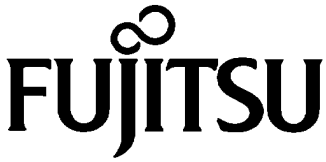
MB86292 <ORCHID>

Application Note

Revision 1.0
Jan, 2002



- The contents in this document are subject to change without notice.
- The contents of this document must not be reprinted or duplicated.
- This document does not give consent to use Fujitsu's industrial proprietary rights.

  Fujitsu takes no responsibility for infringement of proprietary, patents or other rights owned by third parties caused by use of the information and circuit diagrams.

Introduction

The MB86292 (code name: Orchid; hereinafter referred to as Orchid) application notes are compilation of information related to circuit designs and graphics programming for users to refer to when using the graphics display controller Orchid or Fujitsu's MB86290 series graphics driver to design software and hardware.

Always verify operations before using the circuit and programming examples described in this document.

CONTENTS

1. CONNECTION EXAMPLE FOR EACH INTERFACE

1.1 Connecting to Host CPU

Orchid can be connected with Fujitsu's SPARClite, Hitachi's SH4 and SH3 and NEC's V83X with MODE [0:1], without an external circuit.

The MODE2 pin enables you to select "Normally Not Ready" or "Normally Ready" for the action of the ready signal.

You must set Orchid to the software space being mapped that is proper for the ready signal mode (The software should be set to 1 in the "Normally Not Ready" mode; the software should be 2 in the "Normally Ready" mode).

Refer to the ***MB86292 <Orchid> Specifications Manual*** for details of the ready signal.

1.1.1 Example connection with SH3 and SH4

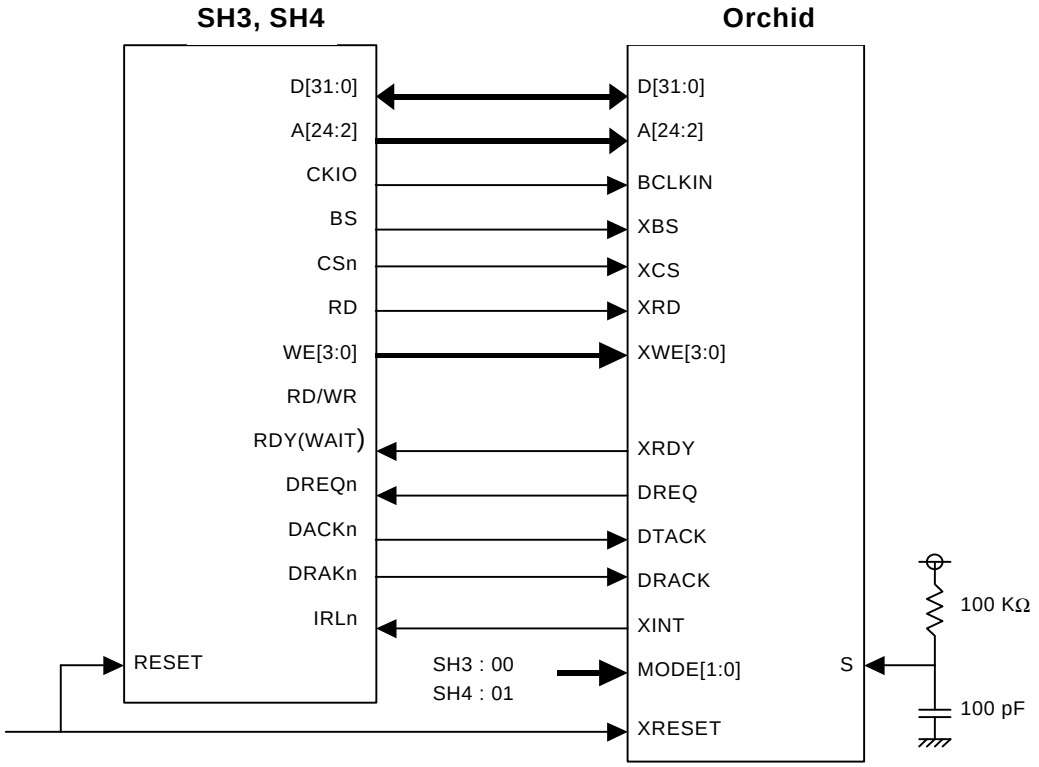


Fig.1 Example Connection with SH3 and SH4

- The RC connection in the above figure is an example for pulse inputs to the S pin higher than 500 nsec.

  A Low level higher than 300 μs is required in the XRESET pin after the S pin is set at a High level, so it is recommended that the CPU RESET pin input be input directly to the XRESET, as shown in the above diagram (For SH, a Low level of 10 msec is retained at power on).

- Set the DMAC register to output DREQ as Low active and DRACK and DACK as High active.
- XRDY output is “Hi-Z,” when CS is not active, so it is recommended that a Pull-Down (or Pull-up for SH3) be setup for the RDY signal line on the outside of the chip.
- To enable a WAIT by the XRDY, insert it into the area the Orchid was mapped by the WCR2 register for the MODE2 pin and set a software WAIT 1 or 2.

1.1.2 Example connection with V83X

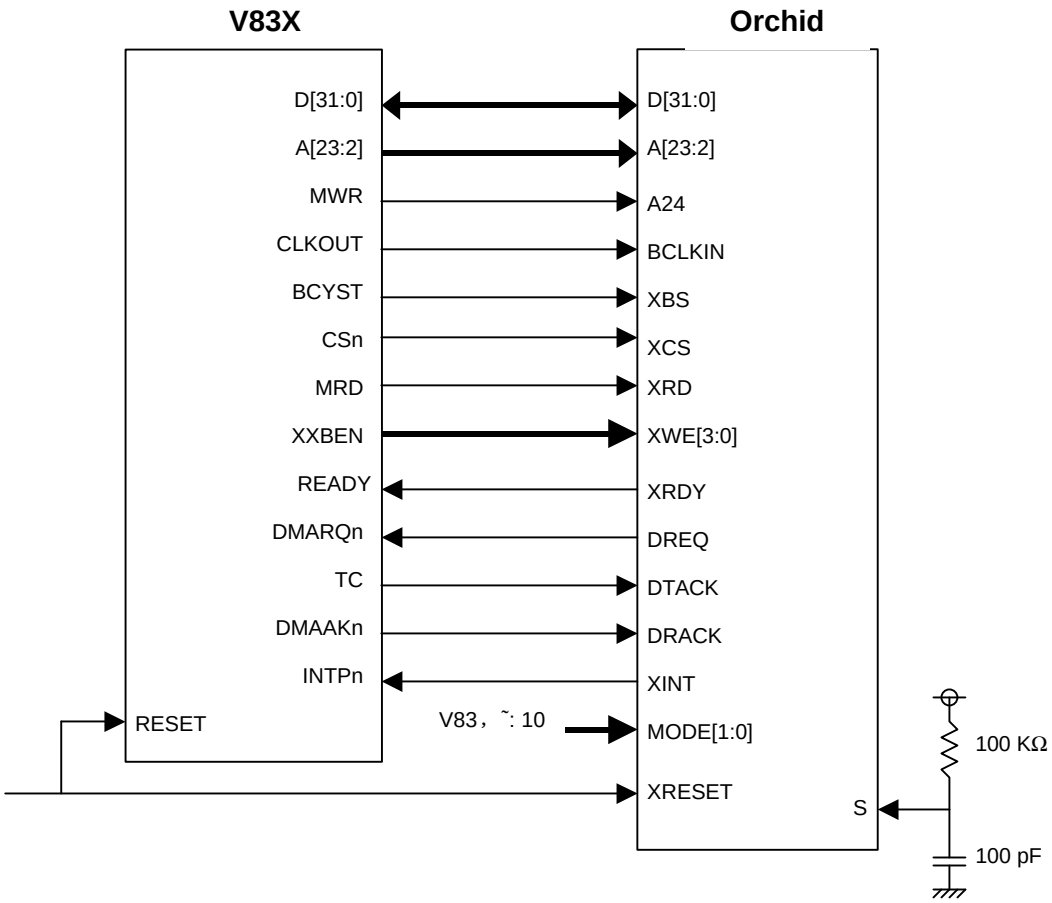


Fig.2 Example Connection with the Video Interface Unit

- The RC connection in the above figure is an example for pulse inputs to the S pin higher than 500 nsec.

