): bit 15**

Sets the MSB to "1" (ON) on the frame buffer pixels that should be drawn, without changing their other bits. Used when VDP2 performs shadow or window processing depending on the MSB of the sprite data.



## High Speed Shrink

### High Speed Shrink (HSS): bit 12

Specifies whether speed or precision is given priority when scaled or distorted sprites are reduced and drawn.

This bit is only valid for drawing commands for scaled sprites and distorted sprites. Set it to "0" for other drawing commands.

When this bit is set to "1", only the pixels at even or odd coordinates of the original picture data are sampled for lines to be reduced when drawn. The selection of even coordinates or odd coordinates is performed by the even/odd coordinate selection (EOS) bit of the frame buffer change mode register (FBCR).

When "1" is specified, the end code of the original picture is ignored regardless of whether the sprite is being enlarged or reduced.

When "0" is specified, the original picture data is sampled at any magnification ratio irrespective of even or odd coordinates.

Specify "0" to give precedence to precision of drawing even though operation may be slowed down, and specify "1" to give fast operation precedence over precision.

HSS	Processing
0	High speed shrink disabled
1	High speed shrink enabled

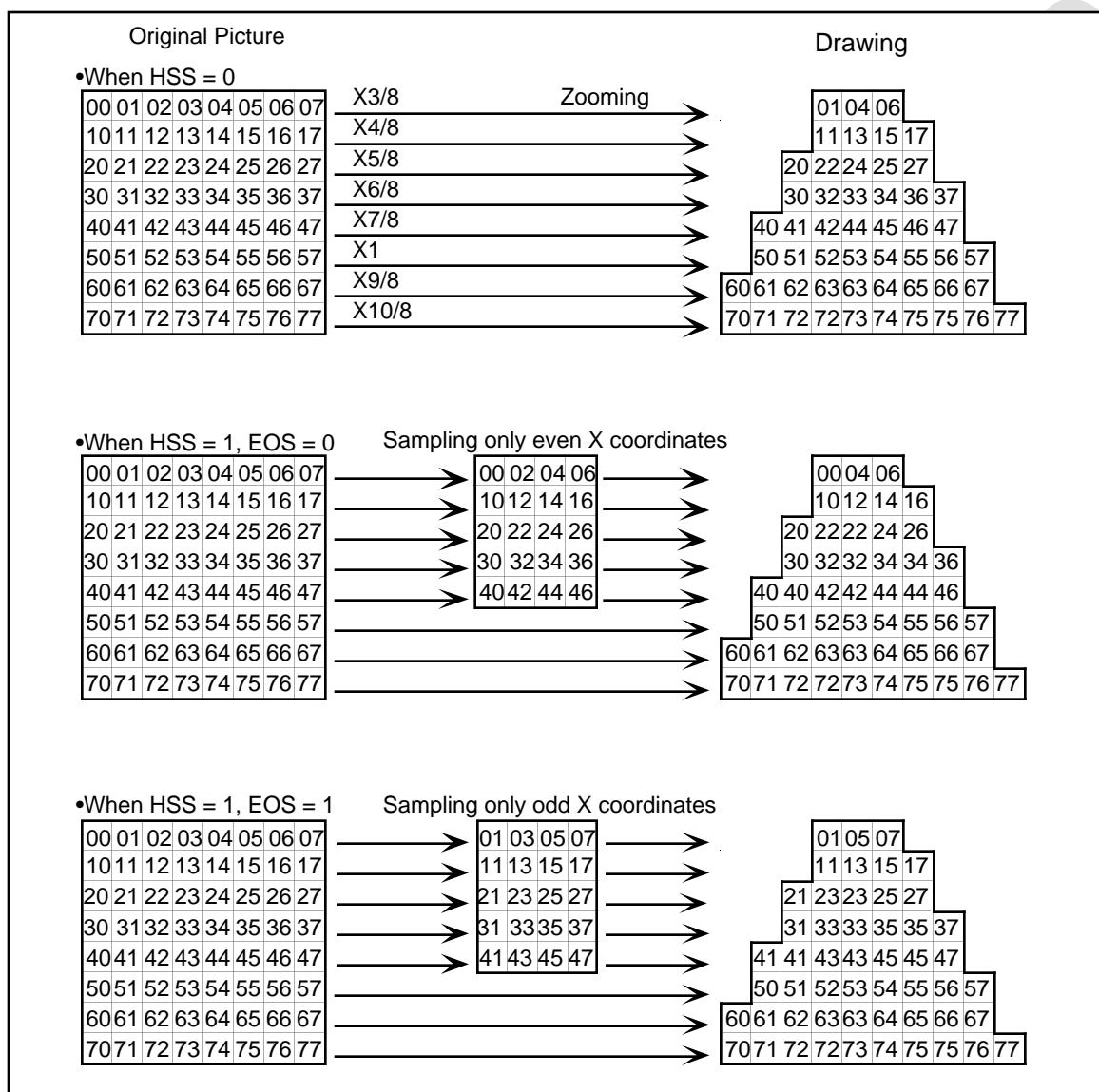


Figure 6.5 High Speed Shrink



## Pre-Clipping Disable

### Pre-Clipping Disable (Pclp): bit 11

Specifies whether pre-clipping is enabled or disabled. When "0" is specified, pre-clipping is performed. When "1" is specified, pre-clipping is not performed. One drawing command comprises a group of several lines, and the respective lines comprise a number of dots. Each dot is drawn based on clipping area (drawing area) information specified by the CPU.

For lines that are completely separated from the drawing area or if drawing of the entire line is not necessary and can be detected in advance, drawing efficiency can be raised by specifying the drawing not be started. Also, when one end of a line is outside the drawing area, efficiency can be improved by starting drawing from inside the drawing area (limited to vertical and horizontal lines).

VDP1 normally performs this detection, but in the case of small elements whose points are in the (A)—(B) or (D)—(C) direction, the overhead required for that detection (up to five CPU clock cycles for one line) becomes conspicuous and can lower the drawing efficiency.

In the case of large elements that extend greatly out of the drawing area, it is more efficient to perform pre-clipping. This bit is only valid for drawing commands. Set it to "0" for other commands.

Pclp	Processing
0	Pre-clipping with horizontal inversion
1	No pre-clipping and no horizontal inversion

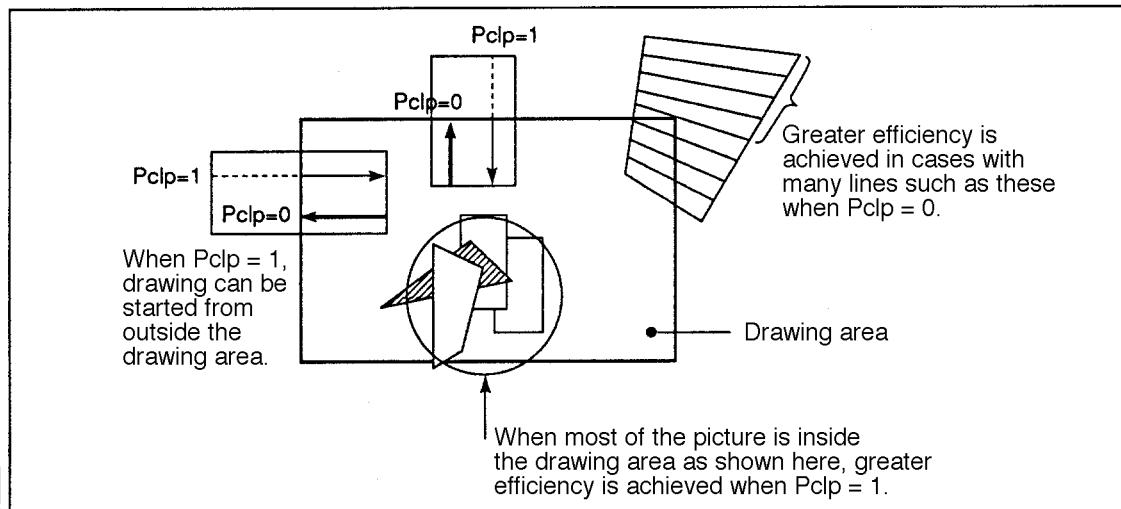


Figure 6.6 Pre-clipping

## User Clipping Enable

### User Clipping Enable Bit (Clip): bit 10

This bit specifies whether or not the part is drawn according to the already set user clipping coordinates when the command is a draw command for a part.

When the bit is “0,” the user clipping coordinates are ignored and the part is clipped and drawn according to the system clipping coordinates. When the bit is “1,” the part is clipped and drawn according to the user clipping coordinates and the specification of the clipping mode bit (Cmod). Even when this bit is “1,” the part is clipped according to the system clipping coordinates.

Both the user and system clipping coordinates become undefined after resetting. Therefore, the clipping register set command must be used to set the system clipping coordinates after resetting and to set the user clipping coordinates prior to the user clipping specification.

## User Clipping Mode

### Clipping Mode Bit (Cmod): bit 9

Specifies whether drawing is performed inside or outside the user clipping coordinates when user clipping is enabled (Clip = 1). When it is 0, drawing is performed inside. When it is 1, drawing is performed outside.

When Cmod = 1, the user clipping rectangle (including the lines) already specified becomes the drawing area. When Cmod = 0, the drawing area does not include the lines of the rectangle.

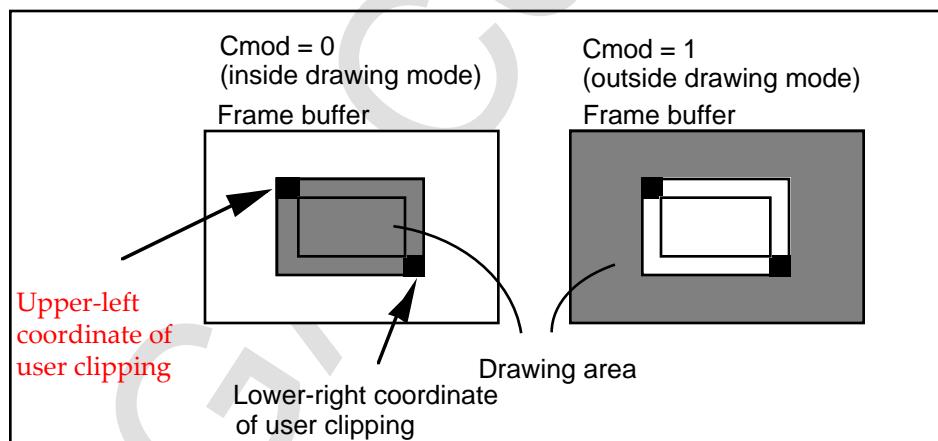


Figure 6.7 Drawing Area

Do not set this bit to “1” when user clipping is disabled (Clip = 0). Combinations of the user clipping enable bit and the clipping mode bit have the following results.

Clip	Cmod	User clipping processing
0	0	User clipping disabled
0	1	Setting prohibited (do not set)
1	0	Inside drawing mode
1	1	Outside drawing mode



## Mesh Enable

### Mesh Enable Bit: bit 8

This bit specifies whether or not mesh processing is performed when the command is a draw command for a part. When it is "0," the part is drawn without mesh processing. When it is "1," the part is drawn with mesh processing.

Mesh	Mesh enable
0	Draw without mesh processing
1	Draw with mesh processing

When mesh processing is specified (Mesh = 1), every other pixel of the part is drawn to form a mesh. Only pixels for which (X coordinate value + Y coordinate value) is even ( $X_{LSB} \text{ XOR } Y_{LSB} = 0$ ) are drawn, and odd pixels are skipped and not drawn. When the starting point of a  $45^\circ$  diagonal line is an odd coordinate, nothing is drawn. Nothing is drawn in some cases when the point of a  $45^\circ$  polyline is an odd coordinate.

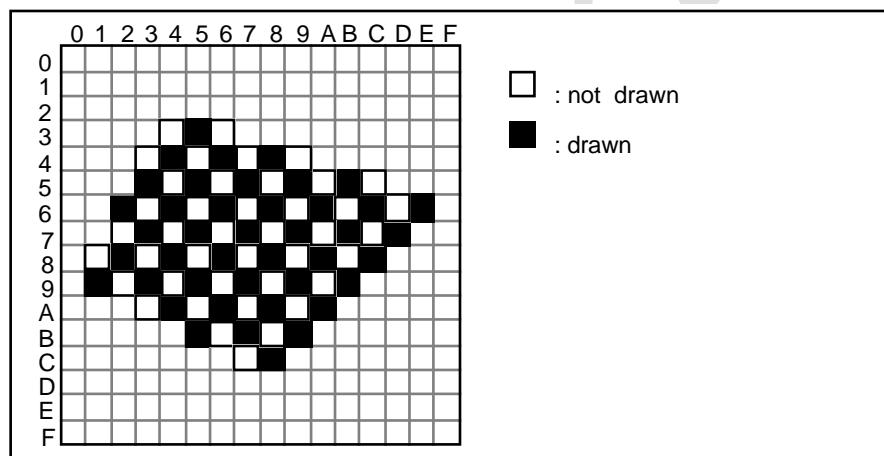


Figure 6.8 Mesh Processing

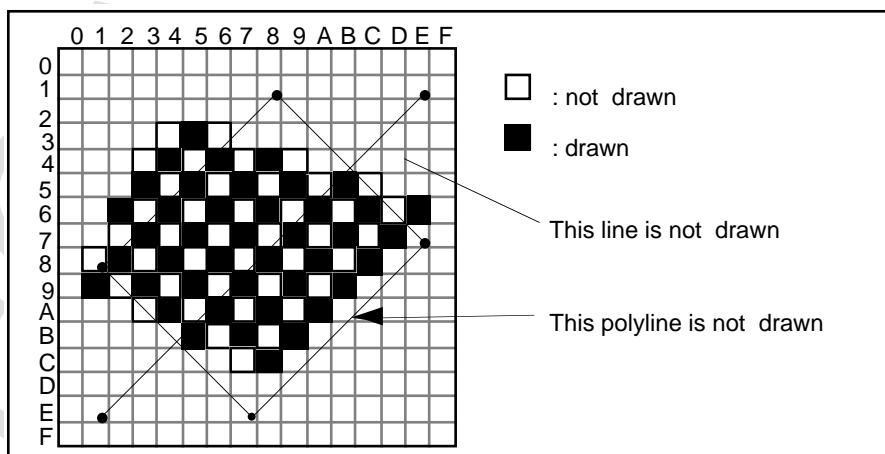


Figure 6.9 Mesh Processing of Lines and Polylines

## End Code Disable

### End Code Disable (ECD): bit 7

This bit specifies whether or not the end code of the character pattern is disabled. When it is “0,” the end code is enabled; when it is “1,” the end code is disabled.

The end code disable bit is only valid for drawing sprites with a character pattern. Set this bit to “1” for polygons, polylines, and lines.

If a second end code is read in the horizontal direction during drawing of a character pattern when end code is enabled (ECD = 0), drawing of that row is terminated there and drawing moves to the head of the next row. End codes are not drawn and that pixel becomes transparent. When the end code is disabled (ECD = 1), the end code is processed in the same way as other color codes.

Drawing in the horizontal direction is terminated when an end code is read twice, and an end code is only processed in the horizontal direction of the character pattern and is not affected by the vertical direction. Processing is performed irrespective of the transparent pixel disable bit (SPD).

When the end code is enabled (ECD = 0), it cannot be used for color display, and therefore, the number of colors that can be used is reduced by one. Use caution. Use end code disable (ECD = 1) in the case of sprites reduced in the horizontal direction by HSS = 1.

HSS	ECD	End code processing
0	0	End code enabled: drawing in horizontal direction is disabled when second end code is read and end code becomes transparent.
0	1	End code disabled: end code is not processed, color of code is expressed.
1 & enlarge	0	End code enabled: drawing in horizontal direction is disabled when second end code is read and end code becomes transparent.
1 & reduce	0	End code disabled: end code is not processed, color of code is expressed.
1	1	End code disabled: end code is not processed, color of code is expressed.

The relationship between the color mode and the end code is as follows. The number of bits of the end code differs depending on the color mode.

	Color mode	End code
0	16 colors (color bank mode)	FH ( 4 bits)
1	16 colors (lookup table mode)	FH ( 4 bits)
2	64 colors (color bank mode)	FFH ( 8 bits)
3	128 colors (color bank mode)	FFH ( 8 bits)
4	256 colors (color bank mode)	FFH ( 8 bits)
5	32,768 colors (RGB mode)	7FFFH (16 bits)



An example of end code processing is shown below.

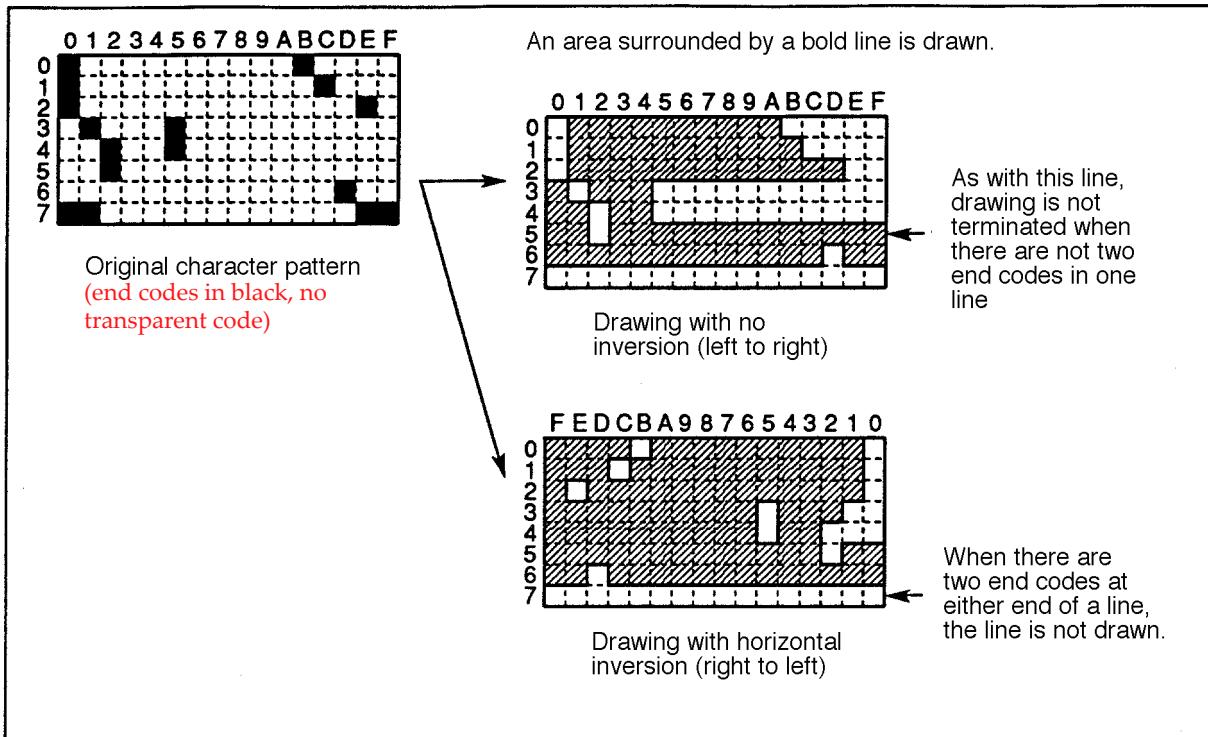


Figure 6.10(a) End Code Processing (1 of 2)

Since drawing is not allowed toward the outside from the end code when read from the left or right, place only a transparent pixel there. Therefore, when ECD = 0, SPD must equal 0. Do not use the combination ECD = 0 and SPD=1.