  A Low level of higher than 300 μs is required in the XRESET pin after the S pin is set at a High level. To activate the V83X with the PLL mode, it is recommended that the CPU RESET pin input be input directly to the XRESET, as shown in the above figure (In the PLL mode, a Low level of 10 msec is retained at power on to allow the PLL to stabilize).

- The DMARQn should be low active; DMAAKn should be high active. Select TC for the TC/STOPAK dual use pin, then set the V83X register to end transfers at a Low signal.

- XRDY output is “Hi-Z,” when CS is not active, so it is recommended that a Pull-Down be setup for the XRDY signal line on the outside of the chip.

- To enable a WAIT by the XRDY, insert it into the area the Orchid was mapped by the PWC register for the MODE2 pin and set a software WAIT 1 or 2.

1.1.3 Example connection with SPARClite

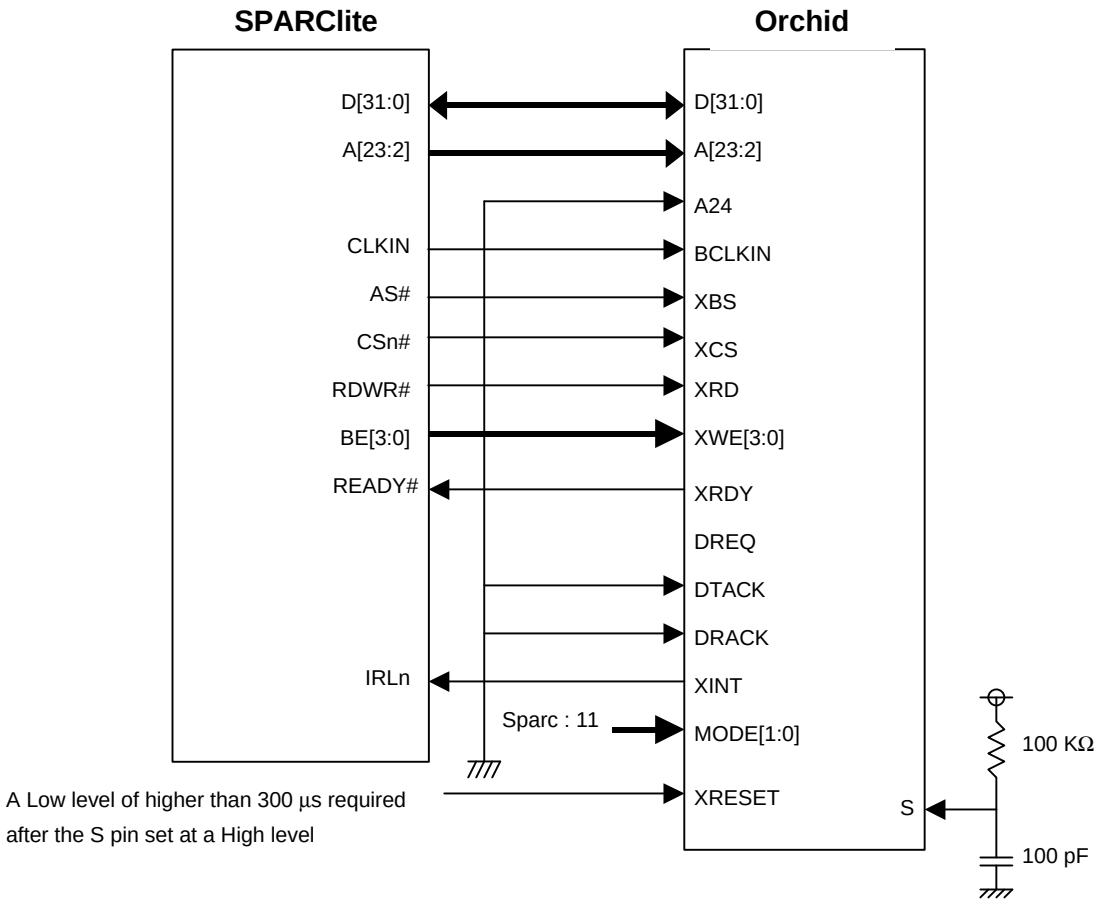
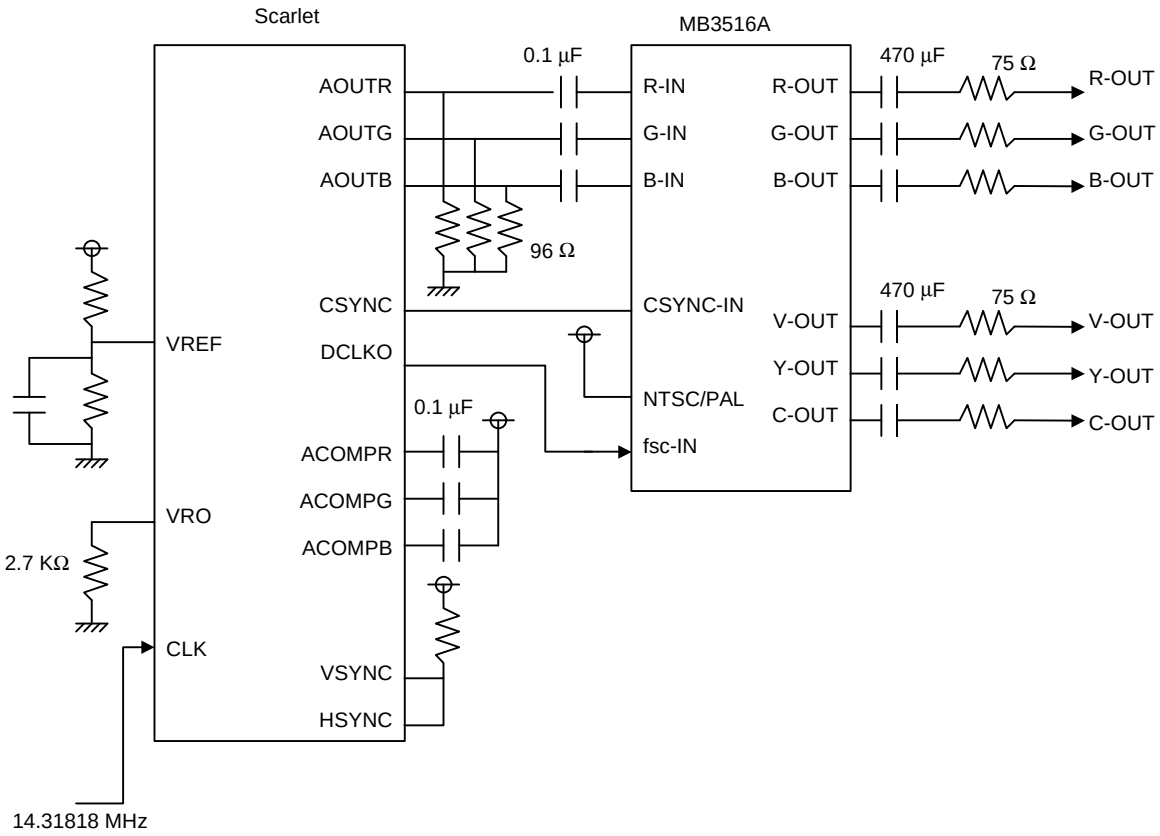


Fig.2 Example Connection of V83X and Orchid

- The RC connection in the above figure is an example for pulse inputs to the S pin higher than 500 nsec.
- Because No DMAC is provided on SPARClite, the DMA pins are open and are set to a low clamp.
- XRDY output is “Hi-Z,” when CS is not active, so it is recommended that a Pull-Down be setup for the XRDY signal line on the outside of the chip.

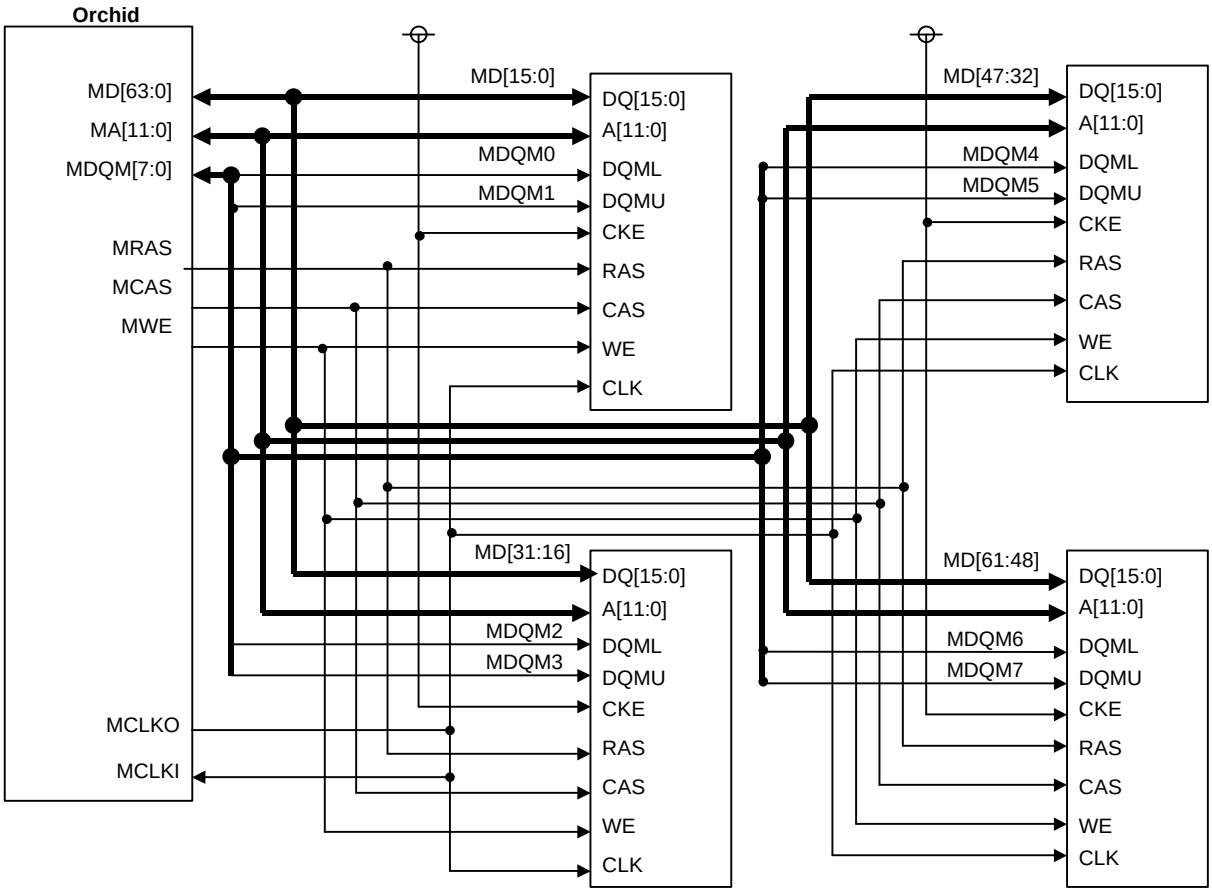
1.2 Connection Example for the Video Interface Section

1.2.1 Example connection with MB3516A

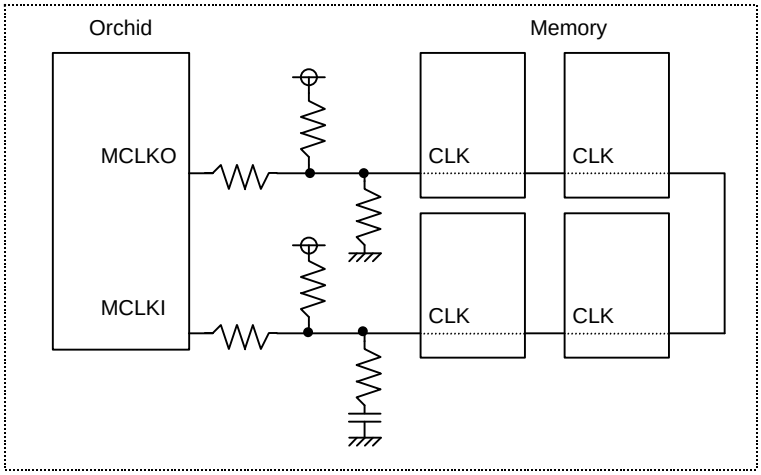


Note 1) Vertical resolution is relatively higher for the interlacing and video mode but the screen may flicker. To prevent the screen flickering, use the non-interlacing mode.

1.3 Example Connection for the Memory Interface Section



- The above figure is an example of four connections of 16 Mbit × 16 FCRAM. Orchid can be set to Row and Column and Bank with the MMR (memory mode register) so it can be used by simply connecting to an address.



Example External Connection of the Memory Clock Signal

- Orchid has been designed to allow you to use it directly with the memory. However, there may be the need to make fine timing adjustments that depend on the board layout.

  In such cases, it is recommended that the memory clocks be laid out as shown in the above figure.

  Note that the resistor in series is inserted for the noise prevention, and the resistor connected to the power and ground, and the capacitor are inserted for timing adjustments.

2. STARTING UP ORCHID

2.1 Setting Memory Interface

It is necessary to always set the MMR register (memory interface register) before setting the display and drawing when starting up Orchid.

Refer to ***Section 1.5.1*** in the ***MB86292<Orchid> Specifications Manual*** for an example of the MMR settings.

2.2 Transferring Display List

On Orchid, a display list is transferred to the built-in FIFO to process drawing.

There are three ways to transfer display lists.

- DMA transfer (Main memory => Orchid's built-in FIFO) Only on SH4 and V83x.
- Local transfer (Graphic memory => Orchid's built-in FIFO)
- CPU transfer (CPU => Orchid's built-in FIFO)

2.2.1 Executing a display list using the graphics driver

There are two modes to execute the display list with the graphics driver.

- Synchronized Mode: Mode to transfer display lists each time driver function related to the drawing processing is executed. This mode is mainly used when debugging software.
- Asynchronized Mode: Mode to transfer display lists that has been stacked on the memory until that time at any timing. There are two types of functions to start the transfer of display lists.

  Function Name: GdcFlush(): Function that exits immediately after starting to transfer display lists.

  Function Name GdcSync (): Function that waits for the drawing processing to end after starting to transfer display lists

The following shows an example program.

```
/* In synch mode */
GdcExecMode ( GDC_ENABLE );                          /* Set to synch mode */
            :
GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );
                                                     /* Transfers display list for register setting 1 */
GdcSetLinePattern ( 0x0000FFFF );                    /* Transfers display list for register setting 2 */
GdcPrimType ( GDC_LINES_FAST );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );          /* Transfers display list for dram command 3 */
GdcPrimEnd ( );

/* In asynch mode */
GdcExecMode ( GDC_DISABLE );                         /*  Set to asynch mode */
            :
GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );
GdcSetLinePattern ( 0x0000FFFF );
GdcPrimType ( GDC_LINES_FAST );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcPrimEnd ( );
GdcFlush ( );                                        /* Batchs display list transfer */

/* In asynch mode (waiting end of drawing processing) */
GdcExecMode ( GDC_DISABLE );                         /* Set to asynch mode */
GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );
GdcSetLinePattern ( 0x0000FFFF );
GdcPrimType ( GDC_LINES_FAST );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcPrimEnd ( );
GdcSync ( );                                         /* Batchsdisplay list transfer and waits for drawing
                                                        engine processing to end */
```

The procedures for the DMA transfer using the driver function, the local display transfer and the CPU transfer is set by using the GdcFlushDisplayList () function. Refer to the ***MB86290 Series Graphic Driver User's Manual*** for details.

2.2.2 Executing display list

The following is an example program for transferring using CPU write, DMA, and local transfer.

Note that the following are the functions in the example program.

```
void wrb(unsigned long addr, unsigned char data) {          /* byte write */
    unsigned char *paddr = ((unsigned char *) addr);
    *paddr = data;
}
void wrw(unsigned long addr, unsigned short data) {         /* word write */
    unsigned short *paddr = ((unsigned short *) addr);
    *paddr = data;
}
void wrl(unsigned long addr, unsigned long data) {          /* long word write */
    unsigned long *paddr = ((unsigned long *) addr);
    *paddr = data;
}
```

2.2.3 CPU write transfer (Main Memory => FIFO)

It is possible to write the program from each CPU to Orchid to transfer the display lists.