The drawing direction may be inverted by pre-clipping. When using end codes in the original picture, do so as shown below.

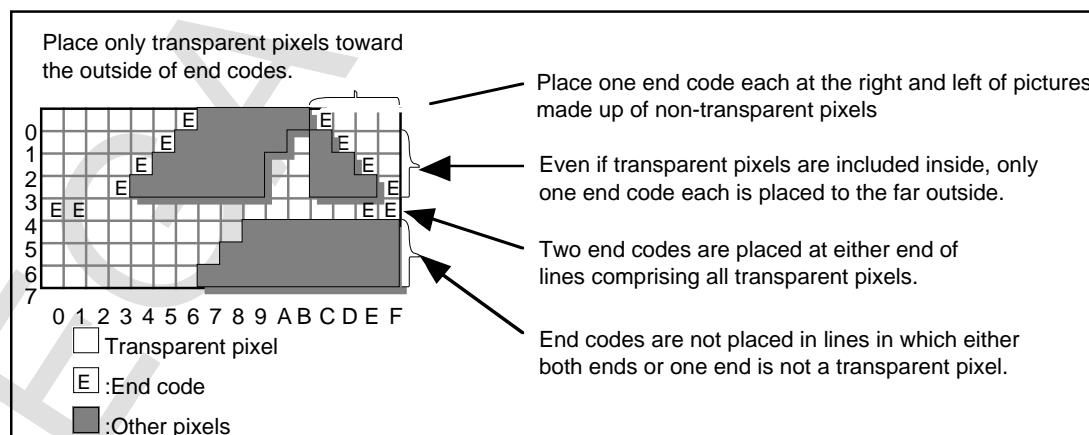


Figure 6.10(b) End Code Processing (2 of 2)

## Transparent Pixel Disable

### Transparent Pixel Disable (SPD): bit 6

This bit specifies whether or not the transparent pixel of the character pattern is disabled. When it is “0,” the transparent pixel is enabled; when it is “1,” the transparent pixel is disabled. The transparent pixel disable bit is only valid for drawing sprites with a character pattern. Be sure to set this bit to “1” for polygons, polylines, and lines.

When the transparent pixel is enabled (SPD = 0), transparent color codes in the character pattern become transparent pixels and are not drawn. **When the transparent pixel is disabled (SPD = 1)**, the transparent color code is processed like other color codes. When the transparent pixel is enabled (SPD = 0), the number of colors that can be used decreases by one, so use caution. For example, only 14 colors can be displayed when ECD = 0, SPD = 0, and color mode = 0.

SPD	Transparent code processing
0	Transparent pixel enable: transparent color codes are not drawn; transparent color codes become transparent.
1	Transparent pixel disable: <b>transparent color code is processed and drawn like other color codes.</b>

The relationship between the color mode and the transparent color code is as follows. The number of bits of the transparent color code differs depending on the color mode.

	Color mode	Transparent color code
0	16 colors (color bank mode)	0H ( 4 bits)
1	16 colors (lookup table mode)	0H ( 4 bits)
2	64 colors (color bank mode)	00H ( 8 bits)
3	128 colors (color bank mode)	00H ( 8 bits)
4	256 colors (color bank mode)	00H ( 8 bits)
5	32,768 colors (RGB mode)	0000H (16 bits)



## Color Mode

### Color Mode Bits: bits 5~3

These bits specify the method by which the number of colors to be drawn is expressed. It is only valid for sprites. Set the color mode to 000B for non-textures. The possible color modes include the color bank mode, which specifies the color with a palette code and a color bank; the lookup table mode, which uses a color lookup table; and the RGB mode, which specifies the luminance directly. In the color bank mode, the color can be selected from among 16, 64, 128, or 256 colors, depending on the number of bits of the pixel data. In the lookup table mode, the color can be selected from among 16 colors. And in the RGB mode, 32,768 colors can be drawn.

The color mode bits are described in the following table.

Color mode			Mode	Description		Bits per pixel
Bit 5	4	3		Number of colors	Mode	
0	0	0	0	16	Color bank mode	4 bits
0	0	1	1	16	Lookup table mode	4 bits
0	1	0	2	64	Color bank mode	8 bits
0	1	1	3	128	Color bank mode	8 bits
1	0	0	4	256	Color bank mode	8 bits
1	0	1	5	32,768	RGB mode	16 bits
Other than above			Setting prohibited (do not set)			

### Character Patterns in Each Mode

Character patterns are stored in VRAM in each mode. An example is shown that is 8 horizontal pixels and 1 vertical pixel.

#### Mode 0

This is the 16-color color bank mode. Color is expressed using palette codes and a color bank. One pixel is represented by 4 bits. 16 colors can be drawn. 4 bytes are necessary to express 8 pixels. The data for 2 pixels are contained in 1 byte, and when there is no horizontal inversion, the upper 4 bits represent the left pixel and the lower 4 bits represent the right pixel.

The palette code is represented by 4 bits, and the upper 12 bits are the color bank added from the color bank word (top address + 06H of command table), resulting in 16-bit data being written to the frame buffer. When there are 8 bits/pixel (frame buffer rotation or high resolution), the lower 8 bits of the 16 bits are written to the frame buffer. The upper 8 bits are ignored. **Color calculation can be executed when the color code is color bank code, but for other modes than replace and shadow, on-screen results depend on the ordering of VDP2 color RAM.**

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+00H	Pixel 0						1			2				3		
+02H		4					5			6				7		

#### Mode 1

This mode is the lookup table mode which uses a color lookup table. One pixel is represented by 4 bits. 16 colors can be drawn. The character data is the same as in mode 0. The data for 1 color of the 16-bit 16 colors stored in the color lookup table is selected with 4 bits and written to the frame buffer. When there are 8 bits/pixel (frame buffer rotation or high resolution), the lower 8 bits of the 16 bits are written to the frame buffer.

The address of the color lookup table is written to the lookup table address (top address + 06H of command table; also used as color bank word). Either color bank code or RGB code can be specified as the color code of the lookup table. However, RGB code is prohibited when there are 8 bits per pixel.

Original picture with 8 horizontal pixels									
+00H →	0 1 2 3   4 5 6 7					← +02H			
+04H →	8 9 A B   C D E F					← +06H			

Figure 6.11 Example of Drawing in Modes 0 and 1



**Mode 2, Mode 3, Mode 4**

Mode 2, mode 3, and mode 4 are respectively 64-color, 128-color, and 256-color palette bank modes. Colors are represented by palette code and a color bank.

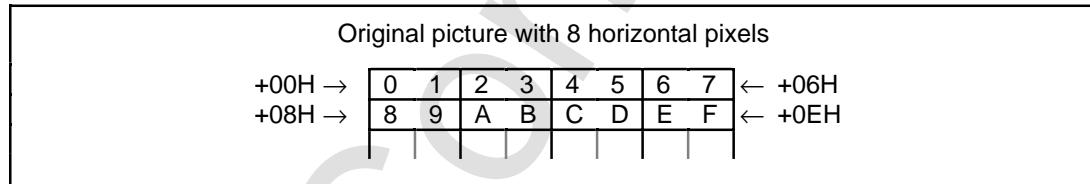
One pixel is represented by 8 bits. In the respective modes, 64 colors, 128 colors, and 256 colors can be drawn. 8 bytes are required to represent 8 pixels.

The palette code is represented respectively by 6, 7, and 8 bits, and the upper 10, 9, and 8 bits are the color bank added from the color bank word (top address + 06H of command table), resulting in 16-bit data being written to the frame buffer. In mode 2 and mode 3, the respective upper 2 bits and upper 1 bit of the pixels are ignored.

When there are 8 bits/pixel

(frame buffer rotation or high resolution), the lower 8 bits of the 16 bits are written to the frame buffer. Color calculation can be executed when the color code is color bank code, but for other modes than replace and shadow, on-screen results depend on the ordering of VDP2 color RAM.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+00H																1
+02H																3
+04H																5
+06H																7



**Figure 6.12 Example of Drawing in Modes 2, 3, and 4**

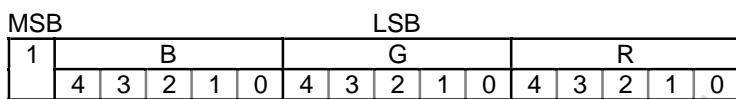
**Mode 5**

This is the 32,768-color RGB mode. The color is expressed by the respective luminances of red, green, and blue (RGB). One pixel is represented by 16 bits. It is possible to draw 32,768 colors. 16 bytes are required to express 8 pixels.

The RGB code is represented by a MSB (value 1), which indicates that the code is RGB code, and the respective luminances of R, G, and B are represented by 5 bits each. The respective luminances of R, G and B are represented by the values 00H to 1FH. The closer the value is to 00H, the darker the color; the closer it is to 1FH, the brighter the color. When all of R, G, and B are 00H, the value is 8000H, which represents black; when all are 1FH, the value is FFFFH, which represents white.

RGB codes are written in their original 16-bit form to the frame buffer. When there are 8 bits / pixel (frame buffer rotation or high resolution), this mode cannot be used. Color calculation can be performed on RGB code.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+00H	Pixel 0 (upper byte)								Pixel 0 (lower byte)							
+02H	1 (upper byte)								1 (lower byte)							
+04H	2 (upper byte)								2 (lower byte)							
+06H	3 (upper byte)								3 (lower byte)							
+08H	4 (upper byte)								4 (lower byte)							
+0AH	5 (upper byte)								5 (lower byte)							
+0CH	6 (upper byte)								6 (lower byte)							
+0EH	7 (upper byte)								7 (lower byte)							

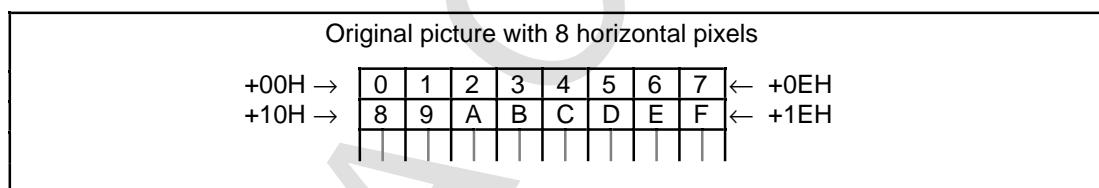


Note: The MSB is 1.

**Figure 6.13** RGB Code Format

**Table 6.2 Pixel Data**

Pixel data	Draw data
0000H	Transparent color code
0001H : 7FFEH	Setting prohibited (Do not set in RGB mode) (Palette bank code)
7FFFH	End code
8000H : FFFFH	RGB code



**Figure 6.14 Example of Drawing in Mode 5**



## Color Calculation

### Color Calculation Bits: bits 2~0

These bits specify Gouraud shading, shadow, half-luminance, and half-transparency. The functions of each of the bits are as follows.

Bit	Function
2	Gouraud shading enable bit
1	1/2 original graphic enable bit
0	1/2 background enable bit

Color calculation is performed on sprites or non-textured pixel data to be drawn and on pixel data already drawn in the frame buffer.

Except for color calculation of replace and shadow, color calculation can only be fully determined by VDP1 when the color code of the original picture is RGB code. Color calculation can be executed when the color code is color bank code, but on-screen results depend on the ordering of VDP2 color RAM.

In the case of sprites, color calculation can be performed when the color mode is mode 1 (lookup table mode) and the color code of the color lookup table is RGB code, or when the color mode is mode 5 (RGB mode). Color calculation can be performed for non-textured colors when the color code is RGB code.

In color calculation of parts with an original picture, the transparent code is valid when SPD = 0 and the end code is valid when ECD = 0, and therefore color calculation cannot be performed on those pixels.

Color calculation for polygons, polylines, and lines is enabled only when the non-textured color is RGB code. Color calculation cannot be performed when there are 8 bits/pixel (high resolution or rotation 8).

## Color Calculation

Color calculation includes replace, shadow, half-luminance, half-transparent, Gouraud shading, and the combination of half-luminance with Gouraud shading and half-transparent with Gouraud shading. These are specified as follows.

Color calculation (in Bit)			Background MSB	Original graphic	Background	Type of color calculation	Usable modes	
2	1	0					Original graphic	Background
0	0	0	—	1	0	Replace	Not restricted	Not restricted
0	0	1	0	0	1	Cannot rewrite	Not restricted	Palette
			1	0 <sup>1</sup>	1/2	Shadow		RGB
0	1	0	—	1/2	0	Half-luminance	RGB	Not restricted
0	1	1	0	1	0	Replace	RGB	Palette
			1	1/2	1/2	Half-transparent	RGB	RGB
1	0	0	—	Gouraud	0	Gouraud shading	RGB	Not restricted
1	0	1	—	—	—	Setting prohibited (do not set)	—	—
1	1	0	—	Gouraud 1/2	0	Gouraud shading + half-luminance <sup>2</sup>	RGB	Not restricted
1	1	1	0	Gouraud	0	Gouraud shading	RGB	Palette
			1	Gouraud 1/2	1/2	Gouraud shading + half-transparent <sup>3</sup>		RGB

**Notes** —: doesn't matter

Original graphic: sprite or pixel data to be drawn in non-textured color.

Background: pixel data already drawn in the frame buffer.

<sup>1</sup>Original graphic (transparent pixels, end code) is referenced.

<sup>2</sup>Data that has undergone saturation processing after Gouraud calculation is reduced by half.

<sup>3</sup>Background is added to data that has undergone saturation processing after Gouraud calculation is reduced by half.

### Replace (Color Calculation Mode = 0)

Color calculation is not performed for replace. Parts to be drawn are written, as is, to the frame buffer. **Pixel data already written to the frame buffer is not used.** Color calculation cannot be performed when there are 8 bits/pixel, so replace should be specified.

### Shadow (Color Calculation Mode = 1)

Processing differs depending on the MSB of the pixel data already written to the frame buffer. When the MSB of the frame buffer is "0," processing of color calculation, including replace, is not performed and the frame buffer is left as is. When the MSB of the frame buffer is "1," shadow is performed.

In shadow, pixel data already written to the frame buffer is subjected to color calculation. The area in the frame buffer on which color calculation is to be performed is sought from the character pattern and its draw coordinates in the case of sprites, and from the draw coordinates in the case of non-textures. The luminance of pixel data already written in the frame buffer in the area where the part is to be drawn becomes one half for each of R, G, and B.

Shadow does not change the MSB of the background. Transparent areas remain transparent.



Shadow halves the luminance of pixels in the frame buffer. To make the luminance one fourth, set the same command table in VRAM twice. To make it one eighth, set the same command table in VRAM three times.

In shadow, calculation is performed on the pixel data read from the coordinates to which the pixel data of the original graphic is written. Drawing in this case slows down, so use caution—It takes six times longer than when color calculation is not performed.

#### **Half-Luminance (Color Calculation Mode = 2)**

Half-luminance halves the luminance of each of R, G, and B of the pixel data of the part to be drawn. **Pixel data already written to the frame buffer is not used.**

#### **Half-Transparency (Color Calculation Mode = 3)**

Processing differs depending on the MSB of the frame buffer. When the MSB is “0,” replace is performed. When the MSB of the frame buffer is “1,” half-transparency results. In half-transparency, one half (average) the sum of the data of the original graphic and the background is written to the frame buffer.

Color calculation of half-transparency is performed on the pixel data of the original graphic and the pixel data read from the write coordinates. Drawing in this case slows down, so use caution—it takes six times longer than when color calculation is not performed.

#### **Gouraud Shading (Color Calculation Mode = 4)**

Processes Gouraud shading using the RGB code set in the Gouraud shading table. **Pixel data already written to the frame buffer is not used.** The Gouraud shading table address is specified by CMDGRDA (top address + 1CH of command table).

#### **Gouraud Shading + Half-luminance (Color Calculation Mode = 6)**

Gouraud shading + half-luminance is processing that combines half-luminance with Gouraud shading. The part undergoes half-luminance processing after receiving Gouraud shading processing.

#### **Gouraud Shading + Half-Transparency (Color Calculation Mode = 7)**

Processing differs depending on the MSB of the frame buffer. When the MSB of the frame buffer is “0,” Gouraud shading is performed. When the MSB of the frame buffer is “1,” Gouraud shading + half-transparency is performed.

### Original Graphic

In parts (sprites) with an original graphic, color calculation other than shadow is enabled only in the RGB mode (lookup table is RGB code in color mode 1 and color mode 5). If not in the RGB mode (characters in color bank mode), the results cannot be guaranteed.

In color calculation of parts with an original graphic, the transparent code is enabled when SPD = 0 and the end code is enabled when ECD = 0; therefore color calculation of that pixel cannot be performed.

When color calculation other than shadow is performed in the RGB mode (color mode 5) and when SPD = 1, the result for pixels with transparent color code (0000H) of the original graphic cannot be guaranteed. They are treated as black in color calculation.