To execute drawing by writing the program from the CPU, program it so as to write to the FIFO if there is more than half of the FIFO capacity remaining.

2.2.4 DMA transfer (Main Memory => FIFO)

Orchid supports each CPU single addresses and dual address DMA transfers. The DSU register for the host interface register group is used to set the procedure for transfer.

When using SH3, transferring display lists to the Orchid FIFO using DMA is prohibited. There is no problem in transferring to the graphics memory.

Furthermore, when using SH4 or V83x, insert an interrupt command after the display list and perform DMA transfer. Do not access the Orchid register or the graphics memory until the interrupt has been generated from Orchid.

2.2.4.1 When using SH4

<Setting Example Conditions>

- Uses the P2 region as the logic space
- Maps Orchid in CS4 as the physical space
- Already have written display list to physical space C3.
- Uses DMA channel 1

```
#define SAR1            0xFFA00010      /* DMA source address register 1 */
#define DAR1            0xFFA00014      /* DMA destination address register 1 */
#define TCR1            0xFFA00018      /* DMA transfer count register 1 */
#define CHCR1           0xFFA0001C      /* DMA channel control register 1 */
#define DMAOR           0xFFA00040      /* DMA operation register */

#define GDC_DSU         0xB1FC0004      /* Scarlet DMA setup register */
#define GDC_DTC         0xB1FC0000      /* Scarlet DMA transfer count register */
#define GDC_DRQ         0xB1FC0018      */ Scarlet DMA request register */

/*-----------------------------------------------------------------------------
Setting/Starting up SH4 DMAC:
-----------------------------------------------------------------------------*/

    wrb ( GDC_DSU,0x02 );               /* Sets DMA transfer mode */
    wrl ( CHCR1, 0x00001241 );          /* Fixes DRAK, DACK hi-active, destination address, and increase
                                           source addresses, external request, single address mode, 32 byte
                                           transfer */
    wrl ( DMAOR, 0x00000001 );          /* DMA startup allowed */
    wrl ( SAR1, 0xAC000000 );           /* DMA source address 0xAC000000 */
    wrl ( DAR1, 0xB1FF04A0 );           /* DMA destination address DrawBase + 4A0 (DFIFO) */
    wrw ( TCR1, 0x1000 );               /* DMA transfer count 0x1000 */

    wrw ( GDC_DTC, 0x1000 );            /* DMA transfer count 0x1000 */
    wrb ( GDC_DRQ, 0x01 );              /* DMA startup */
```

2.2.4.2 When using V83X

<Setting Example Conditions>

- Maps Orchid to block 3 (CS3) in the memory space
- Display list
- Uses DMA channel 0

```
#define V_DSA0H      0xC0000030        /* DMA source address register 0 High */
#define V_DSA0L      0xC0000032        /* DMA source address register 0 Low */
#define V_DDA0H      0xC0000034        /* DMA destination address register 0 High */
#define V_DDA0L      0xC0000036        /* DMA destination address register 0 Low */
#define V_DBC0H      0xC0000038        /* DMA byte count register 0 High */
#define V_DBC0L      0xC000003A        /* DMA byte count register 0 Low */
#define V_DCH0       0xC000003C        /* DMA byte count register 0 Low */

#define GDC_DRQ      0x53fc0018        /* Scarlet DMA request register */
#define GDC_DSU      0x53fc0004        /* Scarlet DMA setup register */

/*-----------------------------------------------------------------------------
Setting/Starting up of V83x DMAC:
-----------------------------------------------------------------------------*/

    wrw_IO(V_DSA0H, (short)(glbDmaSAR1 >> 16));        /* Sets transfer source address */
    wrw_IO(V_DSA0L, (short)(glbDmaSAR1));

    wrw_IO(V_DDA0H, (short)(glbDmaDAR1 >> 16));        /* Sets transfer destination address */
    wrw_IO(V_DDA0L, (short)(glbDmaDAR1));

    wrw_IO(V_DBC0H, (short)((glbDmaTCR1-1) >> 14));    /* Sets transfer count */
    wrw_IO(V_DBC0L, (short)((glbDmaTCR1-1) << 2));

    wrw_IO (V_DCHC0, 0x00a5);                          /* Set channel transfer mode */
    wrw_IO(V_DC, 0x0001);                              /* Control DMA transfer mode */

    wrb (GDC_HOST_DSU, 0x02 );                         /* Sets DMA transfer mode */
    wrb (GDC_HOST_DRQ, 0x01 );                         /* Starts up DMA */
```

Note) When selecting the CPU mode for V83x (MODE[1:0]=0x2), instead of counting DMA transfers on Orchid, it checks for the end of the transfers with the XTC signal from V83x. For that reason, it is not necessary to set the DTC register in the Orchid host interface register group.

2.2.5 Local transfer (Graphic Memory => FIFO)

This allocates any space in the graphics memory as the display list region and after storing the display list, Orchid reads the display list sequentially.

To use the local transfer, write 0x00000001 to the HostBase + 0x001ch.

<Transfer example conditions>

- Display list storage graphics memory has the base address: 0x00000000
- Display list is stored in the graphics memory.
- Memory space allocation has the same conditions as DMA transfer.

```
#define GDC_LSA         0xB1FC0040          /* Orchid display list source address register */
#define GDC_LCO         0xB1FC0044          /* Orchid display list transfer count register */
#define GDC_LREQ        0xB1FC0048          /* Orchid display list transfer request register */

/*-----------------------------------------------------------------------------------------------
Reads the display list from the graphics memory
-----------------------------------------------------------------------------------------------*/

        wrl ( GDC_LSA, 0x00000000 );        /* Sets display list strange address on graphic memory */
        wrl ( GDC_LCO, 0x00001000 );        /* Sets display list transfer count */
        wrb ( GDC_LREQ, 0x01 );             /* Starts up display list transfer */
```

3. DISPLAY FUNCTION

Orchid has a four hierarchical structured display screen, the lower two of which can be split into an M and B layer on the left and right. This enables the use of six logical display screens. Note that the M and B layers cannot be displayed on the right side (MR, BR) only. To display on one screen, use the ML and BL layers.

Two hardware cursors are available as well as the six display screens.

3.1 Setting Display Parameters

3.1.1 Programming the horizontal and vertical directions

It is necessary to set each type of register of the display control register group for the display resolution when using Orchid. The following shows an example register setting for each resolution.

Resolution	SC	HTP	HSP	HDP	HSW	VTR	VSP	VDP	VSW
320 × 200	29	423	350	319	30	262	224	199	2
320 × 240	29	423	350	319	30	262	244	239	2
360 × 200	26	470	389	359	34	262	224	199	2
400 × 200	23	529	435	399	38	262	224	199	2
480 × 200	19	635	520	479	46	262	224	199	2
640 × 400i	14	847	700	639	62	262	224	199	2
640 × 480	7	799	655	639	95	524	489	479	1
640 × 480i	14	847	700	639	62	262	242	239	2
854 × 480	5	1061	874	853	125	524	489	479	1
800 × 600	4	1055	839	799	127	632	600	599	3
1024 × 768	2	1388	1047	1023	135	805	770	767	5

‘i’ represents Interlace.

The sizes of the ML and MB and BL and BR layers when split are set using the HDB register (the register that sets the display period of the ML and BL registers).

Refer to the ***MB86291 <Orchid> Product Specifications*** for details regarding definitions of HTP.

3.1.2 Programming logic image space

To use multiple layers, set drawing frame size for each layer and memory addresses.

The following shows an example setting.

- C layer: drawing frame size (640 × 480)

C layer logic frame height (CH)	480 raster
C layer logic frame memory width (CW)	10 (640 pixel)
C layer color mode (CC)	8-bit indirect color
C layer logic frame origin address (COA)	0x000000
C layer display origin address (CDA)	0x000000
C layer X direction display starting position (CDX)	0x0000
C layer Y direction display starting position (CDY)	0x0000
C layer transparent color setting (CTC)	Palette number 0 is transparent color.

```
/*---------------- Driver setting example ----------------------------------------------*/
        /*Sets display frame */
GdcDispDimension ( GDC_DISP_LAYER_C, GDC_ENABLE, GDC_DISP_8_BPP, GDC_FLIPMODE_0,
                                        0, 0, 640, 480 );          /*C layer frame attribute */
        /*Display starting position */
GdcDispPos ( GDC_DISP_LAYER_C, 0, 0, 0 );                          /* Sets C layer display starting position */
GdcColorTransparent ( GDC_DISP_LAYER_C, 0x0 );                     /* Sets C layer transparent color */
GdcColorZeroMode(GDC_DISP_LAYER_C, GDC_COLOR_TRANSPARENT );   /* Sets color zero mode */
GdcColorPalette ( GDC_C_LAYER_PALETTE, 0, 256, ucolorC );          /* Sets C layer palette */

/*---------------- Register Setting Example ---------------------------------------------*/
CM Reg { CC, CW, CDY Reg  =  0x0000, CTC Reg  =  0x8000
/*---------------- Sets palette register ------------------------------------------------*/
For ( I=0 ; I < 256 ; I + + ) {
          CPAL ( I ) Reg  =  * ( ucolorC);
          ucolorC  + +  ;
 }
*Set logic frame height to size - 1.
```

- BL layer: drawing frame size (1280 × 960)

BL layer logic frame height (BLH)	960 raster
BL layer logic frame memory width (BLW)	20 (1280 pixels)
BL layer display screen (BLFLP)	Screen 0
BL layer color mode (BLC)	8-bit indirect color
BL layer logic frame origin address (BLOA0)	0x4b000
BL layer display origin address (BLDA0)	0x4b000
BL layer X direction display starting position (BLDX)	0x0000
BL layer Y direction display starting position (BLDY)	0x0000

```
/*---------------------- Driver setting example ----------------*/
        /* Set display frame */
GdcDispDimension ( GDC_DISP_LAYER_BL, GDC_ENABLE, GDC_DISP_8_BPP, GDC_FLIPMODE_0, 0x4b000,
                        0x4b000, 1280, 960 );          /* Sets BL layer frame attribute */
        /* Display starting position */
GdcDispPos ( GDC_DISP_LAYER_BL, 0, 0, 0 );             /* Sets BL layer display starting position */
/* ---------------- Register setting example -------------------*/
BLM Reg { BLC, BLFLP, BLW, BLH } = 0x001403BF, BLOA0 Reg = 0x4b000,
BLDA0 Reg = 0x4b000,  BLDX Reg = 0x0000, BLDY Reg = 0x0000
* Set logic frame height to size - 1.
```

- BR layer: drawing frame size (320 × 480)

BR layer logic frame height (BRH)	480 raster
BR layer logic frame memory width (BRW)	10 (320 pixels)
BR layer display screen (BRFLP)	Screen 0
BR layer color mode (BRC)	16-bit indirect color
BR layer logic frame origin address (BROA0)	0xE1000
BR layer display origin address (BRDA0)	0xE1000
BR layer X direction display starting position (BRDX)	0x0000
BR layer Y direction display starting position (BRDY)	0x0000

```
/*---------------------- Driver setting example -----------------------------------------*/
        /* Sets display frame */
GdcDispDimension ( GDC_DISP_LAYER_BR, GDC_ENABLE, GDC_DISP_16_BPP, GDC_FLIPMODE_0, 0xE1000,
                        0xE1000, 320, 480 );           /* Sets BR layer frame attribute */
        /* Display starting position */
GdcDispPos ( GDC_DISP_LAYER_BR, 0, 0, 0 );             /* Sets BL layer display starting position */

/* ---------------------- Register setting example ---------------------------------------*/
BRM Reg { BRC, BRFLP, BRW, BRH } = 0x800A03BF, BROA0 Reg = 0xE1000,
BRDA0 Reg = 0xE1000,  BRDX Reg = 0x0000, BRDY Reg = 0x0000
* Set logic frame height to size - 1.
```

3.1.3 Setting overlay mode

The results of overlaying the C layer and other layers can be α-blended. The following shows a setting example of the overlay blend mode.

<Setting example>

Overlay mode: Blend mode

Blend coefficient: 0x80

```
        /* When using driver */
GdcOverlayPriorityMode ( GDC_OVERLAY_C_BLEND );     /* Sets overlay display mode */
GdcOverlayBlend ( GDC_ENABLE, 0x80 );               /* Sets blend coefficient */
        /* Register setting example */
BMODE Reg  =  0x0001
BRATIO Reg  =  0x0080
```

* Blending is performed only when the C layer pixel is 1.

3.1.4 Setting display control mode register

<Setting example>

Setting synch mode	: Non-interlace
Synch signal generation polarity	: Negative logic output
EO I/O format	: Low level output with even numbered frames
Scaling	: 1/8 (25 MHz)
Reference clock	: Built-in PLL
Display layers	: C, ML, MR, BL, BR

```
/* Sets display control mode register  */
        /* When using driver */
GdcDispClock ( 0x00000700 ) ;
        /* Register setting example */
DCM Reg { CKS, SC, EO, SF, ESY, SYNC } = 0x0700
DCE Reg { DEN, BE, ME, WE, CE } = 0x800D
```

* Display control mode must be set in the last step of the process for setting the display parameters.

* The driver function **`GdcDispClock ()`** is the driver function that sets the DCM register. The DCE register is set to each layer by the driver function **`GdcDispDimension()`** layer enable.

3.1.5 Screen scroll

Screen scroll is performed by changing the X direction starting position, the Y direction starting position and the display origin address.

The following shows a screen scroll programming example.

<Program operation example >

- BL layer has the same conditions as the BL settings described in ***Section 3.1.2***.
- BL layer scrolls 1 pixel from the right to the left for each time a vertical synch interrupt occurs.

```
/* ---------- Screen scroll program example ----------*/
#define     BLda        0x4b000     /* BL layer memory address */
#define     Ldx         0           /* X direction moving amount */
#define     Ldy         0           /* Y direction moving amount */

main ( ) {
. . . . .
. . . . .
    /* When using driver */
void VSyncINT ( void )                          /* VSYNC Interrupt */
{
    . . . . .
    GdcDispPos ( GDC_DISP_LAYER_BL, GDC_BANK_0, Ldx, Ldy );
    Ldx  =  Ldx + 1 ;
    Ldy  =  Ldy + 0 ;
    if ( Ldx > 1279)  Ldx  =  0;  if ( Ldy > 479 ) Ldy  =  0 ;
    . . . . .
    . . . . .
    GdcClearInterruptStatus ( 0xFB );           /* Clears interrupt status */
}
    /* When not using driver */
void VSyncINT ( void )                          /* VSYNC Interrupt */
{
    . . . . .
    BLDA0 Reg  =  Lda ;  BLDX Reg  =  Ldx ;  BLDY Reg  =  Ldy ;
    Ldx  =  Ldx + 1 ;
    Ldy  =  Ldy + 0 ;
    if ( Ldx > 1279)  Ldx  =  0;  if ( Ldy > 479 ) Ldy  =  0 ;
    Lda  =  Ldx + 1280 × Ldy + BLda ;
    . . . . .
    . . . . .
    IST Reg  =  0xf                             /* Clears interrupt status */
}
```

* The following shows the calculation of the display origin address.

- For 16-bit color:

  $DA = 2 \times DX + 64 \times W \times DY + OA$

- For 8-bit color mode:

  $DA = DX + 64 \times W \times DY + OA$

  OA: logic frame origin address

  DA: Display origin address

  W: Logic frame memory width

  DX, DY: Display origin coordinates

3.1.6 Flipping display

There is a flipping method automatically performed by Orchid and a method performed according to the user settings.

The following shows examples of these methods.

3.1.6.1 Automatic flipping by orchid

Alternately displays screen 0 and screen 1 of any desired layer for each frame.

This method enables smooth animation.