When color calculation other than shadow is performed in the RGB mode (color mode 5) and when ECD = 1, the result for pixels with end code (7FFFH) of the original graphic cannot be guaranteed. They are treated as white in color calculation.

In color calculation other than shadow on polygons, polylines, and lines, non-textured color is enabled only for RGB code.

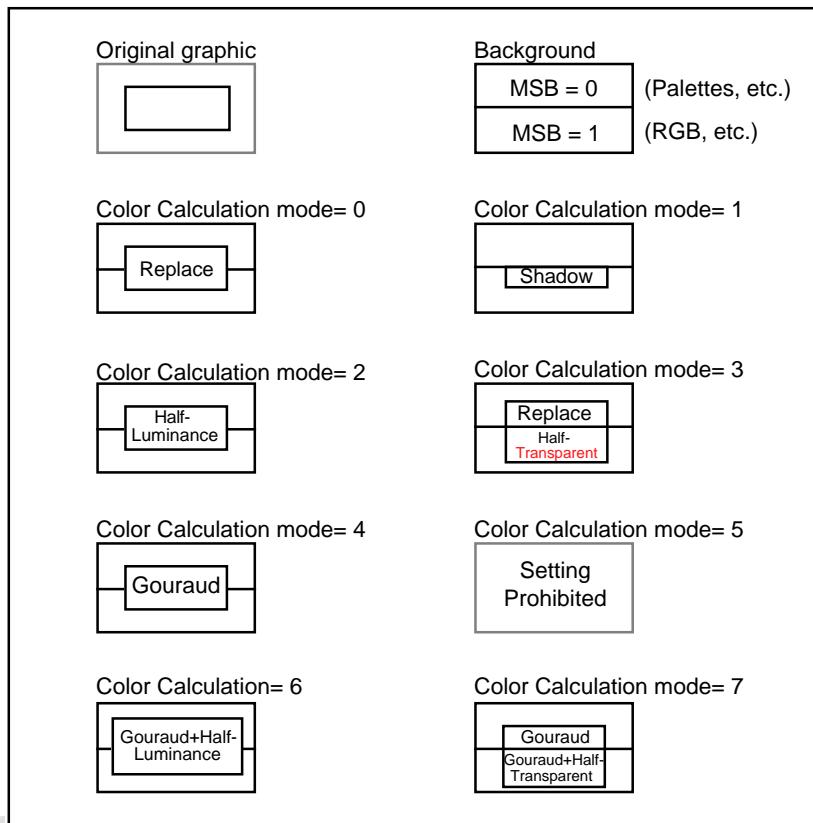
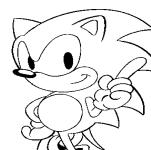


Figure 6.15 Examples of Color Calculation



## MSB ON

### MSB ON Bit (MON): bit 15

MON	Processing
0	MSB of pixel data in frame buffer is not changed
1	Sets MSB of pixel data in frame buffer to 1

When a mode is set in which the VDP2 uses shadow (drops luminance of pixel data in scroll screen, see MSB Shadow, p. 258 of VDP2 User's Manual) or window (displays a different screen in the specified area, see Sprite Window, p. 187 of VDP2 User's Manual), shadow and window processing are performed when the most significant bit (MSB) is "1".

Set the MSB to "1" (ON) for the pixels already written to the frame buffer. On-screen results are guaranteed only when the frame buffer is all in palette format (see Sprite color mode bit, p. 207 of VDP2 User's Manual). This is because the sprite pixels MSB is used for shadow or window processing by VDP2, instead of an RGB recognition bit.

In the case of textured parts, this can be used only when the original graphic data has or does not have pixels (no pixels for transparent and end codes and pixels for everything else), and therefore RGB code or color bank code can be used as the color code. In the case of non-textured parts, the non-textured color is not reflected in the drawing.

Do not specify color calculation (specify replace instead) when the MSB is set to "1." When the MSB is set to "1," color calculation has no meaning and processing takes a long time, so use caution.

For parts that undergo mesh processing, the MSB is set to ON in the mesh condition. An example of MSB ON is shown below.

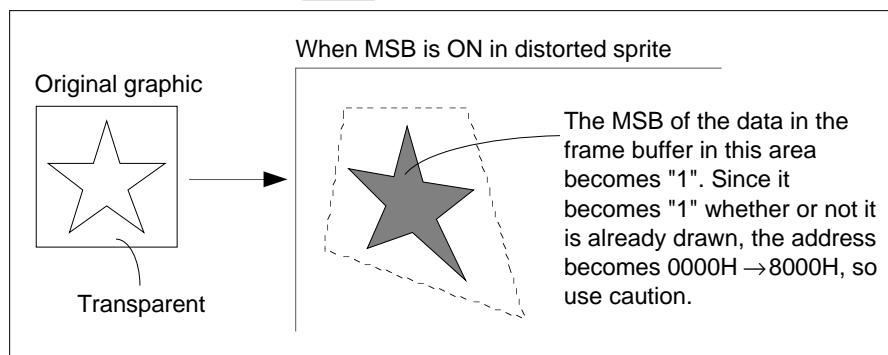


Figure 6.16 MSB ON

## 6.4 CMDCOLR (Color Control Word)

CMDCOLR specifies the color of the part. The function differs depending on the part and the color mode. It specifies the color bank, the color lookup table address, or non-textured colors. CMDCOLR is at the top address + 06H of the command table and is 16 bits.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCOLR +06H																

Color bank, color lookup table/8H (LSB fixed at 00), non-textured colors

### CMDCOLR: top address + 06H of command table

When a textured part is in the color bank mode, color bank is specified; when it is a textured part in the lookup table mode, the lookup table address is specified, and when it is a non-textured part, a non-textured color is specified. When it is a textured part in the RGB mode, this word is ignored.

**Table 6.3 CMDCOLR**

Part	Color mode	CMDCOLR
Textured part	Color bank mode	Color bank
	Lookup table mode	Lookup table address
	RGB mode	Ignored
Non-textured part		Non-textured color



## Color Bank

- Color banks are bits added to the upper bit when the pixel data of the character pattern are written to the frame buffer when the color mode of a textured part is the color bank mode.
- Color bank is specified using 16 bits, but the lower 4 bits must be fixed at "0." Depending on the color mode, 12, 10, 9 or 8 bits are added. Figure 6.17 shows the addition of bits.
- When there are 16 bits/pixel, the data is written to the frame buffer as is. When there are 8 bits/pixel, the lower 8 bits are written.

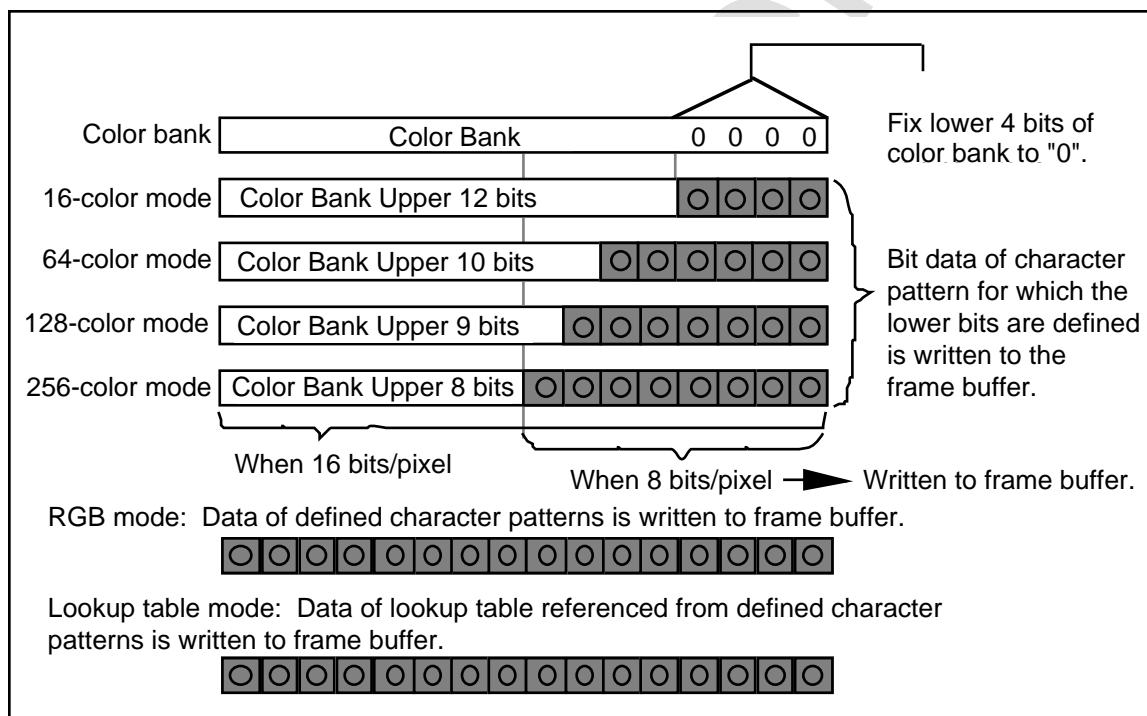


Figure 6.17 Color Bank

- The data written to the frame buffer are transferred to the VDP2 as is (16-bit data) when the data written to the frame buffer become the display buffer in the next frame.
- Table 6.4 shows the relationship between the color bank, defined character pattern data, and draw frame buffer data.

**Table 6.4 Example of Relationship of Defined Data and Draw Data to Color Bank**

Color Bank	Defined Data		Draw Data			
0000H	16-color mode	DH	0	0	0	D H
	64-color mode	CDH	0	0	0	D H
	128-color mode	CDH	0	0	4	D H
	256-color mode	CDH	0	0	C	D H
	RGB mode	ABCDH	A	B	C	D H
0010H	16-color mode	DH	0	0	1	D H
	64-color mode	CDH	0	0	0	D H
	128-color mode	CDH	0	0	4	D H
	256-color mode	CDH	0	0	C	D H
	RGB mode	ABCDH	A	B	C	D H
1230H	16-color mode	DH	1	2	3	D H
	64-color mode	CDH	1	2	0	D H
	128-color mode	CDH	1	2	4	D H
	256-color mode	CDH	1	2	C	D H
	RGB mode	ABCDH	A	B	C	D H

Note: The shaded areas represent color bank code.



## Color Lookup Table

- Defines the address of the color lookup table.
- When the color mode is the lookup table mode, the 4-bit data of the character pattern of textured parts is converted to 16-bit pixel data by referencing the color lookup table, and is written to the frame buffer.
- The color lookup table defines 16-bit color codes for 16 colors. The 16-bit data in the table is written as is to the frame buffer, and therefore there is no distinction between palette bank code and RGB code in the character pattern of textured parts. When read to the VDP2, the code is handled as color bank code when the MSB of the 16 bits is “0,” and as RGB code when it is “1.”
- The color lookup table address defines the address /8H of the color lookup table. Because the color lookup table is defined with 20H-byte boundaries, the two LSBs of the lookup table address are fixed at “00.”

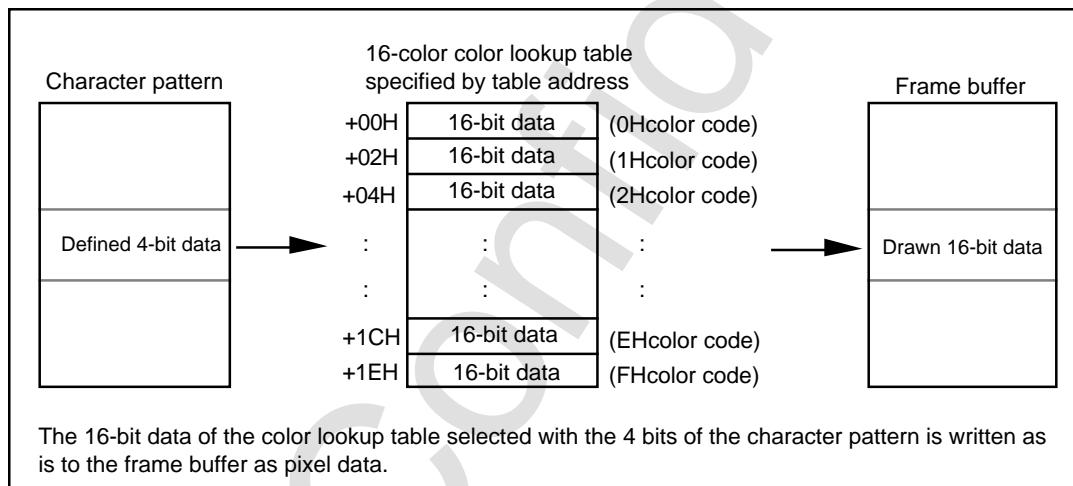


Figure 6.18 Color Lookup Table

## Non-Textured Color

- Non-textured color defines the color of non-textured parts.
- In the case of a non-textured part, the non-textured color is written, as is, to the frame buffer as pixel data.
- Non-textured colors are not related to transparent color codes and end codes.
- When the display mode is 8 bits / pixel, the lower 8 bits become valid. Only palette bank codes can be specified at this time.
- When color calculation (except shadow) is performed on non-textured parts, be sure to set RGB code for the non-textured color.



## 6.5 CMDSRCA (Character Address)

- CMDSRCA specifies the address that defines the character pattern.
- CMDSRCA is valid for texture draw commands. The address is specified with 16 bits at the top address + 08H of the command table. Set unused bits to “0.”

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSRCA +08H																0 0

- Specifies an address that defines the character pattern drawn by the texture draw command.
- This is an address in VRAM of the character pattern to be drawn, and the value resulting from dividing the address by 8H is specified with 16 bits.
- The character address specifies the top address at which the pixel data of the defined character pattern is stored. The pixel data of the upper-left point of the character pattern is at the top address. Even if the character is rotated or inverted horizontally or vertically, the address of the upper-left point of the defined character pattern is specified.
- The content of the pixel data of the character pattern is 4, 8, or 16 bits / pixel, depending on the color mode.
- The size of the table is specified by CMDSIZE (Character size).
- Because the character pattern table is defined with 20H-byte boundaries, the two LSBs are fixed at “00B.”

## 6.6 CMDSIZE (Character Size)

CMDSIZE specifies the size of the defined character pattern. CMDSIZE is valid for texture draw commands. It is specified with 16 bits at the top address + 0AH of the command table. Set unused bits to "0."

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSIZE +0AH	0	0														

Character size X/8                      Character size Y

- The character size is defined in multiples of 8 horizontally (X) and in 1-pixel units vertically.
- Horizontally, values from 1 to 63 can be specified with 6 bits.** Because the size is specified in 8-pixel units, sizes from 8 to 504 pixels can actually be specified. Do not specify "0."
- Vertically, values from 1 to 255 can be specified with 8 bits. Do not specify "0."

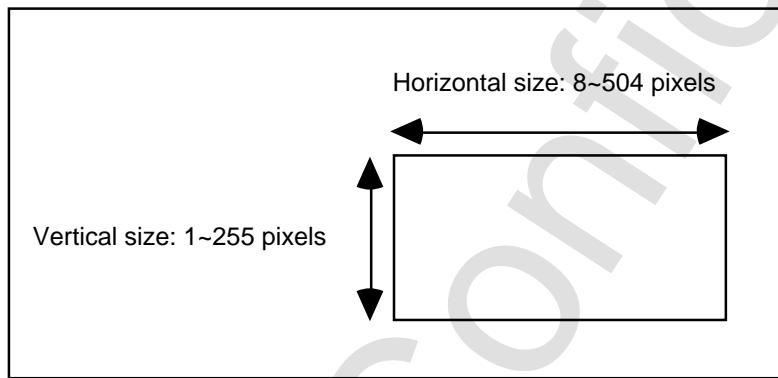


Figure 6.19 CMDSIZE

- Table 6.5 shows the relationships between the values defined in the command table and the number of drawn pixels.

Table 6.5 Relationships between Settings and Drawn Pixels

Horizontal (X) direction		Vertical (Y) direction	
Setting in command table	Number of pixels actually drawn	Setting in command table	Number of pixels actually drawn
0	Setting prohibited	0	Setting prohibited
1	8	1	1
2	16	2	2
:	:	:	:
63	504	255	255



## 6.7 CMDXA~CMDYD (Vertex Coordinate Data)

The coordinates of clipping areas, the point coordinates for drawing sprites and other coordinates are specified by CMDXA~CMDYD. The coordinates for a maximum of four vertex (X coordinates, Y coordinates) can be specified. They are defined in the 16 bytes (8 words) from the top address + 0CH to 1AH of the command table.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDXA +0CH	Code extension															
CMDYA +0EH	Code extension															
CMDXB +10H	Code extension															Vertex (B), X coordinates (XB); or display X width (XB)
CMDYB +12H	Code extension															Vertex (B), Y coordinates (YB); or display Y width (YB)
CMDXC +14H	Code extension															Vertex (C), X coordinates (XC)
CMDYC +16H	Code extension															Vertex (C), Y coordinates (YC)
CMDXD +18H	Code extension															Vertex (D), X coordinates (XD)
CMDYD +1AH	Code extension															Vertex (D), Y coordinates (YD)

**Note:** The top bit of the vertex coordinates is a sign bit. A negative value is indicated by a complement of two. Extend the code to the upper 5 bits.

- Each of the coordinates is specified CMDXA~CMDYD with 16 bits. A negative value is specified by a complement of 2. The range that can be specified is -1024 to 1023. The upper 5 bits are code-extension bits, and the same value as bit 10 is written to them.
- The meaning of CMDXA~CMDYD differs depending on the command. Table 6.6 shows the correspondence between the commands and CMDXA~CMDYD. For more information, refer to the section for each command in chapter 7.