<Program example conditions>

- Alternately displays screen 0 and screen 1 of the BL layer.
- BL layer frame size 640 × 480
- Mapping of the BL layer screen 0 drawing frame address to 0x00000.
- Mapping of the BL layer screen 1 drawing frame address to 0x96000.
- Change the screen to draw for each vertical synch interrupt

```
/*---------- Flipping program example ----------*/
        /* When using driver */
void DISP_INIT ( void ) {          /* Initial setting of display parameter */
            ⋮
GdcDispDimension  (  GDC_DISP_LAYER_BL,  GDC_ENABLE,  16BPP_FORMAT,  GDC_FLIPMODE_AUTO,
                    0x00000000,  0x96000, 640, 480);          /* BL layer automatically set to flip mode */
            ⋮
}
void VSyncINT ( void ) {                                      /* VSYNC Interrupt */
            ⋮
if ( flag )          GdcDrawDimension ( GDC_16BPP_FORMAT, 0x00000, 640, 480 );
                                                              /* Sets draw screen to screen 0 */
else                 GdcDrawDimension ( GDC_16BPP_FORMAT, 0x96000, 640, 480 );
                                                              /* Sets draw screen to screen 1 */
            ⋮
flag = to flag
}
        /* When not using driver */
void DISP_INIT ( void ) {                                     /* Initial setting of display parameter */
            ⋮
BLM Reg { BLC, BLFLP, BLW, BLH } = 0xC02803BF,  BLOA0 Reg = 0x000000,
 BLOA1 Reg = 0x96000
            ⋮
}
void VSyncINT ( void ) {                                      /* VSYNC Interrupt */
            ⋮
if ( flag )     FBR Reg = 0x00000,  XRES Reg = 0x00000280
else            FBR Reg = 0x96000,  XRES Reg = 0x00000280
            ⋮
flag = to flag
}
```

3.1.6.2 Flipping by user settings

Flips screen 0 and screen 1 of any desired layer at any timing.
Using this method performs flips without displaying graphics partially drawn.
Note that the timing for flipping the actual screen is performed in synchronization with Vsync.

<Program Example Conditions>

- BL layer frame size 640 × 480
- Mapping of the BL layer screen 0 drawing frame address to 0x00000.
- Mapping of the BL layer screen 1 drawing frame address to 0x96000.
- Draws screen 0 while displaying screen 1, then displays screen 0 after completing the drawing of screen 0.

```
/*---------- Flipping program example ----------*/
      /* When using driver */
GdcDispDimension ( GDC_DISP_LAYER_BL, GDC_ENABLE, 16BPP_FORMAT, GDC_FLIPMODE_1,
                          0x00000000,  0x96000, 640, 480);          /* Displays BL layer screen 1 */
GdcDrawDimension ( GDC_16BPP_FORMAT, 0x000000, 640, 480 ) ;
                                                                    /* Sets draw screen to screen 0 */
            ⋮
GdcSync ( ) ;                              /* Waits for drawing of display list and completion of that process */
GdcDispDoFlip ( GDC_DISP_LAYER_BL GDC_BANK_0 );          /* Sets display screen to screen 0 */
      /* When not using driver */
BLM Reg { BLC, BLFLP, BLW, BLH } = 0xA02803BF,  BLOA0 Reg = 0x000000,
BLOA1 Reg = 0x96000,  FBR Reg = 0x000000,  XRES Reg = 0x0280
            ⋮
while ( CTR Reg  PS = 01) { }          /* Waits for pixel engine processing */
BLM Reg { BLC, BLFLP, BLW, BLH } = 0x802803BF
```

3.2 Setting Hardware Cursor

Two hardware cursors of the size 64 × 64 (pixels) can be used. Color can only be used in 8 bit/pixel indirect color mode and the look up table uses the C layer color palette.

Priority order for the C layer can be set independently for the two cursor displays. The portion of the cursor protruding from the screen is not displayed.

The following shows a hardware cursor setting programming example.

<Conditions>

- Pattern data stored in the pointer curpat, curpat1 is loaded to 0x96000, 0x97000 on the graphics memory.

```
/*-------------------------------------------------------------------------------------
            Cursor setting programming example
-------------------------------------------------------------------------------------*/
/*--------------------- Driver use example ----------------------------------------*/
#define         CURSOE0                                 0x96000
#define         CURSOE1                                 0x97000
/*--------------- Program example ---------------------------*/
GdcCursorAddres ( 0, CURSOE0 );                         /* Sets address that stores the cursor 0 pattern */
GdcCursorAddres ( 1, CURSOE1 );                         /* Sets address that stores the cursor 1 pattern */
GdcCursorPos ( 0, 128, 256 );                           /* Cursor 0 display position */
GdcCursorPos ( 1, 384, 256 );                           /* Cursor 1 display position */
GdcCursorPriority ( 0, GDC_PRIORITY_CURSOR );           /* Sets priority display with the C layer */
GdcCursorPriority ( 1, GDC_PRIORITY_CURSOR );           /* sets priority display with the C layer */
GdcCursorColorTransparent ( 0x0 );                      /* Sets cursor transparent color */
GdcCursorColorZeroMode ( GDC_COLOR_TRANSPARENT );       /* Sets color zero mode */
GdcCursorPattern ( 0, curpat );                         /* Loads cursor 0 pattern */
GdcCursorPattern ( 1, curpat2 );                        /* Loads cursor 1 pattern */
GdcCursorDisplay ( 0, GDC_ENABLE );                     /* Sets cursor 0 display */
GdcCursorDisplay ( 1, GDC_ENABLE );                     /* Sets cursor 1 display */
```

3.3 Video Capture Function

Orchid can compress the video streams in compliance with ITU-RBT656 into any ratio, and read them to the graphics memory to display them in the W layer.

3.3.1 Conversion into non-interlace

There are two modes for converting video streams into non-interlace. They are the BOB and WEAVE modes.

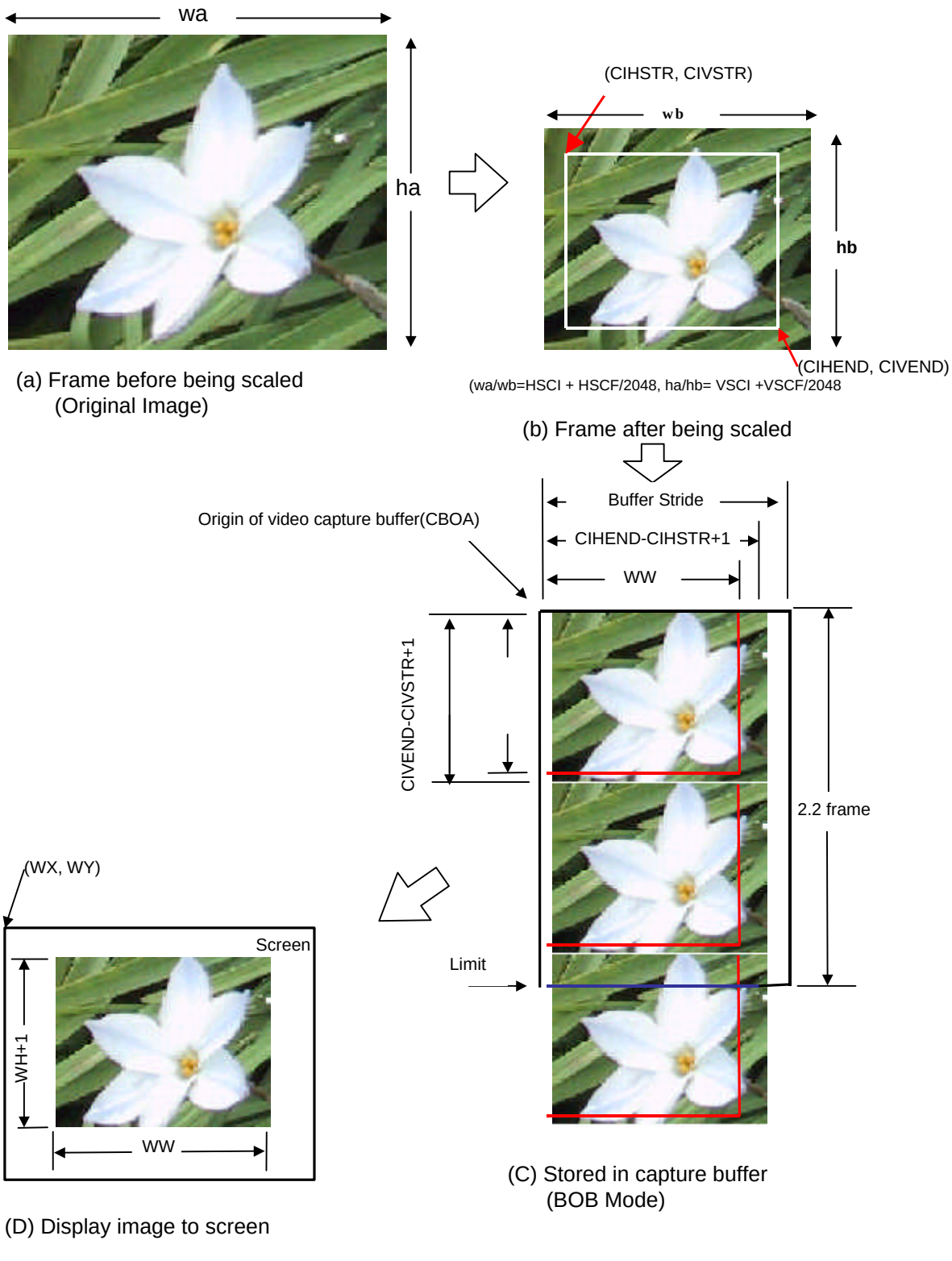
The BOB mode interpolates vertically the odd and even fields that enter every 1/60 sec for one frame. Frames are updated every 1/60 sec.

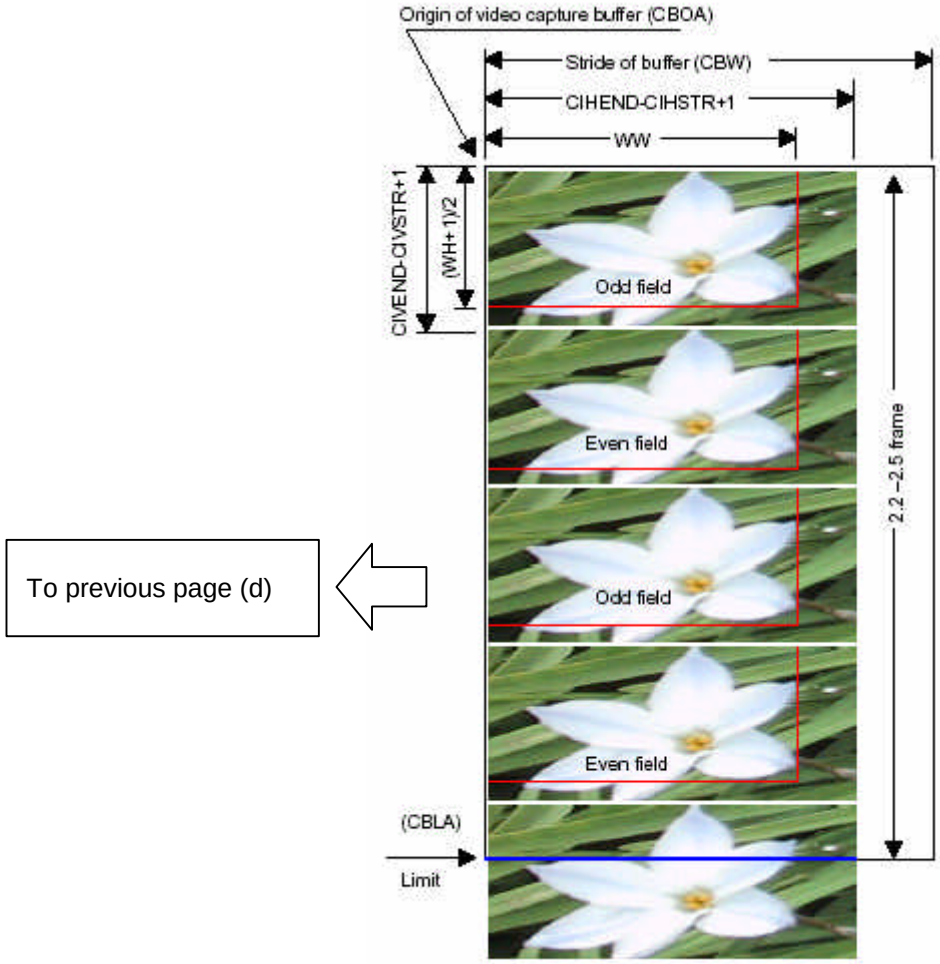
The WEAVE mode merges odd and even fields that enter every 1/60 sec for one frame. For that reason, frames are updated every 1/30 sec.

Because frames are updated every 1/60 sec in the BOB mode, this mode is good for displaying contents that comparatively have movement. Because the WEAVE mode has high vertical resolution, it is optimum for comparatively still displays.

3.3.2 Various parameters

The following shows the various parameters from the original screen to display of the screen.





(c) Store in capture buffer
(WEAVE Mode)

1) Processing with the Capture Scaler

Compresses image to wb*hb size compared to the original image (wa*ha). Vertical and horizontal compression ratios are set with the CSC register's VSCI, VSCF, HSCI, HSCF. Refer to ***Section 6.3*** in the ***MB86292 <Orchid> Specifications*** for details regarding methods of calculating setting values..

Also, the starting and ending points set by CIHSTR, CIVSTR, CIHEND and CIVEND are clipped at the frame after scaling and written to the buffer.

2) Processing with the Buffer Controller

Stores images passed through the buffer region scaler set by the CBOA/CBLA/CBW registers. The buffer is controlled using the ring buffer method.

3) Processing with the Display Controller

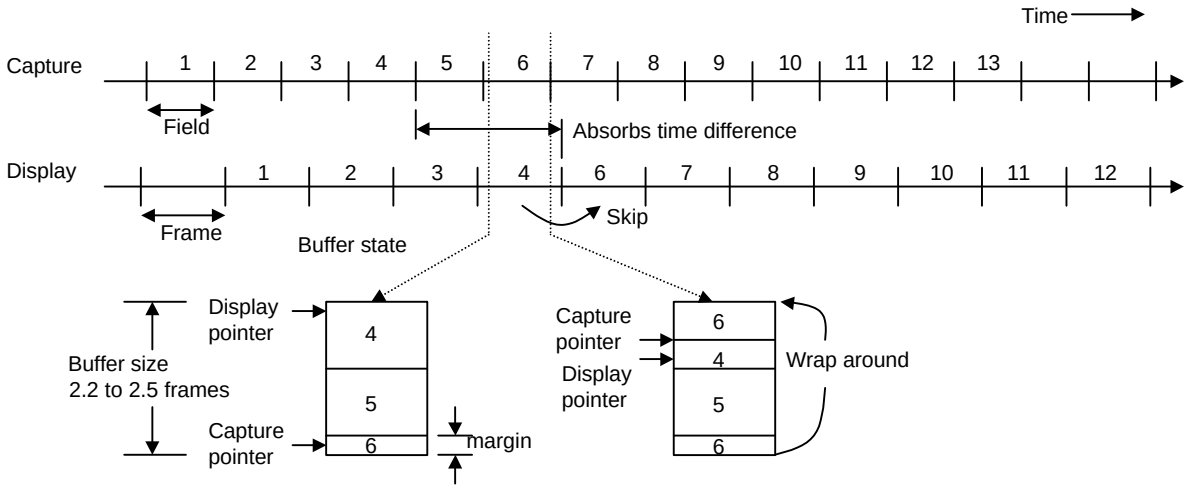
The display controller displays the data stored in the capture buffer in the W layer. The display controller always displays the latest captured frames.

In the WEAVE mode, the display controller generates frames from the odd and even fields for display.

3.3.3 Synchronization of frames

The capture buffer controller writes the sequentially captured rasters to the buffer. On the other hand, the display controller reads the captured-latest frames for display. The result is that frames can be skipped or the same frame can be displayed two or more times.