**Table 6.6 Correspondence between Commands and CMDXA~CMDYD**

Command	CMDXA CMDYA	CMDXB CMDYB	CMDXC CMDYC	CMDXD CMDYD
Normal sprite draw command	Vertex (A)	—	—	—
Scaled sprite draw command (two methods)	Specify coordinates for two points	Vertex (A)	—	Vertex (C)
	Specify fixed point	Zoom point coordinates	Display width	—
Distorted sprite draw command	Vertex (A)	Vertex (B)	Vertex (C)	Vertex (D)
Polygon draw command	Vertex (A)	Vertex (B)	Vertex(C)	Vertex (D)
Polyline draw command	Vertex (A)	Vertex (B)	Vertex (C)	Vertex (D)
Line draw command	Vertex (A)	Vertex(B)	—	—
User clipping coordinate set command	Upper-left coordinates	—	Lower-right coordinates	—
System clipping coordinate set command	—	—	Lower-right coordinates	—
Local coordinate set command	Local coordinates	—	—	—

**Note:** "—" indicates unused.

## 6.8 CMDGRDA (Gouraud Shading Table)

1. Specifies the address of the Gouraud shading table.
2. The Gouraud shading table address is valid when Gouraud shading processing is specified for color calculation and is specified with 16 bits at the top address + 1CH of the command table.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDGRDA +1CH																Gouraud shading table/8H

3. The address of the Gouraud shading table is divided by 8H and is specified with 16 bits.
4. The Gouraud shading table is a table for Gouraud shading processing, and it defines the amount of change in the luminances of R, G, and B at four vertices (two points for lines). The size of the table is 8H bytes (4 words), and the table is defined with 8H-byte boundaries.



# **Chapter 7**

## **Commands**

### **Contents**

7.1	System Clipping Coordinate Set Command .....	110
	System Clipping .....	111
7.2	User Clipping Coordinate Set Command .....	112
	User Clipping .....	113
7.3	Local Coordinate Set Command .....	116
	Local Coordinates .....	117
7.4	Normal Sprite Draw Command .....	118
7.5	Scaled Sprite Draw Command.....	120
	Specification of Two Coordinate Points .....	120
	Specification of Zoom Point .....	122
7.6	Distorted Sprite Draw Command .....	124
7.7	Polygon Draw Command .....	126
7.8	Polyline Draw Command .....	128
7.9	Line Draw Command .....	130
7.10	Draw End Command .....	132

Page 108 was blank in the original document.

SEGA Confidential



Commands are determined by the end bit and the command selection bits. Table 7.1 lists the commands.

**Table 7.1 Commands**

END	Comm				Function		Command
Bit 15	3	2	1	0			
0	1	0	0	0	Coordinate set commands	Clipping coordinate set commands	User clipping coordinate set command
				1			System clipping coordinate set command
				1			Local coordinate set command
	0	0	0	0		Texture draw commands	Normal sprite draw command
			0	1			Scaled sprite draw command
			1	0			Distorted sprite draw command
			1	0	Non-texture draw commands	Polygon draw command	Polygon draw command
			0	1			Polyline draw command
			1	0			Line draw command
1	0	0	0	0	Draw terminate command		

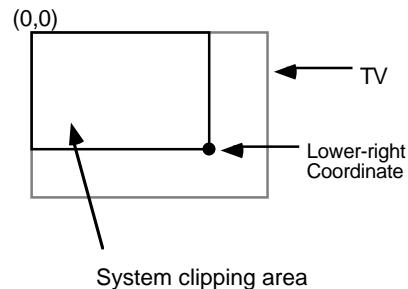
## 7.1 System Clipping Coordinate Set Command

Clipping is the removal of graphics outside the set display area so that they are not drawn. Clipping includes system clipping, which sets the drawing area for the system, and user clipping, which makes it possible to freely set the clipping area using software.

When the command select bits (bits 3 to 0) are 1001B, the system clipping coordinate set command is set; when they are 1000B, the user clipping coordinate set command is set. The contents of the system clipping coordinate set command table are shown in the following figure.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	0	JP	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
CMDLINK	+00H																
	+02H																
	+04H																
	+06H																
	+08H																
	+0AH																
	+0CH																
	+0EH																
	+10H																
	+12H																
CMDXC	+14H	0	0	0	0	0	0										
	+16H	0	0	0	0	0	0										
CMDYC	+14H	Lower-right X coordinate (XC)															
	+16H	Lower-right Y coordinate (YC)															
	+18H																
	+1AH																
	+1CH																

Note:  is ignored.



Upper-left coordinates are fixed at (0,0).

The system clipping coordinate set command is defined as follows:

- When the end bit (bit 15) is set to 0B and the command select bits are set to 1001B, the system clipping coordinate set command is enabled.
- The jump mode is specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and is set in CMDLINK.
- The lower-right coordinates of the clipping area are defined by CMDXC, CMDYC. The upper-left coordinates are fixed at (0,0). The lines of the rectangle (quad-angle) represented by the upper-left coordinates (0,0) and the lower-right coordinates (XC, YC) and inside the rectangle make up the drawing area.



## System Clipping

System clipping coordinates are always valid when drawing, so that outside the set area is clipped; that is, the area inside the coordinates is drawn. Clipping processing is performed in a rectangle. The coordinates are specified by defining the values of the lower-right coordinates (XC, YC) in the command table, since the upper-left coordinates are fixed at (0,0).

Because the clipping coordinates are not checked, they should be set in advance so that  $XC \geq 0$  and  $YC \geq 0$ . Operation cannot be ensured if  $XC < 0$  or  $YC < 0$ . Points on the clipping line are treated as being inside the clipping area and are drawn. The clipping coordinate set command rewrites the internal clipping coordinate register. Parts subsequent to rewriting are drawn by referencing those values.

Because this command can be defined in any number in one frame, it is possible to give part groups different clipping coordinates. The system clipping coordinates become undefined after powering on or after resetting, and therefore they must be set before drawing starts.

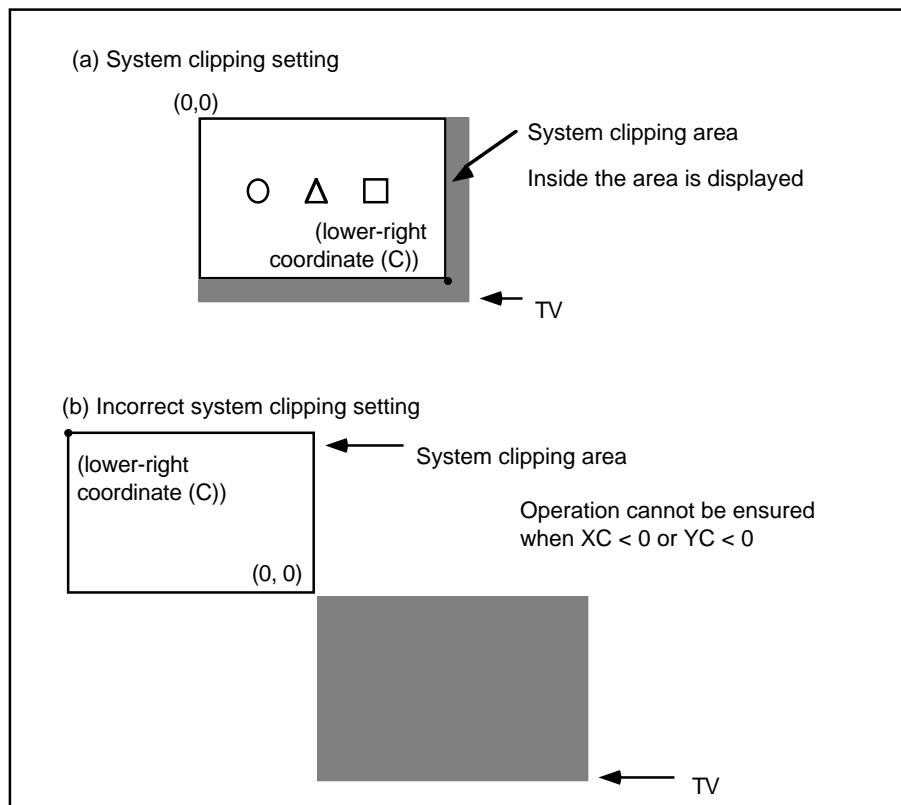
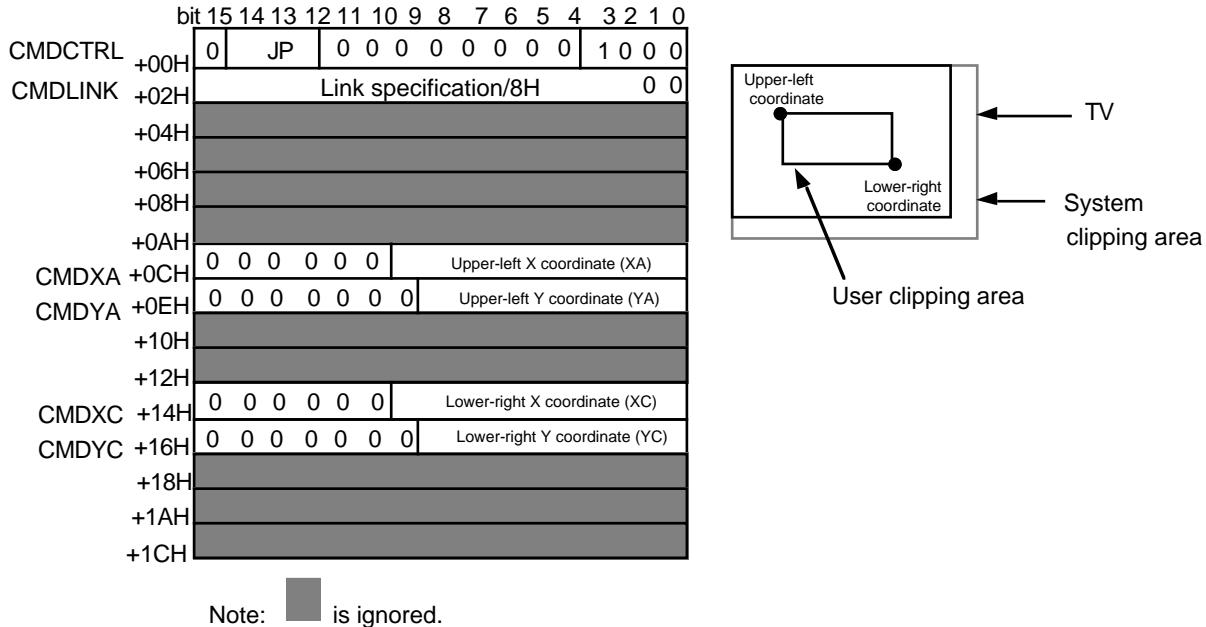


Figure 7.1 System Clipping

## 7.2 User Clipping Coordinate Set Command

User clipping coordinates can be freely set using software. When the command selection bits are 1000B, the user clipping coordinate set command is enabled. The contents of the user clipping coordinate set command table are shown in the following figure.



The user clipping coordinate set command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 1000B, the user clipping coordinate set command is enabled.
- The jump mode is then specified. When the jump mode is assign or call, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The upper-left coordinates of the clipping area are defined as vertex (A) (CMDXA, CMDYA), and the lower-right coordinates are defined as vertex (C) (CMDXC, CMDYC). The rectangle (quadrangle) represented by the upper-left coordinates (XA, YA) and the lower-right coordinates (XC, YC) is the clipping area. User clipping is enabled, and whether the inside or the outside of the area is clipped is determined by the draw mode of the draw command for the part.



## User Clipping

User clipping can be selected using software, and it is possible to specify whether or not user clipping is enabled for each part and whether the outside of the set area or the inside of the set area is clipped.

User clipping coordinates can be enabled or disabled using the user clipping enable bit (Clip bit, bit 10 at top address + 04H of command table). When this bit is disabled, the user clipping coordinates become disabled, and parts are only clipped by the system clipping coordinates. When the user clipping coordinates are enabled, parts are clipped by both the system clipping coordinates and the user clipping coordinates.

When user clipping is enabled, user clipping can be set to the inside drawing mode or the outside drawing mode with the clipping mode bit (Cmod bit, bit 9 at top address + 04H of command table). When set to the inside drawing mode, only those parts inside the area set by the user clipping coordinates are drawn; when set to the outside drawing mode, the area outside the coordinates is drawn. System clipping is performed regardless of user clipping.

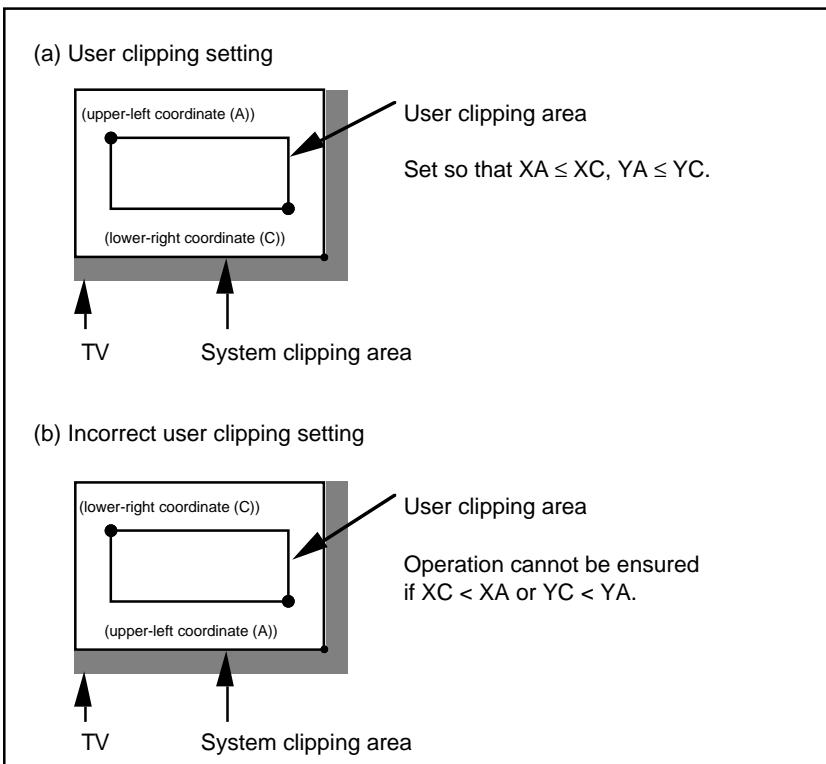
The clipping coordinates form a rectangle. The coordinates are the upper-left coordinates (XA, YA) and the lower-right coordinates (XC, YC), and they are specified by defining the values of the coordinates of these two points in the command table.

Because the clipping coordinates are not checked, they must be set in advance so that  $XA \leq XC$  and  $YA \leq YC$ . Operation cannot be guaranteed if  $XC < XA$  or  $YC < YA$ . Set the user clipping coordinates inside the set area (including on-line) of the system clipping coordinates. Operation cannot be guaranteed when the user clipping coordinates are set beyond the set area of the system clipping coordinates.

Points on the clipping line are treated as being inside the clipping area. They are drawn in the inside drawing mode and not drawn in the outside drawing mode.

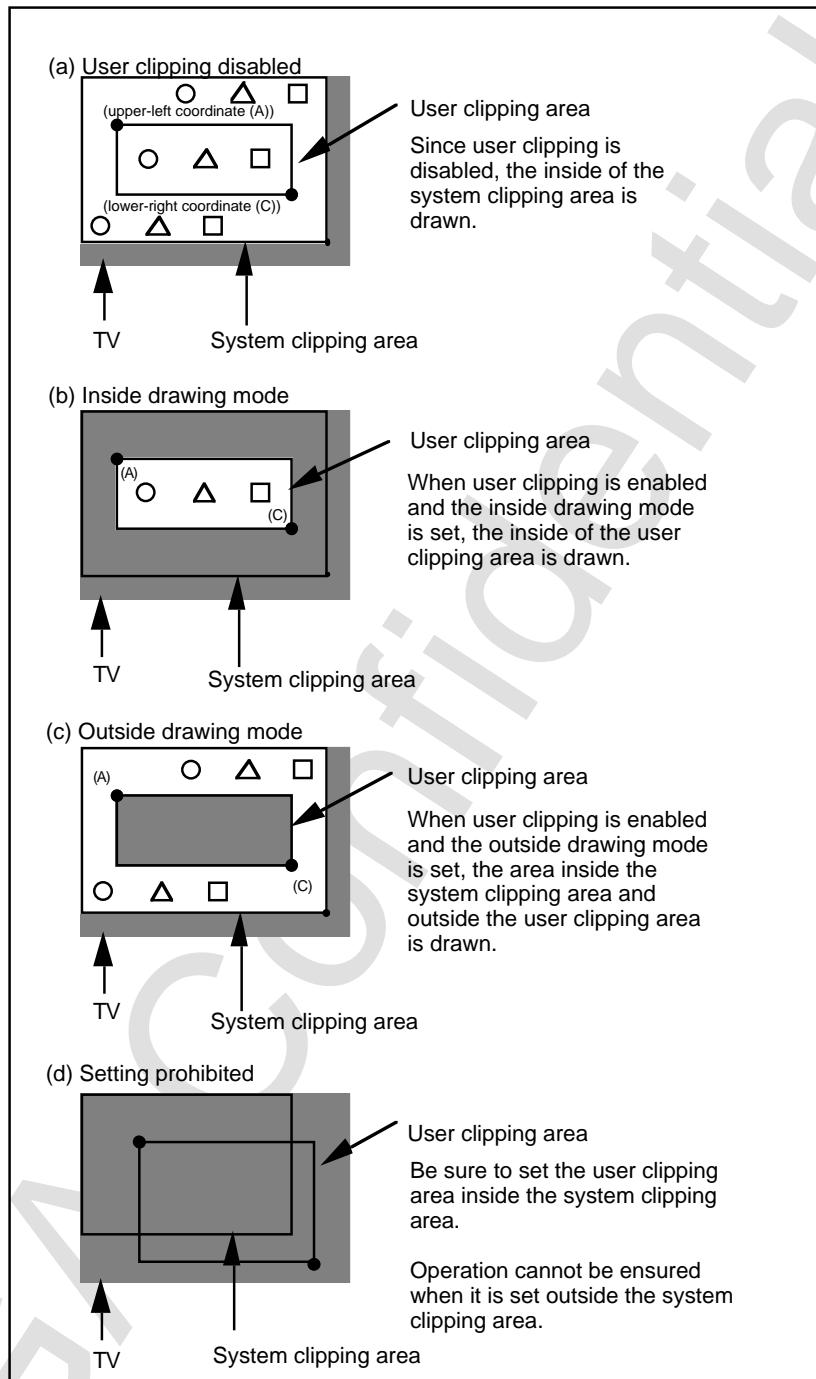
Two or more user clipping commands can be set in the same frame. The clipping coordinate set commands rewrite the internal clipping coordinate registers. Parts subsequent to rewriting are drawn by referring to these values.

Because the user clipping coordinates become undefined after powering on or after resetting, they must be set before drawing starts.



**Figure 7.2 User Clipping Settings**



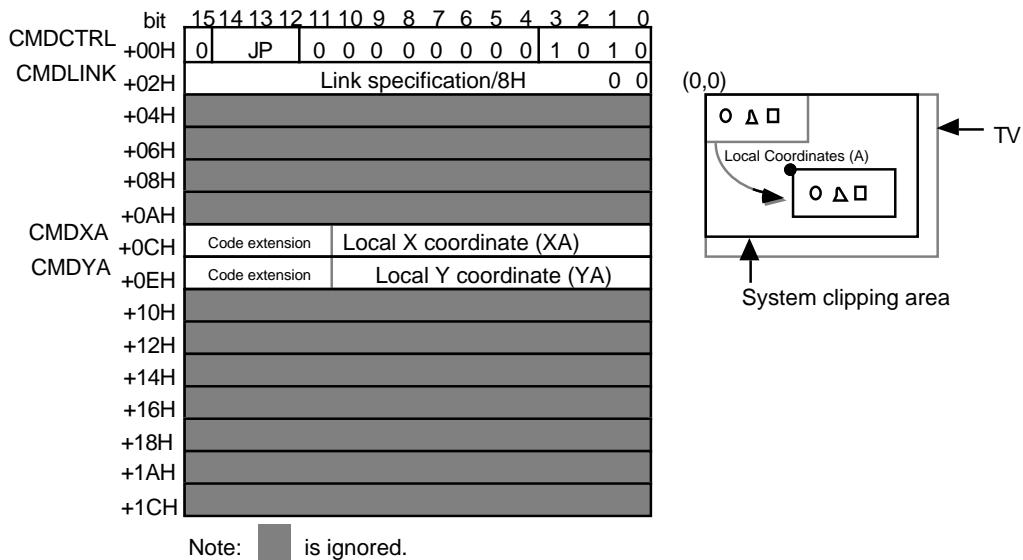


**Figure 7.3 User Clipping**

## 7.3 Local Coordinate Set Command

The local coordinate set command makes the coordinates specified by the draw command local coordinates, and then makes them drawing coordinates by adding the value specified by the local coordinate command.

When the command selection bits are 1010B, the local coordinate set command is enabled. The contents of the command table are as shown in the following figure.



The local coordinate set command is defined as follows.

- When the end bit is set to 0B and the command selection bits are set to 1010B, the local coordinate set command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The local coordinates are defined in CMDXA, CMDYA.



## **Local Coordinates**

Local coordinates are set in the local coordinate register. The values of the local coordinates are added to the coordinates specified by the draw command, and they become the drawing coordinates. Parts are drawn in the frame buffer using the drawing coordinates as a reference.

When the local coordinates are (0,0), the part is drawn at the coordinates specified by the draw command using the upper-left corner of the screen as (0,0). When (0,0) is set approximately in the center of the screen, (160,112) are specified as the local coordinates.

Because the local coordinates are retained in the register until they are set again, in order to move the coordinates of several parts and draw them together, the local coordinates are set before their respective draw commands.

Coordinate comparison in clipping is processed using the value resulting from adding these local coordinates to the coordinates specified by drawing of the part.

Furthermore, because the local coordinates are not added to the clipping coordinates, the clipping area does not move. Because the values in the local coordinate register become undefined after powering on or after resetting, they must be set before drawing starts.

## 7.4 Normal Sprite Draw Command

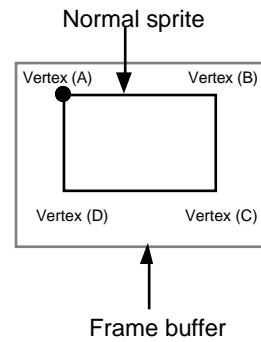
The normal sprite draw command draws character patterns in the frame buffer. When drawing, the character pattern can be inverted vertically or horizontally at the specified coordinates and drawn.

When the end bit is 0B and the command selection bits are 0000B, the normal sprite draw command is enabled. Normal sprites cannot be rotated 90°. To rotate 90°, specify the distorted sprite draw command. The contents of the command table are shown in the following figure.

	bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
CMDCTRL +00H	0	JP	0	0	0	0	0	0	Dir	0	0	0	0	0	0	0					
Link specification/8H																					
CMDLINK +02H	MO	0	0	0	Pc	Cl	Cm	Me	EC	SP	Color mode	Color calculation									
Color bank, lookup table/8H (LSB fixed at 00)																					
CMDCOLR +04H	Character address/8H										0	0									
CMDSRCA +06H	0	0	Character size X/8			Character size Y															
CMDSIZE +08H	Character extension		Vertex (A), X coordinate (XA)																		
CMDXA +0AH	Code extension		Vertex (A), Y coordinate (YA)																		
CMDYA +0CH																					
CMDYB +0EH																					
+10H																					
+12H																					
+14H																					
+16H																					
+18H																					
CMDGRDA +1AH	Gouraud shading table/8H																				
+1CH																					

Note:  is ignored.

MO = MON, HS = HSS, Pc = Pclp, Cl = Clip, Cm = Cmod,  
Me = Mesh, EC = CD and SP = SPD.



The normal sprite draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0000B, the normal sprite draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The read direction of the character pattern is set. The character pattern can be drawn with vertical or horizontal inversion by the specification of the read direction.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.



- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh can be enabled, end code can be disabled, and transparent pixel can be disabled. MSB ON can be set to perform shade or window processing in the VDP2.
- The color mode that defines the character pattern is set. Color bank is specified when the color mode is the color bank mode. When the color mode is the lookup table mode, the address of the color lookup table is defined by dividing by 8H.
- Color calculation is specified. Color calculation is enabled in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled in the RGB mode.
- The address of the character pattern table is defined in CMDSRCA by dividing by 8H. The horizontal and vertical lengths of the character size defined in the character pattern table are defined in CMDSIZE. A value divisible by 8 is defined for the horizontal size.
- The upper-left coordinates of the area to be drawn are defined in vertex (A) (CMDXA, CMDYA).

## 7.5 Scaled Sprite Draw Command

The scaled sprite draw command draws character patterns in the frame buffer. Character patterns can be drawn at specified coordinates inverted vertically and / or horizontally, enlarged or reduced, and stretched.

When the end bit is 0B and the command selection bits are 0001B, the scaled sprite draw command is enabled. When the zoom point is 0000B, the character pattern is defined by specifying the coordinates for two points. When the zoom point is other than 0000B, the zoom point is defined.

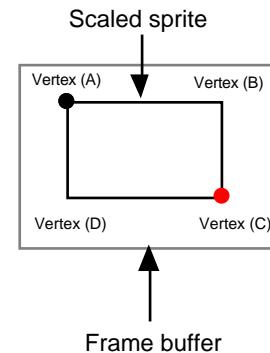
The scaled sprite can not be rotated at 90°. To rotate 90°, specify the distorted sprite draw command.

### Specification of Two Coordinate Points (Scaled Sprite Draw Command)

The contents of the command table for specifying the coordinates of two points are shown in the following figure.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																					
CMDCTRL	+00H	0		JP	0	0	0	0	0	0	0	Dir	0	0	0	1																						
Link specification/8H																																						
CMDLINK	+02H																																					
CMDPMOD	+04H	MO	0	0	HS	Pc	Ci	Cm	Me	EC	SP	Color mode	Color calculation																									
CMDCOLR	+06H	Color bank, lookup table/8H (LSB fixed at 00)																																				
CMDSRCA	+08H	Character address/8H																																				
CMDSIZE	+0AH	0	0	Character size X/8				Character size Y																														
CMDXA	+0CH	Code extension		Vertex (A), X coordinate (XA)																																		
CMDYA	+0EH	Code extension		Vertex (A), Y coordinate (YA)																																		
	+10H																																					
	+12H																																					
CMDXC	+14H	Code extension		Vertex (C), X coordinate (XC)																																		
CMDYC	+16H	Code extension		Vertex (C), Y coordinate (YC)																																		
	+18H																																					
	+1AH																																					
CMDGRDA	+1CH	Gouraud shading table/8H																																				

Note:  is ignored.



Specification of coordinates for two points by the scaled sprite draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0001B, the scaled sprite draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The zoom point is set to 0000B. The coordinates for two points are specified.



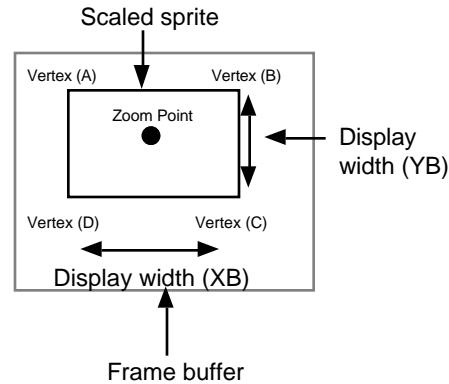
- The read direction of the character pattern is set. The character pattern can be drawn with vertical or horizontal inversion by the specification of the read direction. When inversion is specified twice by the read direction and by the specification of vertex (A) (CMDXA, CMDYA) and vertex (C) (CMDXC, CMDYC), the inversions cancel each other out and the direction returns to the original direction.
- Set high speed shrink. Specify priority to precision or to speed.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.
- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- **Mesh can be enabled, end code can be disabled, and transparent pixel can be disabled. MSB ON can be set to perform shade or window processing in the VDP2.**
- The color mode that defines the character pattern is set. The color bank is specified when the color mode is the color bank mode. When the color mode is the lookup table mode, the address of the color lookup table is defined by dividing by 8H.
- Color calculation is specified. Color calculation is enabled in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled in the RGB mode.
- The address of the character pattern table is defined in CMDSRCA by dividing by 8H. The horizontal and vertical lengths of the character size defined in the character pattern table are defined. A value divisible by 8 is defined for the horizontal size.
- **The start coordinates of the area to be drawn are defined as vertex (A) (CMDXA, CMDYA), and the end coordinates are defined as vertex (C) (CMDXC, CMDYC).** Enlargement, reduction, and stretching are possible by the specification of vertex (A) and vertex (C). Also, when X of (A) is greater than X of (C), the part is inverted horizontally. When Y of (A) is greater than Y of (C), the part is inverted vertically.
- When vertex A and vertex C are set at the same coordinates, they are drawn with one pixel.

## Specification of Zoom Point (Scaled Sprite Draw Command)

The contents of the command table for specifying the zoom point are shown in the following figure.

	bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																													
CMDCTRL <sub>+00H</sub>	0	JP		ZP		0	0	Dir	0	0	0	1																																	
Link specification/8H																																													
CMDLINK <sub>+02H</sub>	MO	0	0	HS	Pc	Ci	Cm	Me	EC	SP	Color mode	Color calculation																																	
Color bank, lookup table/8H (LSB fixed at 00)																																													
CMDCOLR <sub>+04H</sub>											Character address/8H																																		
CMDSRCA <sub>+06H</sub>											0 0																																		
CMDSIZE <sub>+08H</sub>	0	0	Character size X/8			Character size Y																																							
CMDXA <sub>+0AH</sub>	Code extension				Zoom point, X coordinate (XA)																																								
CMDYA <sub>+0CH</sub>	Code extension				Zoom point, Y coordinate (YA)																																								
CMDXB <sub>+0EH</sub>	Code extension				Display, X width (XB)																																								
CMDYB <sub>+10H</sub>	Code extension				Display, Y width (YB)																																								
CMDGRDA <sub>+1AH</sub>																																													
CMDGRDA <sub>+1CH</sub>																																													
Gouraud shading table/8H																																													

Note:  is ignored.



Specification of the zoom point of the scaled sprite draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0001B, the scaled sprite draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The zoom point is specified. When the zoom point is set to other than 0000B, it becomes the display width specification. When 0000B is set, it becomes the coordinate specification for two points.
- For more information on zoom point, refer to “Zoom Point” in section 6.1.
- The read direction of the character pattern is set. The character pattern can be drawn with vertical or horizontal inversion by the specification of the read direction.
- Set high speed shrink. Specify priority to precision or to speed.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.



- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh can be enabled, end code can be disabled, and transparent pixel can be disabled. MSB ON can be set to perform shade or window processing in the VDP2.
- The color mode that defines the character pattern is set. Color bank is specified when the color mode is the color bank mode. When the color mode is the lookup table mode, the address of the color lookup table is defined by dividing by 8H.
- Color calculation is specified. Color calculation is enabled in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled in the RGB mode.
- The address of the character pattern table is defined in CMDSRCA by dividing by 8H. The horizontal and vertical lengths of the character size defined in the character pattern table are defined in CMDSIZE. A value divisible by 8 is defined for the horizontal size.
- The coordinates of the zoom point of the character to be drawn are set at zoom point coordinates (CMDXA, CMDYA). The display width is defined in CMDXB, CMDYB. The character pattern can be drawn enlarged, reduced, or stretched by specifying the display width.
- When the value of CMDXB, CMDYB is set to (0, 0), the sprite is drawn as one pixel.

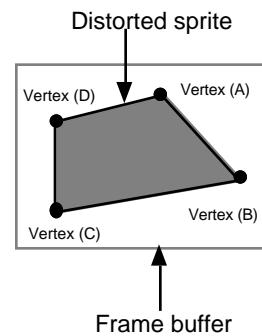
## 7.6 Distorted Sprite Draw Command

The distorted sprite draw command draws character patterns in the frame buffer. Character patterns can be drawn at specified coordinates inverted vertically and / or horizontally, enlarged or reduced, stretched, rotated or twisted.

When the end bit is 0B and the command selection bits are 0010B, the distorted sprite draw command is enabled. The contents of the command table are shown in the following figure.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CMDCTRL	+00H	0	JP	0	0	0	0	0	0	Dir	0	0	1	0				
CMDLINK	+02H														Link specification/8H	0	0	
CMDPMOD	+04H	MO	0	0	HS	Pc	Cl	Cm	Me	EC	SP	Color mode				Color calculation		
CMDCOLR	+06H															Color bank, lookup table/8H (LSB fixed at 00)		
CMDSRCA	+08H															Character address/8H	0	0
CMDSIZE	+0AH	0	0													Character size X/8		Character size Y
CMDXA	+0CH															Code extension	Vertex (A), X coordinate (XA)	
CMDYA	+0EH															Code extension	Vertex (A), Y coordinate (YA)	
CMDXB	+10H															Code extension	Vertex (B), X coordinate (XB)	
CMDYB	+12H															Code extension	Vertex (B), Y coordinate (YB)	
CMDXC	+14H															Code extension	Vertex (C), X coordinate (XC)	
CMDYC	+16H															Code extension	Vertex (C), Y coordinate (YC)	
CMDXD	+18H															Code extension	Vertex (D), X coordinate (XD)	
CMDYD	+1AH															Code extension	Vertex (D), Y coordinate (YD)	
CMDGRDA	+1CH																Gouraud shading table/8H	

Note:  is ignored.



The distorted sprite draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0010B, the distorted sprite draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- The read direction of the character pattern is set. The character pattern can be drawn with vertical or horizontal inversion by the specification of the read direction. When inversion is specified twice by the read direction and by the specification of vertex (A), vertex (B), vertex (C), and vertex (D), the inversions cancel each other out.
- Set high speed shrink. Specify priority to precision or to speed.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.



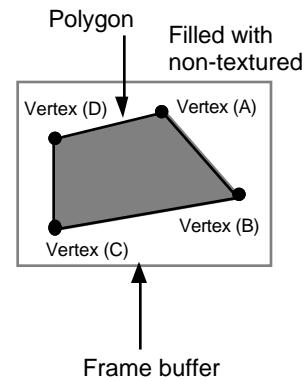
- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh can be enabled, end code can be disabled, and transparent pixel can be disabled. MSB ON can be set to perform shade or window processing in the VDP2.
- The color mode that defines the character pattern is set. Color bank is specified when the color mode is the color bank mode. When the color mode is the lookup table mode, the address of the color lookup table is defined by dividing by 8H.
- Color calculation is specified. Color calculation is enabled in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled in the RGB mode.
- The address of the character pattern table is defined in CMDSRCA by dividing by 8H. The horizontal and vertical lengths of the character size defined in the character pattern table are defined in CMDSIZE. A value divisible by 8 is defined for the horizontal size.
- The start and end coordinates of the first line of the area to be drawn are defined as vertex (A) (CMDXA, CMDYA) and vertex (B) (CMDXB, CMDYB), the start and end coordinates of the last line of the area to be drawn are defined as vertex (D) (CMDXD, CMDYD) and vertex (C) (CMDXC, CMDYC). The relationships between the positions of the four vertices can be set as desired, and therefore enlargement, reduction, stretching, rotation, and twist can be specified when drawing the character pattern by the specification of vertex (A), vertex (B), vertex (C), and vertex (D). Vertical and horizontal inversion are also possible.
- When two vertices are set at the same coordinates, the sprite is drawn as a triangle. When four vertices are set at the same coordinates, the sprite is drawn as one pixel.

## 7.7 Polygon Draw Command

The polygon draw command draws quadrangles in the frame buffer. The inside of the quadrangle is filled with the specified color. When the end bit is 0B and the command selection bits are 0100B, the polygon draw command is enabled. The contents of the command table are shown in the following figure.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL +00H	0	JP	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
CMDLINK +02H																	Link specification/8H
CMDPMOD +04H	MO	0	0	0	Pc	Cl	Cm	Me	1	1	0	0	0				Color calculation
CMDCOLR +06H																	Non-textured color
+08H																	
+0AH																	
CMDXA +0CH	Code extension																Vertex (A), X coordinate (XA)
CMDYA +0EH	Code extension																Vertex (A), Y coordinate (YA)
CMDXB +10H	Code extension																Vertex (B), X coordinate (XB)
CMDYB +12H	Code extension																Vertex (B), Y coordinate (YB)
CMDXC +14H	Code extension																Vertex (C), X coordinate (XC)
CMDYC +16H	Code extension																Vertex (C), Y coordinate (YC)
CMDXD +18H	Code extension																Vertex (D), X coordinate (XD)
CMDYD +1AH	Code extension																Vertex (D), Y coordinate (YD)
CMDGRDA +1CH																	Gouraud shading table/8H

Note:  is ignored.



The polygon draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0100B, the polygon draw command is enabled.
- The jump mode is specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.
- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh can be enabled. MSB ON can be set to perform shade or window processing in the VDP2.
- The color mode is set to 000B. The color code is set to *non-textured color* (CMDCOLR).

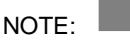


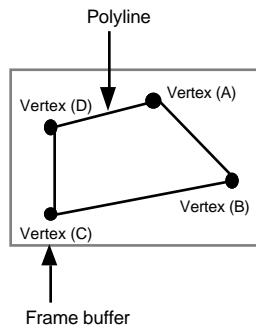
- Specify color calculation. Color calculation is enabled when the non-textured color is in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled when the non-textured color is in the RGB mode.
- The start and end coordinates of the first line of the area to be drawn are defined as vertex (A) (CMDXA, CMDYA) and vertex (B) (CMDXB, CMDYB), the start and end coordinates of the last line of the area to be drawn are defined as vertex (D) (CMDXD, CMDYD) and vertex (C) (CMDXC, CMDYC). The relationships between the positions of the four vertices can be set as desired, and therefore rotation and twist can be specified when drawing the character pattern by the specification of vertex (A), vertex (B), vertex(C), and vertex (D).
- When two vertices are set at the same coordinates, a polygon is drawn as a triangle. When four vertices are set at the same coordinates, the polygon is drawn as one pixel.

## 7.8 Polyline Draw Command

The polyline draw command draws quadrangles in the frame buffer. The inside of the quadrangle is not filled. When the end bit is 0B and the command selection bits are 0101B, the polyline draw command is enabled. The contents of the command table are shown in the following figure.

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	+00 H	0	JP	0	0	0	0	0	0	0	0	0	0	0	1	0	1
CMDLINK	+02 H																0 0
CMDPMOD	+04 H	MO	0	0	0	Pc	C1	Cm	Me	1	1	0	0	0			Color calculation
CMDCOLR	+06 H																Non-textured color
CMDXA	+0A H																
CMDYA	+0C H																Code extension Vertex (A), X coordinate (XA)
CMDXB	+0E H																Code extension Vertex (A), Y coordinate (YA)
CMDYB	+10 H																Code extension Vertex (B), X coordinate (XB)
CMDXC	+12 H																Code extension Vertex (B), Y coordinate (YB)
CMDYC	+14 H																Code extension Vertex (C), X coordinate (XC)
CMDXD	+16 H																Code extension Vertex (C), Y coordinate (YC)
CMDYD	+18 H																Code extension Vertex (D), X coordinate (XD)
CMDGRDA	+1A H																Code extension Vertex (D), Y coordinate (YD)
	+1C H																Gouraud shading table/8 H

NOTE:  is ignored.



The polyline draw command is defined as follows:

- When the end bit is set to 0B and the command selection bits are set to 0101B, the polyline draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.
- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh **can be** enabled. When mesh is enabled, the entire sprite may not be drawn depending on the position and shape of the polyline.
- MSB ON **can be** set to perform shade or window processing in the VDP2.
- The color mode is set to 000B. The color code is set to *non-textured color* (CMDCOLR).



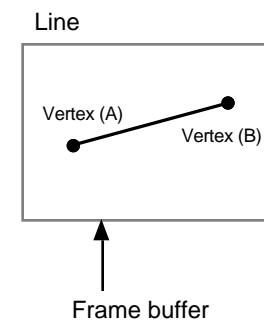
- Color calculation is specified. Color calculation is enabled when the non-textured color is in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled when the non-textured color is in the RGB mode.
- When half-transparency processing is performed on polylines, two lines overlap at each point. As a result half-transparency is applied twice, thus causing original picture 3/4 + background 1/4.
- When shadow processing is performed on polylines, two lines overlap at each point. As a result shadow is applied twice, thus causing one half the luminance.
- The start and end coordinates of the first line of the area to be drawn are defined as vertex (A) (CMDXA, CMDYA) and vertex (B) (CMDXB, CMDYB), the start and end coordinates of the last line of the area to be drawn are defined as vertex (D) (CMDXD, CMDYD) and vertex (C) (CMDXC, CMDYC). The relationships between the positions of the four vertices can be set as desired, and therefore rotation and twist can be specified when drawing the character pattern by the specification of vertex (A), vertex (B), vertex (C), and vertex (D).
- When two vertices are set at the same coordinates, a polygon is drawn as a triangle. When four vertices are set at the same coordinates, the polygon is drawn as one pixel.

## 7.9 Line Draw Command

The line draw command draws straight lines that connect two points. The inside of the quadrangles is not filled. When the end bit is 0B and the command selection bits are 0110B, the line draw command is enabled. The contents of the command table are shown below.

	bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																												
CMDCTRL +00H	0 JP 0 0 0 0 0 0 0 0 0 0 0 1 1 0																												
CMDLINK +02H	Link specification/8H 0 0																												
CMDPMOD +04H	MO 0 0 0 Pc Cl Cm Me 1 1 0 0 0 Color calculation																												
CMDCOLR +06H	Non-textured color																												
+08H																													
CMDXA +0AH	Code extension	Vertex (A), X coordinate (XA)																											
CMDYA +0CH	Code extension	Vertex (A), Y coordinate (YA)																											
CMDXB +0EH	Code extension	Vertex (B), X coordinate (XB)																											
CMDYB +10H	Code extension	Vertex (B), Y coordinate (YB)																											
+12H																													
+14H																													
+16H																													
+18H																													
+1AH	Gouraud shading table/8H																												
+1CH																													

Note:  is ignored.



The line draw command is defined as follows.

- When the end bit is set to 0B and the command selection bits are set to 0110B, the line draw command is enabled.
- The jump mode is then specified. When the jump mode is assigned or called, the address of the command table to be processed next is divided by 8H and set in CMDLINK.
- Set pre-clipping. Specify enable or disable in consideration of the clipping area and the drawing position of the part.
- The user clipping enable bit and clipping mode are specified. Depending on the clipping mode, either the outside or the inside of the user clipping area is specified.
- Mesh can be enabled. When mesh is enabled, the entire sprite may not be drawn depending on the position and shape of the line.
- MSB ON can be set to perform shade or window processing in the VDP2.
- The color mode is set to 000B. The color code is set to *non-textured color* (CMDCOLR).



- Color calculation is specified. Color calculation is enabled when the non-textured color is in the RGB mode. When color calculation is not used, replace is specified. When Gouraud shading is used, the address of the Gouraud shading table is specified by dividing by 8H. Gouraud shading processing is enabled when the non-textured color is in the RGB mode. Only vertex (A) and vertex (B) in the Gouraud shading table are enabled; vertex (C) and vertex (D) are ignored.
- Vertex (A) of the line to be drawn is defined in CMDXA, CMDYA and vertex (B) is defined in CMDXB, CMDYB.
- When the two vertices are set to the same coordinates, the line is drawn as one pixel.

## 7.10 Draw End Command

The draw end command specifies the end of drawing. When the end bit is “1”, the draw end command is enabled. The contents of the command table are shown in the following figure.

CMDCTRL	bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+00H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
+02H																
+04H																
+06H																
+08H																
+0AH																
+0CH																
+0EH																
+10H																
+12H																
+14H																
+16H																
+18H																
+1AH																
+1CH																

Note:  is ignored.

Gouraud shading table/8H

- When the +00H address is set to 8000H, the draw end command is enabled.



# **Chapter 8**

## **Quick Reference**

### **Contents**

8.1	System Registers .....	134
8.2	Tables .....	140
8.3	Command Tables .....	142
8.4	Commands .....	151

## 8.1 System Registers

### TV Mode Selection Register (TVMR)

TVMR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100000H		0	0	0	0	0	0	0	0	0	0	0	0	VBE	TVM		
Write-only																	

V-blank Erase/Write Enable (VBE): bit 3

VBE	FCM	FCT	Change mode	Change time
0	0	0	One cycle mode	Changes every 1/60 second
0	0	1	Setting prohibited	—
0	1	0	Manual mode (erase)	Erase in next field
0	1	1	Manual mode (change)	Change in next field
1	0	0	Setting prohibited	—
1	0	1	Setting prohibited	—
1	1	0	Setting prohibited	—
1	1	1	Manual mode (erase & change)	Erase with V-blank, change in next field

TV Mode Select (TVM): bits 2 - 0

Table 8.1 Screen Modes

TVM			Screen Mode		Bit width (bit/pixel)	Frame buffer screen size	Interlace	VDP CLK (MHz)	
Bits 2	1	0	Frame buffer name	Possible VDP2 resolutions H x V				NTSC	PAL
0	0	0	Normal (NTSC, PAL)	320x224	16	512 H x 256 V	Yes	26.8426	26.6564
				320x240				26.8426	26.6564
				352x224				28.6364	28.4375
				352x240				28.6364	28.4375
				640x224				26.8426	26.6564
				640x240				26.8426	26.6564
				704x224				28.6364	28.4375
				704x240				28.6364	28.4375
0	0	1	High resolution (NTSC, PAL, Hi-Res)	640x224	8	1024 H x 256 V	Yes	26.8426	26.6564
				640x240				26.8426	26.6564
				704x224				28.6364	28.4375
				704x240				28.6364	28.4375
0	1	0	Rotation 16 (NTSC, PAL Rotation)	320x224	16	512 H x 256 V	Single Only	26.8426	26.6564
				320x240				26.8426	26.6564
				352x224				28.6364	28.4375
				352x240				28.6364	28.4375
				640x224				26.8426	26.6564
				640x240				26.8426	26.6564
				704x224				28.6364	28.4375
				704x240				28.6364	28.4375
0	1	1	Rotation 8 (NTSC, PAL Rotation)	320x224	8	512 H x 512 V	Single Only	26.8426	26.6564
				320x240				26.8426	26.6564
				352x224				28.6364	28.4375
				352x240				28.6364	28.4375
1	0	0	HDTV (31KC, HDTV)	640x224	16	512 H x 256 V	No	26.8426	26.6564
				640x240				28.6364	28.4375
				704x224				28.6364	28.4375
				704x240				28.6364	28.4375
All other			Setting prohibited (do not use)						

If VDP2 is high-resolution, and the frame buffer horizontal size is 512, then each frame buffer pixel is displayed on 2 horizontal pixels on screen.



## Frame Buffer Change Mode Register

FBCR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100002H		0	0	0	0	0	0	0	0	0	0	0	EOS	DIE	DIL	FCM	FCT
Write-only																	

Frame Buffer Change Mode (FCM): bit 1

Frame Buffer Change Trigger (FCT): bit 0

VBE	FCM	FCT	Change mode	Change time
0	0	0	1 cycle mode	Changes every 1/60 second
0	0	1	Setting prohibited	—
0	1	0	Manual mode (erase)	Erase in next field
0	1	1	Manual mode (change)	Change in next field
1	0	0	Setting prohibited	—
1	0	1	Setting prohibited	—
1	1	0	Setting prohibited	—
1	1	1	Manual mode (erase & change)	Erase with V-blank and change in next field

Double Interlace Enable (DIE): bit 3

Double Interlace Draw Line (DIL): bit 2

DIE	DIL	Interlace mode	Draw after next frame change
0	0	Non-interlace/single interlace	Draw all lines
0	1	Setting prohibited	—
1	0	Double-interlace	Only even numbered (EVEN) lines drawn
1	1	Double-interlace	Only odd numbered (ODD) lines drawn

Even/Odd Coordinate Select Bit (EOS): bit 4

EOS	Even/odd coordinate select bit
0	Samples only pixels at even coordinates
1	Samples only pixels at odd coordinates

## Plot Trigger Register

PTMR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	PTM
100004H		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PTM	

Write-only

Plot Trigger Mode (PT): bits 1, 0

PTM		Drawing mode
Bit 1	0	
0	0	Idle at frame change
0	1	Starts drawing when 01B is written
1	0	Starts drawing automatically with frame change
1	1	Setting prohibited (do not set)

## Erase/Write Data Register

EWDR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100006H																	
16 bits/pixel		Erase/write Data															
8 bits/pixel		Erase/write data for even X coordinates								Erase/write data for odd X coordinates							
Write-only																	

Erase/Write Data: bits 15~0



## Erase/Write Upper-Left Coordinate Register

EWLR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100008H		0															
Write-only																	

Erase/Write Upper-Left Coordinate X1: bits 14~9

Erase/Write Upper-Left Coordinate Y1: bits 8~0

## Erase-Write Lower-Right Coordinate Register

EWRR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10000AH																	
Write-only																	

Erase/Write Lower-Right Coordinate X3: bits 15~9

Erase/Write Lower-Right Coordinate Y3: bits 8~0

Value set in register	16 bits/pixel		8 bits/pixel					
			High Resolution		Rotated 8			
Upper-left coordinate X1	Lower-right coordinate X3	Upper-left coordinate X1						
0	0	Setting prohibited						
1	8	7	16	15	16	15	16	15
2	16	15	32	31	32	31	32	31
:	:	:	:	:	:	:	:	:
31	248	247	496	495	496	495	496	495
32	256	255	512	511	Setting prohibited	Setting prohibited	511	Setting prohibited
33	264	263	528	527	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited
:	:	:	:	:	:	:	:	:
40	320	319	640	639	:	:	:	:
:	:	:	:	:	:	:	:	:
43	344	343	688	687	:	:	:	:
44	352	351	704	703	:	:	:	:
:	:	:	:	:	:	:	:	:
62	496	495	992	991	:	:	:	:
63	504	503	1008	1007	:	:	:	:
64	Setting prohibited	511	Setting prohibited	1023	:	:	:	:
65 and over	Setting prohibited	Setting prohibited						

### Draw Forced Termination Register

ENDR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10000CH		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write-only																	

### Transfer End Status Register

EDSR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100010H		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read-only																	

- Current End Bit Fetch Status (CEF): bit 1

CEF	End bit fetch status
0	The end bit in current frame has not been fetched.
1	The end bit in current frame has been fetched and that drawing is terminated.

- Before End Bit Fetch Status (BEF): bit 0

BEF	End bit fetch status
0	The end bit in previous frame has not been fetched.
1	The end bit in previous frame has been fetched and that drawing is terminated.

### Last Operation Table Address Register

LOPR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100012H																0	0
Read-only																	

- Last Operation Table Address: bits 15~0

### Current Operation Table Address

COPR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100014H																0	0
Read-only																	

- Current Operation Table Address: bits 15~0



## Mode Status Register

MODR	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100016H				VER	—	—	—	PTM1	EOS	DIE	DIL	FCM	VBE				
Read-only																	

**Version Number (VER): bits 15~12**

**Plot Trigger Mode 1 (PTM1): bit 8**

**Even/Odd Coordinate Select Bit (EOS): bit 7**

**Double Interlace Enable Bit (DIE): bit 6**

**Double Interlace Draw Line (DIL): bit 5**

**Frame Buffer Change Mode Bit (FCM): bit 4**

**V-Blank Erase/Write Enable Bit (VBE): bit 3**

**TV Mode Selection Bits (TVM): bits 2~0**

## 8.2 Tables

### Character Pattern Tables

For 4 bits/pixel, 0CH (12) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+	0	+	1	+	2	+	3
+04H	+	4	+	5	+	6	+	7
+08H	+	8	+	9	+	A	+	B

← Value is local address from character pattern

For 8 bits/pixel, 18H (24) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+0	+1	+2	+3	+4	+5	+6	+7
+08H	+8	+9	+A	+B	+C	+D	+E	+F
+10H	+10	+11	+12	+13	+14	+15	+16	+17

For 16 bits/pixel, 30H (48) bytes is required.

pixel	0	1	2	3	4	5	6	7
+00H	+0	+1	+2	+3	+4	+5	+6	+7
+10H	+10	+11	+12	+13	+14	+15	+16	+17
+20H	+20	+21	+22	+23	+24	+25	+26	+27
	+28	+29	+2A	+2B	+2C	+2D	+2E	+2F

Figure 8.1 Examples of Character Pattern Tables

### Color Lookup Table

+00H	16-bit data	(color code of 0H)
+02H	16-bit data	(color code of 1H)
+04H	16-bit data	(color code of 2H)
:	:	
+1CH	16-bit data	(color code of EH)
+1EH	16-bit data	(color code of FH)

Figure 8.2 Color Lookup Table



## Gouraud Shading Table

**Table 8.2 Gouraud Shading Table**

Table address	Corresponding vertex	
	Sprites, polygons, polylines	Lines
Table top address	Vertex (A)	Line start vertex
Table top address + 2	Vertex (B)	Line end vertex
Table top address + 4	Vertex (C)	Ignored
Table top address + 6	Vertex (D)	Ignored

### 8.3 Command Tables

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CMDCTRL +00H	END	JP			ZP			0	0	Dir	Comm						
CMDLINK +02H	LINK specification/8H													0	0		
CMDPMOD +04H	MON	0	0	HSS	Pclip	Clip	Cmod	Mesh	ECD	SPD	Color mode			Color calculation bit			
CMDCOLR +06H	Color bank, color lookup table/8H (LSB is set to 00), non-textured color																
CMDSRCRA +08H	Character address/8H													0	0		
CMDSIZE +0AH	0	0	Character size X/8					Character size Y									
CMDXA +0CH	Code extension					Vertex (A) and X coordinate (XA) *											
CMDYA +0EH	Code extension					Vertex (A) and Y coordinate (YA)											
CMDXB +10H	Code extension					Vertex (B) and X coordinate (XB)											
CMDYB +12H	Code extension					Vertex (B) and Y coordinate (YB)											
CMDXC +14H	Code extension					Vertex (C) and X coordinate (XC)											
CMDYC +16H	Code extension					Vertex (C) and Y coordinate (YC)											
CMDXD +18H	Code extension					Vertex (D) and X coordinate (XD)											
CMDYD +1AH	Code extension					Vertex (D) and Y coordinate (YD)											
CMDGRDA +1CH	Gouraud Shading Table/8H																
+1EH	(Dummy) Skipped during table fetch																
+20H	:																
:	Succeeding table																
+40H	:																
:	Succeeding table																
+60H	:																
:	:																

\* Note: The top bit of the vertex coordinate is a sign bit. A negative value is indicated by the complement of 2. Extend the sign to the upper 5 bits.

Figure 8.3 Command Table



## CMDCTRL (Control Words)

CMDCTRL +00H	bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	END		JP		ZP		0	0	Dir		Comm					

### End bit (END): bit 15

END	End bit
0	Drawing command or coordinate setting command (other than draw end command)
1	Draw end command

### Jump Mode (JP): bits 14~12

Bit 14	JP		Jump mode	Processing
	13	12		
0	0	0	Jump next	Automatically jumps to next table (address +20H) after this table is processed (CMDLINK is ignored).
0	0	1	Jump assign	Jumps to CMDLINK table after this table is processed.
0	1	0	Jump call	CMDLINK table receives subroutine call after this table is processed.
0	1	1	Jump return	Returns to main routine after this table is processed (CMDLINK is ignored).
1	0	0	Skip next	Jumps to next table (address +20H) without processing this table (CMDLINK is ignored).
1	0	1	Skip assign	Jumps to CMDLINK without processing this table.
1	1	0	Skip call	CMDLINK table receives subroutine call without processing this table.
1	1	1	Skip return	Returns to main routine without processing this table (CMDLINK is ignored).

**Zoom Point (ZP): bits 11~8**

ZP				Code	Zoom point
Bit 11	10	9	8		
0	0	0	0	0H	Specifies two coordinates
0	1	0	1	5H	Upper-left
0	1	1	0	6H	Upper-center
0	1	1	1	7H	Upper-right
1	0	0	1	9H	Center-left
1	0	1	0	AH	Center-center
1	0	1	1	BH	Center-right
1	1	0	1	DH	Lower-left
1	1	1	0	EH	Lower-center
1	1	1	1	FH	Lower-right
Other than above				Setting prohibited (do not set)	

**Character Read Direction (Dir): bits 5 and 4**

Dir		Inversion processing
Y	X	
0	0	Not inverted
0	1	Inverted horizontally
1	0	Inverted vertically
1	1	Inverted vertically and horizontally



### Command Select (Comm): bits 3~0

**Table 8.3 Commands**

END	Comm				Function	Command
Bit 15	3	2	1	0		
0	0	0	0	0		Textured draw command
		0	1	0		Scaled sprite draw command
		1	0	0		Distorted sprite draw command
		1	0	0		Polygon draw command
		0	1	0		Polyline draw command
		1	0	0		Line draw command
	1	0	0	0	Coordinates set commands	User clipping coordinates set command
		0	0	1		System clipping coordinates set command
		1	0	0		Local coordinate set command
1	0	0	0	0	End draw command	
All other codes					Setting prohibited (do not set)	

### CMDLINK (Link Specification)

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CMDLINK +02H																0	0

## CMDPMD (Draw Mode Word)

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDPMD +04H	MON	0	0	HSS	Pclp	Clip	Cmod	Mesh	ECD	SPD	Color mode	Color calculation bits				

### High Speed Shrink (HSS): bit 12

HSS	Processing
0	High speed shrink disabled
1	High speed shrink enabled

### Pre-clipping Disable (Pclp): bit 11

Pclp	Processing
0	Pre-clipping with horizontal inversion
1	No pre-clipping and no horizontal inversion

### User Clipping Enable Bit (Clip): bit 10

### Clipping Mode Bit (Cmod): bit 9

Clip	Cmod	User clipping processing
0	0	User clipping disabled
0	1	Setting prohibited (do not set)
1	0	Inside drawing mode
1	1	Outside drawing mode

### Mesh Enable Bit (Mesh): bit 8

Mesh	Mesh enable
0	Draw without mesh processing
1	Draw with mesh processing



**End Code Disable (ECD): bit 7**

HSS	ECD	End code processing
0	0	End code enable: drawing horizontally is disabled when second end code is read and end code becomes transparent.
0	1	End code disable: end code is not processed, color of code is expressed.
1 & enlarge	0	End code enable: drawing horizontally is disabled when second end code is read and end code becomes transparent.
1 & reduce	0	End code disable: end code is not processed, color of code is expressed.
1	1	End code disable: end code is not processed, color of code is expressed.

**Transparent Pixel Disable (SPD): bit 6**

SPD	Transparent code processing	
0	Transparent pixel enable:	transparent color codes are not drawn; transparent color codes become transparent.
1	Transparent pixel disable:	transparent color code is processed and drawn like other color codes.

**Color Mode Bits: bits 5~3**

Color mode			Mode	Description		Bits per pixel
Bit 5	4	3		Number of colors	Mode	
0	0	0	0	16	Color bank mode	4 bits
0	0	1	1	16	Lookup table mode	4 bits
0	1	0	2	64	Color bank mode	8 bits
0	1	1	3	128	Color bank mode	8 bits
1	0	0	4	256	Color bank mode	8 bits
1	0	1	5	32,768	RGB mode	16 bits
Other than above				Setting prohibited (do not set)		

**Color Calculation Bits: bits 2–0**

Bit	Function
2	Gouraud shading enable bit
1	1/2 original graphic enable bit
0	1/2 background enable bit

Color calculation (in Bit)			Background MSB	Original graphic	Background	Type of color calculation	Usable modes	
2	1	0					Original graphic	Background
0	0	0	—	1	0	Replace	Not restricted	Not restricted
0	0	1		0	1	Cannot rewrite	Not restricted	Palette
				0 <sup>1</sup>	1/2	Shadow		RGB
0	1	0	—	1/2	0	Half-luminance	RGB	Not restricted
0	1	1	—	1	0	Replace	RGB	Palette
				1/2	1/2	Half-transparent		RGB
1	0	0	—	Gouraud	0	Gouraud shading	RGB	Not restricted
1	0	1	—	—	—	Setting prohibited (do not set)	—	—
1	1	0	—	Gouraud 1/2	0	Gouraud shading + Half-luminance <sup>2</sup>	RGB	Not restricted
1	1	1		0	0	Gouraud shading	RGB	Palette
				Gouraud 1/2	1/2	Gouraud shading + Half-transparent <sup>3</sup>		RGB

**Notes**

—: doesn't matter

Original graphic: sprite or pixel data to be plotted in non-textured color.

Background: pixel data already plotted in the frame buffer.

<sup>1</sup>Original graphic (transparent pixels, end code) is referenced.<sup>2</sup>Data that has undergone saturation processing after Gouraud calculation is reduced by half.<sup>3</sup>Background is added to data that has undergone saturation processing after Gouraud calculation is reduced by half.**MON Bit (MSB on): bit 15**

MON	Processing
0	MSB of pixel data in frame buffer is not changed
1	Sets MSB of pixel data in frame buffer to 1



## CMDCOLR (Color Control Word)

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCOLR +06H																

Color bank, color lookup table/8H (LSB fixed at 00), non-textured colors

**Table 8.4 CMDCOLR**

Part	Color mode	Color control word
Textured part	Color bank mode	Color bank
	Lookup table mode	Color lookup table address
	RGB mode	Ignored
Non-textured part	Non-textured color	

## CMDSRCA (Character Address)

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSRCA +08H																0 0

Character address/8H

## CMDSIZE (Character Size)

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSIZE +0AH	0	0														Character size Y

Character size X/8

**Table 8.5 Relationships between Settings and Drawn Pixels**

Horizontal (X) direction		Vertical (Y) direction	
Setting in command table	Number of pixels actually drawn	Setting in command table	Number of pixels actually drawn
0	Setting prohibited	0	Setting prohibited
1	8	1	1
2	16	2	2
:	:	:	:
63	504	255	255

**CMDXA~CMDYD (Vertex Coordinate Data)**

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDXA +0CH	Code extension															Vertex (A), X coordinates (XA)
CMDYA +0EH	Code extension															Vertex (A), Y coordinates (YA)
CMDXB +10H	Code extension															Vertex (B), X coordinates (XA); or display, X width (XB)
CMDYB +12H	Code extension															Vertex (B), Y coordinates (YA); or display, Y width (YB)
CMDXC +14H	Code extension															Vertex (C), X coordinates (XC)
CMDYC +16H	Code extension															Vertex (C), Y coordinates (YC)
CMDXD +18H	Code extension															Vertex (D), X coordinates (XD)
CMDYD +1AH	Code extension															Vertex (D), Y coordinates (YD)

**Note:** The top bit of the parameter is a sign bit. A negative value is indicated by a complement of 2. Extend the code to the upper 5 bits.

**Table 8.6 Correspondence between Commands and CMDXA-CMDYD**

Command	CMDXA CMDYA	CMDXB CMDYB	CMDXC CMDYC	CMDXD CMDYD
Normal sprite draw command	Vertex (A)	—	—	—
Scaled sprite draw command (two methods)	Specify coordinates for two points	Vertex (A)	—	Vertex (C)
	Specify zoom point	Zoom point coordinates	Display width	—
Distorted sprite draw command	Vertex (A)	Vertex (B)	Vertex (C)	Vertex (D)
Polygon draw command	Vertex (A)	Vertex (B)	Vertex (C)	Vertex (D)
Polyline draw command	Vertex (A)	Vertex (B)	Vertex (C)	Vertex (D)
Line draw command	Vertex (A)	Vertex (B)	—	—
User clipping coordinate set command	Upper-left coordinates	—	Lower-right coordinates	—
System clipping coordinate set command	—	—	Lower-right coordinates	—
Local coordinate set command	Local coordinates	—	—	—

**Note:** "—" indicates unused.

**CMDGRDA (Gouraud Shading Table)**

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDGRDA +1CH	Gouraud shading table/8H															

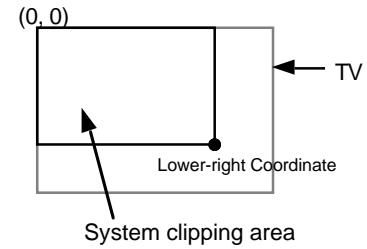


## 8.4 Commands

### System Clipping Coordinate Set Command

	bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CMDCTRL +00H	0 JP 0 0 0 0 0 0 0 0 0 0 1 0 0 1
CMDLINK +02H	Link specification/8H 0 0
+04H	
+06H	
+08H	
+0AH	
+0CH	
+0EH	
+10H	
+12H	
CMDXC +14H	0 0 0 0 0 0 Lower-right X coordinate (XC)
CMDYC +16H	0 0 0 0 0 0 Lower-right Y coordinate (YC)
+18H	
+1AH	
+1CH	

Note:  is ignored.

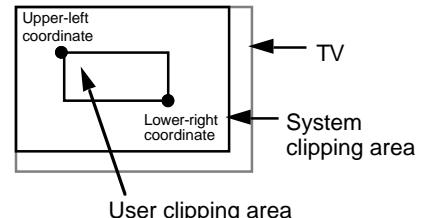


Upper-left coordinates  
are fixed at (0, 0).

### User Clipping Coordinate Set Command

	bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CMDCTRL +00H	0 JP 0 0 0 0 0 0 0 0 0 0 1 0 0 0
CMDLINK +02H	Link specification/8H 0 0
+04H	
+06H	
+08H	
+0AH	
CMDXA +0CH	0 0 0 0 0 0 Upper-left X coordinate (XA)
CMDYA +0EH	0 0 0 0 0 0 Upper-left Y coordinate (YA)
+10H	
+12H	
CMDXC +14H	0 0 0 0 0 0 Lower-right X coordinate (XC)
CMDYC +16H	0 0 0 0 0 0 Lower-right Y coordinate (YC)
+18H	
+1AH	
+1CH	

Note:  is ignored.

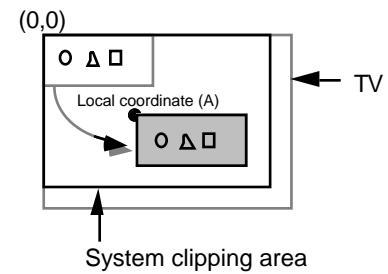


User clipping area

## Local Coordinate Set Command

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	+00H	0	JP	0	0	0	0	0	0	0	1	0	1	0			
CMDLINK	+02H																Link specification/8H
	+04H																
	+06H																
	+08H																
	+0AH																
CMDXA	+0CH																Code extension Local X coordinate (XA)
CMDYA	+0EH																Code extension Local Y coordinate (YA)
	+10H																
	+12H																
	+14H																
	+16H																
	+18H																
	+1AH																
	+1CH																

Note:  is ignored.

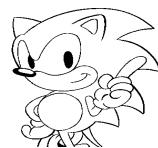
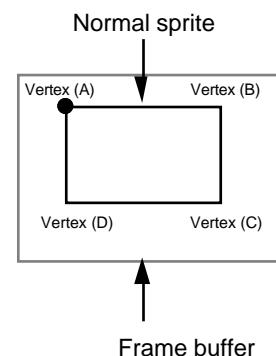


## Normal Sprite Draw Command

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	+00H	0	JP	0	0	0	0	0	Dir	0	0	0	0				
CMDLINK	+02H																Link specification/8H
CMDPMOD	+04H	MO	0	0	0	Pc	A	Cm	Me	EC	SP	Color mode					
CMDCOLR	+06H																Color bank, lookup table/8H (LSB fixed at 00)
CMDSRCA	+08H																Character address/8H
CMDSIZE	+0AH	0	0														Character size X/8 Character size Y
CMDXA	+0CH																Code extension Vertex (A), X coordinate (XA)
CMDYA	+0EH																Code extension Vertex (A), Y coordinate (YA)
	+10H																
	+12H																
	+14H																
	+16H																
	+18H																
	+1AH																Gouraud shading table/8H
	+1CH																

Note:  is ignored.

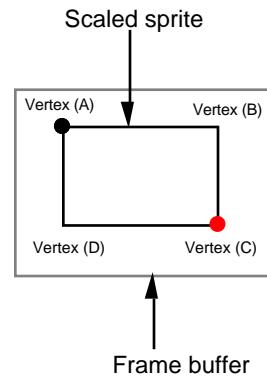
MO = MON, HS = HSS, Pc = Pclip,  
 Cl = Clip, Cm = Cmod,  
 Me = Mesh, EC = CD and



## Scaled Sprite Draw Command (Coordinates of 2 Points Specification )

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	+00H	0	JP	0	0	0	0	0	0	Dir	0	0	0	1	0	0	0
Link specification/8H																	
CMDLINK	+02H	MO	0	0	HS	Pc	Cl	Cm	Me	EC	SP	Color mode	Color calculation				
Color bank, lookup table/8H (LSB fixed at 00)																	
CMDPMOD	+04H	Character address/8H															
CMDCOLR	+06H	0 0 Character size X/8 Character size Y															
CMDSRCA	+08H	Character extension Vertex (A), X coordinate (XA)															
CMDSIZE	+0AH	Code extension Vertex (A), Y coordinate (YA)															
CMDXA	+0CH	+10H															
CMDYA	+0EH	Code extension Vertex (C), X coordinate (XC)															
CMDXC	+12H	Code extension Vertex (C), Y coordinate (YC)															
CMDYC	+14H	+16H															
CMDGRDA	+1AH	Gouraud shading table/8H															
	+1CH																

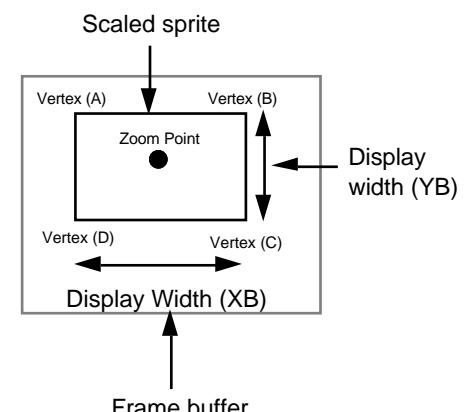
Note:  is ignored.



## Scaled Sprite Draw Command (Specification of Zoom Point )

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDCTRL	+00H	0	JP		ZP			0	0	Dir	0	0	0	1	0	0	0
Link Specification/8H																	
CMDLINK	+02H	MO	0	0	HS	Pc	Cl	Cm	Me	EC	SP	Color mode	Color calculation				
Color bank, lookup table/8H (LSB fixed at 00)																	
CMDPMOD	+04H	Character address/8H															
CMDCOLR	+06H	0 0 Character size X/8 Character size Y															
CMDSRCA	+08H	Character extension Zoom point (A), X coordinate (XA)															
CMDSIZE	+0AH	Code extension Zoom point (A), Y coordinate (YA)															
CMDXA	+0CH	CMDXB															
CMDYA	+0EH	Display, X width (XB)															
CMDXB	+10H	CMDYB															
CMDYB	+12H	Display, Y width (YB)															
	+14H																
	+16H																
	+18H																
	+1AH	Gouraud shading table/8H															
	+1CH																

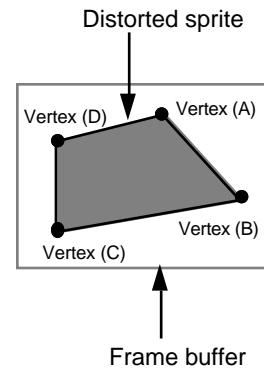
Note:  is ignored.



## Distorted Sprite Draw Command

	bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CMDCTRL <sub>+00H</sub>	0 JP 0 0 0 0 0 0 Dir 0 0 1 0
CMDLINK <sub>+02H</sub>	Link specification/8H 0 0
CMDPMOD <sub>+04H</sub>	MO 0 0 HS P <sub>c</sub> Cl C <sub>m</sub> Me EC SP Color mode Color calculation
CMDCOLR <sub>+06H</sub>	Color bank, lookup table/8H (LSB fixed at 00)
CMDSRCA <sub>+08H</sub>	Character address/8H 0 0
CMDSIZE <sub>+0AH</sub>	0 0 Character size X/8 Character size Y
CMDXA <sub>+0CH</sub>	Code extension Vertex (A), X coordinate (XA)
CMDYA <sub>+0EH</sub>	Code extension Vertex (A), Y coordinate (YA)
CMDXB <sub>+10H</sub>	Code extension Vertex (B), X coordinate (XB)
CMDYB <sub>+12H</sub>	Code extension Vertex (B), Y coordinate (YB)
CMDXC <sub>+14H</sub>	Code extension Vertex (C), X coordinate (XC)
CMDYC <sub>+16H</sub>	Code extension Vertex (C), Y coordinate (YC)
CMDXD <sub>+18H</sub>	Code extension Vertex (D), X coordinate (XD)
CMDYD <sub>+1AH</sub>	Code extension Vertex (D), Y coordinate (YD)
CMDGRDA <sub>+1CH</sub>	Gouraud shading table/8H

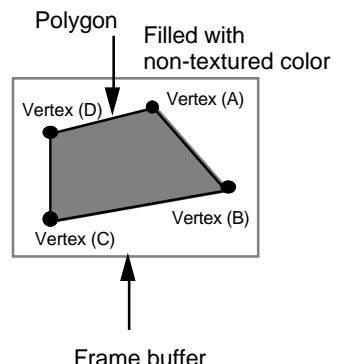
Note:  is ignored.



## Polygon Draw Command

	bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CMDCTRL <sub>+00H</sub>	0 JP 0 0 0 0 0 0 0 0 0 0 0 1 0 0
CMDLINK <sub>+02H</sub>	Link specification/8H 0 0
CMDPMOD <sub>+04H</sub>	MO 0 0 0 P <sub>c</sub> Cl C <sub>m</sub> Me 1 1 0 0 0 Color calculation
CMDCOLR <sub>+06H</sub>	Non-textured color
CMDXA <sub>+0AH</sub>	Code extension Vertex (A), X coordinate (XA)
CMDYA <sub>+0CH</sub>	Code extension Vertex (A), Y coordinate (YA)
CMDXB <sub>+10H</sub>	Code extension Vertex (B), X coordinate (XB)
CMDYB <sub>+12H</sub>	Code extension Vertex (B), Y coordinate (YB)
CMDXC <sub>+14H</sub>	Code extension Vertex (C), X coordinate (XC)
CMDYC <sub>+16H</sub>	Code extension Vertex (C), Y coordinate (YC)
CMDXD <sub>+18H</sub>	Code extension Vertex (D), X coordinate (XD)
CMDYD <sub>+1AH</sub>	Code extension Vertex (D), Y coordinate (YD)
CMDGRDA <sub>+1CH</sub>	Gouraud shading table/8H

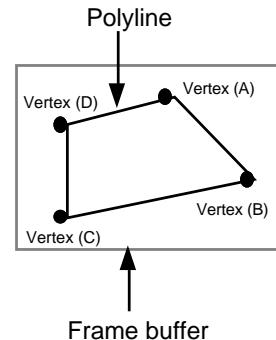
Note:  is ignored.



## Polyline Draw Command

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
CMDCTRL	+00H	0	JP	0	0	0	0	0	0	0	0	0	0	0	1	0	1			
CMDLINK	+02H	Link specification/8H												0	0					
CMDPMOD	+04H	MO	0	0	0	Pc	Ci	Cm	Me	1	1	0	0	0	Color calculation					
CMDCOLR	+06H	Non-textured color																		
CMDXA	+0AH																			
CMDYA	+0CH	Code extension		Vertex (A), X coordinate (XA)																
CMDXB	+0EH	Code extension		Vertex (A), Y coordinate (YA)																
CMDYB	+10H	Code extension		Vertex (B), X coordinate (XB)																
CMDXC	+12H	Code extension		Vertex (B), Y coordinate (YB)																
CMDYC	+14H	Code extension		Vertex (C), X coordinate (XC)																
CMDXD	+16H	Code extension		Vertex (C), Y coordinate (YC)																
CMDYD	+18H	Code extension		Vertex (D), X coordinate (XD)																
CMDGRDA	+1AH	Code extension		Vertex (D), Y coordinate (YD)																
	+1CH	Gouraud shading table/8H																		

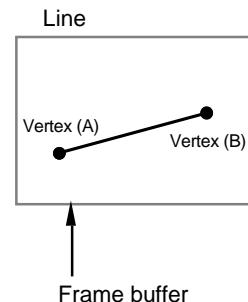
NOTE:  is ignored.



## Line Draw Command

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
CMDCTRL	+00H	0	JP	0	0	0	0	0	0	0	0	0	0	0	1	1	0																	
CMDLINK	+02H	Link specification/8H												0	0																			
CMDPMOD	+04H	MO	0	0	0	Pc	Ci	Cm	Me	1	1	0	0	0	Color calculation																			
CMDCOLR	+06H	Non-textured color																																
CMDXA	+0AH																																	
CMDYA	+0CH	Code extension		Vertex (A), X coordinate (XA)																														
CMDXB	+0EH	Code extension		Vertex (A), Y coordinate (YA)																														
CMDYB	+10H	Code extension		Vertex (B), X coordinate (XB)																														
CMDYC	+12H	Code extension		Vertex (B), Y coordinate (YB)																														
CMDXD	+14H																																	
CMDYD	+16H																																	
CMDGRDA	+1AH																																	
	+1CH	Gouraud shading table/8H																																

Note:  is ignored.



## Draw End Command

CMDCTRL	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+00H		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
+02H																	
+04H																	
+06H																	
+08H																	
+0AH																	
+0CH																	
+0EH																	
+10H																	
+12H																	
+14H																	
+16H																	
+18H																	
+1AH																	
+1CH																	

Note: █ is ignored.



## Chapter 9

### Precautions for Use

SEGA Confidential

**VDP1**

- Rotated reading of the frame buffer is prohibited if TVM is set to normal high resolution or HDTV, or if double interlace is enabled (DIE).,
- The CPU can only access the draw frame buffer.

**System Registers**

- DMA access of the system registers is prohibited. Word access must be used.
- Be sure to set the system registers after powering on.
- Be sure to set bits not used in the system registers to "0."
- Be sure to set DIE to "0" for rotated reading of the frame buffer. Rotated reading is prohibited in double interlace.
- An undefined screen is displayed in the first frame after changing the DIE bit, so use caution.
- When changing the TVM bit, the frame buffer must be masked. An undefined screen is displayed for one frame during the change.
- Set FCM and FCT to "0" during double interlace. The fields can only be changed when they are 1/60 second.
- Set the erase/write areas so that upper-left coordinate XA is less than lower-right coordinate XC and upper-left coordinate YA is less than or equal to lower-right coordinate YC.

**Commands**

- Local coordinates must be set after powering on.
- The clipping coordinate setting range is from (0, 0) to (1023, 511).
- In the clipping coordinate range, upper-left coordinate XA is less than lower-right coordinate XC and upper-left coordinate YA is less than or equal to lower-right coordinate YC.
- Set suitable values for clipping to match the screen mode set by the TVM bit.
- Always set the user clipping area inside the system clipping area. They can be set on the same lines.
- The upper-left coordinate of the system clipping area is fixed at (0,0) in the hardware.
- Set unused bits in words used in commands tables to "0."
- Call (call, jump call, skip call) is prohibited in command subroutines.



- Return (return, jump return, skip return) is prohibited in command main routines.
- Only 0H, 5H, 6H, 7H, 9H, AH (10), BH (11), DH (13), EH (14,) and FH (15) are allowed as zoom-point values. Settings other than these are prohibited.
- When HSS = 1, the end codes of the original graphic are ignored whether the sprite is enlarged or reduced.
- Set the lower 2 bits of CMDLINK to "00".
- **Set the lower 2 bits of lookup table address to "00".**
- When setting a color bank, set the lower 4 bits of the color bank data to "0000."
- Set ECD and SPD to "1" and the color mode to "0" for non-textures.
- When calculating colors in non-textures, set the non-texture color using RGB data (greater than 8000H).
- Set ECD to "0" when the color mode is set to the RGB mode and 7FFFH data are used for the original graphic.
- Set SPD to "0" when the color mode is set to the RGB mode and 0000H data are used for the original graphic.
- When the color mode is set to the RGB mode, the frame buffer cannot be set to 8 bits / pixel. The only color modes in which 8 bits / pixel can be set are modes 0, 1, 2, 3, and 4.
- When the frame buffer is set to 8 bits / pixel, set the color calculation mode to "0." Only replace is possible. Color calculation other than this is prohibited.
- When the frame buffer is set to 8 bits / pixel, the upper 8 bits of non-textured colors are ignored. The use of RGB data is prohibited at this time.
- **When the frame buffer is set to 8 bits / pixel, set the MSB ON bit (MON) to "0."**
- When the MSB ON bit (MON) is set to "1," set the color calculation mode to "0."
- Store command tables to address 000000H of VRAM.

(Page 160 is blank in the original Japanese version.)

SEGA Confidential



# INDEX

#	disable (transparent pixel) .....	88
(color) lookup table mode .....	89	
(color) palette code .....	89	
1-cycle mode .....	38	
31KC .....	37	
<b>A</b>		
access (system registers) .....	23	
address map .....	18	
<b>B</b>		
background .....	94	
boundaries .....	24	
burst transfer (DMA).....	23	
byte access (VRAM) .....	19	
<b>C</b>		
change (manual mode) .....	39	
character address .....	103	
character pattern (each color mode) .....	90	
character pattern .....	60	
character pattern table .....	26, 60	
character read direction .....	77	
character size .....	104	
clipping coordinate set command .....	109	
CMDCOLR .....	98	
CMDCTRL .....	70	
CMDGRDA .....	106	
CMDLINK .....	78	
CMDPMOD .....	79	
CMDSIZE .....	104	
CMDSRCA .....	103	
CMDXA~CMDYD .....	105	
color .....	12	
color bank .....	99	
color bank code .....	100	
color calculation .....	93	
color control word .....	98	
color lookup table .....	26, 62, 101	
color mode .....	89	
command table .....	25, 66, 69	
command table flow .....	30	
commands .....	71, 109	
control words .....	70	
coordinate set command .....	109	
CPU .....	2	
current operation table address (register) .....	55	
<b>D</b>		
disable (end code) .....	86	
disable (pre-clipping) .....	83	
display device .....	2	
display range .....	14	
distorted sprite draw command .....	124	
distorted sprites .....	8	
DMA burst transfer .....	23	
double interlace .....	43	
double interlace enable bit .....	43	
double interlace draw line .....	43	
draw forced termination register .....	51	
draw command .....	109	
draw mode word .....	79	
draw procedure flow .....	28	
draw end command .....	132	
dropout processing .....	9	
<b>E</b>		
end bit .....	70, 71	
end code disable .....	86	
erase & change (manual mode) .....	40	
erase (manual mode) .....	39	
erase/write .....	46	
even lines .....	43	
even/odd coordinate selection .....	44	
<b>F</b>		
fetch .....	29	
fetch status (of end bit) .....	52	
fill data (erase / write) .....	46	
fill-in (polygons) .....	10	
fixed point (zoom point) .....	73	
flow (of command tables) .....	30	
flow (of draw procedure) .....	28	
frame .....	38	
frame buffer .....	2, 20	
frame buffer change mode bit .....	38	
frame buffer change mode register .....	38	
frame buffer change trigger bit .....	38	
<b>G</b>		
Gouraud shading .....	64, 95	
Gouraud shading table .....	26, 64, 106	
<b>H</b>		
half-luminance .....	95	
half-transparent .....	95	
HDTV .....	37	
high resolution .....	37	
high speed shrink .....	81	
horizontal inversion .....	77	

## I

- inside drawing mode ..... 84, 115
- interlace mode ..... 43
- interrupt signal ..... 52
- inversion (horizontal, vertical) ..... 77

## J

- jump assign ..... 72
- jump call ..... 72
- jump mode ..... 72
- jump next ..... 72
- jump return ..... 72

## L

- line draw command ..... 130
- lines ..... 11
- LINK specification ..... 78
- local coordinate set command ..... 116
- local coordinates ..... 117
- lookup table mode ..... 89
- luminance (Gouraud shading) ..... 64
- luminance (RGB) ..... 91

## M

- manual mode ..... 39
- mesh enable ..... 85
- mode 0 (color mode) ..... 90
- mode 1 (color mode) ..... 90
- mode 2 (color mode) ..... 91
- mode 3 (color mode) ..... 91
- mode 4 (color mode) ..... 91
- mode 5 (color mode) ..... 91
- mode status register ..... 57
- MSB ON ..... 97

## N

- non-textured color ..... 13, 102
- non-textured parts ..... 10
- non-textured draw commands ..... 109
- normal sprite ..... 5
- normal sprite draw command ..... 118
- NTSC system ..... 37

## O

- odd lines ..... 43
- order of priority (frame buffer) ..... 20
- order of priority (VRAM) ..... 19
- original picture ..... 94
- outside drawing mode ..... 84, 115

## P

- PAL system ..... 37
- palette code ..... 89
- parts ..... 4

- plot trigger mode ..... 45
- plot trigger mode register ..... 45
- point coordinate data ..... 105
- polling ..... 52
- polygon draw command ..... 126
- polygons ..... 10
- polyline draw command ..... 128
- polylines ..... 11
- pre-clipping ..... 83
- last operation table address (register) ..... 54
- pseudo draw continuation ..... 56

## R

- read direction (character) ..... 77
- read/write access (VRAM) ..... 19
- referencing (of tables) ..... 31
- registers (system) ..... 34
- replace ..... 94
- reset ..... 23
- RGB code ..... 91
- RGB mode ..... 89

## S

- scaled sprite ..... 6
- scaled sprite draw command ..... 120
- screen modes ..... 14, 37
- shadow (color calculation) ..... 94
- shadow (MSB ON) ..... 97
- single interlace ..... 43
- skip assign ..... 72
- skip call ..... 72
- skip next ..... 72
- skip return ..... 72
- specification of coordinates  
of two points (scaled sprite draw command) ..... 120
- specification of coordinates  
of two points (scaled sprites) ..... 6
- specification of zoom point  
(scaled sprite draw command) ..... 122
- specification of zoom point (scaled sprites) ..... 7
- sprite IC ..... 2
- subroutines (jump call, skip call) ..... 30, 72
- system clipping ..... 111
- system clipping coordinate set command ..... 110
- system controller ..... 2
- system registers ..... 2, 23, 34



**T**

table referencing flow .....	31
tables (in VRAM) .....	24, 59
texture draw commands .....	109
textured parts .....	5
transfer end status register .....	52
transfer-over .....	53
transparent color code.....	88
transparent pixel disable .....	88
trigger (draw) .....	45
TV mode selection bit.....	36
TV mode selection register .....	36

**U**

user clipping .....	113
user clipping coordinate set command .....	112
user clipping enable .....	84
user clipping mode .....	84

**V**

V-blank erase / write enable bit .....	36
VDP1 .....	1
VDP1 Functions .....	3
VDP2 .....	2
version number .....	57
vertex coordinate data.....	105
vertical inversion .....	77
VRAM .....	2, 19

**W**

window (MSB ON) .....	97
word access (VRAM).....	19

**Z**

zoom point(fixed point) .....	73
-------------------------------	----

The following table shows the specifications of the horizontal resolution settings for VDP1 and VDP2.

VDP2 setting (HRESO value)	VDP1 setting (TVM value)	Setting status
Normal mode (000 or 001)	Normal (000)	Enabled
	High-resolution (001)	Disabled
	Rotation 16 (010)	Enabled
	Rotation 8 (011)	Enabled
	HDTV (100)	Disabled
High-resolution mode (010 or 011)	Normal (000)	Enabled *
	High-resolution (001)	Enabled
	Rotation 16 (010)	Enabled *
	Rotation 8 (011)	Enabled *
	HDTV (100)	Disabled
Dedicated monitor mode (100, 101, 110, or 111)	Normal (000)	Disabled
	High-resolution (001)	Disabled
	Rotation 16 (010)	Disabled
	Rotation 8 (011)	Disabled
	HDTV (100)	Enabled

The following graphics will be displayed by the interlace setting of VDP2 (LSMD bit) and VDP1 (DIE bit):

LSMD value	DIE value	VDP2 display	VDP1 display
00	0	non-interlace	non-interlace
	1	non-interlace	cannot display correctly
10	0	single-density interlace	single-density interlace
	1	single-density interlace	double-density interlace
11	0	double-density interlace	single-density interlace
	1	double-density interlace	double-density interlace

### The Normal Shadow

Normal shadows are created by drawing a reserved paletted pixel value to the VDP1 frame buffer. What the value actually is depends on the sprite type, but, in general, the value is derived by setting all of the dot color bits except the least-significant bit to 1. Depending on the sprite type, this value may be 0x3e, 0x7e, 0xfe, 0x1fe, 0x3fe, or 0x7fe (see page 201 of the VDP2 manual). The values of the remaining bits in the pixel are not significant. When the VDP2 encounters such a pixel, the effect is the same as when it encounters a transparent MSB shadow pixel: the background pixel immediately underneath the sprite pixel is displayed at a reduced luminance.

Be advised that the paletted pixel value reserved for normal shadow pixels is reserved whether shadow processing is enabled or not. If shadow processing is not enabled, then normal shadow pixels are simply not displayed. This means that, depending on the sprite type being used, one or more entries in color RAM cannot be used by any sprite.

For details on this topic, see Section 14.1, "Shadow Processing," in the VDP2 Hardware Manual.

### Normal Shadow Data According to Sprite Type

Sprite type	Number of sprite colors	Palette code	Color bank
Types 0–3, 5	16	1110	xxxxx11111110000
	64	xx111110	xxxxx11111xx0000
	128	x1111110	xxxxx1111xxx0000
	256	11111110	xxxxx111xxxx0000
Types 4, 6	16	1110	xxxxxx11111110000
	64	xx111110	xxxxxx1111xx0000
	128	x1111110	xxxxxx111xxx0000
	256	11111110	xxxxxx11xxxx0000
Type 7	16	1110	xxxxxxxx111110000
	64	xx111110	xxxxxxxx111xx0000
	128	x1111110	xxxxxxxx11xxx0000
	256	11111110	xxxxxxxx1xxxx0000
Types C–F	16	1110	xxxxxxxxx11110000
	64	xx111110	xxxxxxxxx11xx0000
	128	x1111110	xxxxxxxxx1xxx0000
	256	11111110	Not applicable
Type 8	16	1110	xxxxxxxxxx1110000
	64	xx111110	xxxxxxxxxx1xx0000
	128	x1111110	Not applicable
	256	11111110	Not applicable
Type 9–B	16	1110	xxxxxxxxxxx110000
	64	xx111110	Not applicable
	128	x1111110	Not applicable
	256	11111110	Not applicable