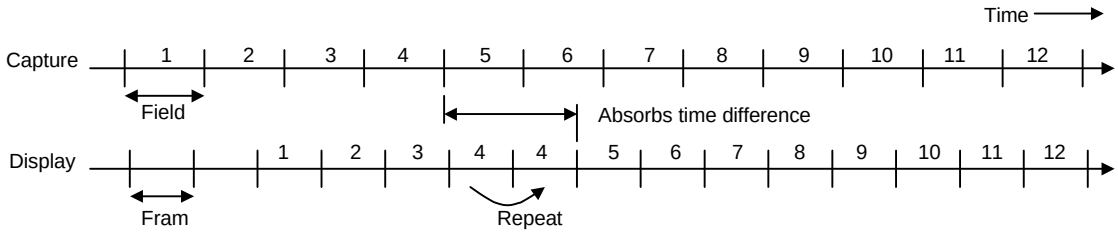
1) When capturing faster than display in the BOB mode:



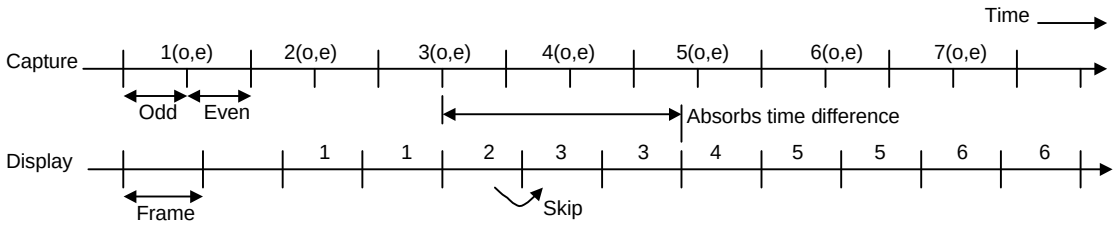
In the operation as shown in the above figure, when the “Display pointer” reaches the last of frame 4, if the “Capture pointer” reaches the last of frame 6, in the worst case, a frame can be skipped. This prevents the “Capture pointer” from passing the “Display pointer.”

To avoid passing the “Capture pointer,” it is necessary to allocate enough margin to the buffer.

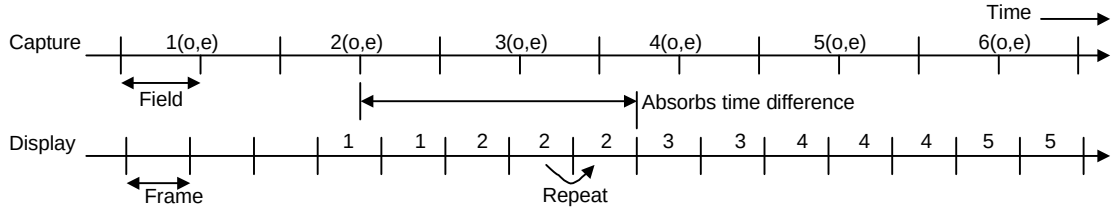
2) When capturing more slowly than display in the BOB mode:



3) When capturing faster than display in the WEAVE mode:



4) When capturing more slowly than display using the WEAVE mode:



3.3.4 Parameter setting values

The following determines each parameter.
The following are variable definitions.

(xs, ys): Image read starting point

(ws, hs): Original image size

(wd, hd): Size to display in window

1) Selecting non-interlacing

The non-interlacing mode is determined by the input image format and the display output size.

n: Raster count around the input image field.

n ≥ hd: BOB and WEAVE conversion unnecessary

n < hd ≤ 2n: BOB or WEAVE

When not using BOB and WEAVE, displays the odd field and even field alternately in the same position. However, in such cases, this is appropriate for displaying in comparatively small windows.

2) Setting compression rate

Refer ***Section 6.3*** in the ***MB86292 <Orchid> Specifications*** to set compression rate.

rh = ws / wd

rv = hs/ hd

Compression rate integers are set to HSCI and VSCI. Fractions are set to HSCF and VSCF.

HSCI * 2048 + HSCF = floor (rh*2048+0.5)

VSCI * 2048 + VSCF = floor (rv*2048+0.5)

*floor calculates the floor function (discard) of the value .

3) Setting image range to store in the capture buffer

CIHSTR = floor(xs/rh+0.5)

CIVSTR = floor(ys/rv+0.5) for not WEAVE mode

= floor(ys/rv/2+0.5) for WEAVE mode

CIHEND = CIHSTR+wd-1

CIVEND = CIVSTR+hd-1 for not WEAVE mode

= CIVSTR+hd/2-1 for WEAVE mode

4) Setting capture buffer and W layer stride

CBW = WW = ceil(wd/32

*ceil calculates the ceiling function (round up) of the value .

5) Setting buffer size

CBLA = CBOA + ceil(K*hd)*CBW*64

K, here, shows the buffer size according to the frame size. This value is set to 2.5 from 2.2. K = 2.2 is a general value for use while K = 2.5 is a safer value.

6) Setting W layer parameter

WW = xd

WH = yd –1

3.3.5 Sample program

This shows sample program when capturing a size of 640 × 480 with the BOB mode (with vertical interpolation).

```
------ When using a driver -------------
#define WIN_WIDTH 640
#deifne WIN_HEIGHT 480
/* Set W layer dimension */
GdcDispDimension(GDC_DISP_LAYER_W, GDC_ENABLE, GDC_16BPP_FORMAT,
 GDC_FLIPMODE_0,0x151800, 0x151800, WIN_WIDTH, WIN_HEIGHT);

/* Set video capture parameter */
GdcCapSetWindowMode(GDC_CAP_YC_MODE, GDC_CAP_CAPTURE_MODE);
GdcCapSetVideoCaptureBuffer(0x151800, (0x151800 + (640*480*2*2.2)),640*2/64);
GdcCapSetImageArea(0,0,639,479);

GdcDispTimingWindow(0, 0,WIN_WIDTH, WIN_HEIGHT);
GdcCapSetVideoCaptureScale(0x0800, 0x0800);
/* Start capture (NTSC) */
GdcCapSetVideoCaptureMode(GDC_CAP_START |GDC_CAP_ENABLE_V_INTERPOLATION|
 GDC_CAP_NTSC);
}
------ When not using a driver -------------
#define DISP_BASE 0xb1fd0000
#define CAP_BASE 0xb1fd8000
unsigned long val;
unsigned long stride;
unsigned long cap_height,cap_size;
stride = 640*2/64;
cap_height = 480;
cap_size = 640;

val=*(unsigned int *)(DISP_BASE + 0x00);
*(unsigned int *)(DISP_BASE + 0x00)=val | 0x00020000; //W On

*(unsigned int *)(CAP_BASE + 0x14) = 0x151800; //CBOA
*(unsigned int *)( CAP_BASE + 0x18) = 0x151800 + (640*480*2*2.2);//CBLA
*(unsigned int *)( CAP_BASE + 0x10) = stride<<16;//CBM
*(unsigned int *)( CAP_BASE + 0x1c) = 0x00000000;//CIHSTR
*(unsigned int *)( CAP_BASE + 0x20) = ((cap_height)<<16) | cap_size;//CIHEND

*(unsigned int *)( DISP_BASE + 0x18) = ( 0<<16) |   0;//WY,WX
*(unsigned int *)( DISP_BASE + 0x1c) = ((cap_height-1)<<16) | cap_size;//WH,WW
*(unsigned int *)( DISP_BASE + 0x34) = 0x151800;//WOA

*(unsigned int *)( CAP_BASE + 0x04) = 0x08000800;//CSC

*(unsigned int *)( DISP_BASE + 0x30) = 0xe0000000 | (stride<<16);//WM
*(unsigned int *)( CAP_BASE + 0x00) = 0x83000000; //VCM
```

3.4 Display Performance

Orchid can overlap 4 layers. However, when using high resolutions or capturing, there may be limitations to the count of layers and pixel data that can be displayed simultaneously according to the data supply capacity of the graphics memory.

4. DRAWING FUNCTION

Orchid sets the draw register and executes the draw command by sending the display list to FIFO when drawing.

If there is no particular specification regarding drawing attributes such as the drawing frame or color, the attributes set just before will be used.

For that reason, there is no need to reset attributes for drawing primitives or for each drawing frame, but create programs in consideration of the relationships of attributes before and after.

4.1 Setting Drawing Frame

The display frame can be set for each layer; on the other hand the drawing frame is set as one current frame. For that reason, it is necessary to set the drawing frame when drawing.

When setting the drawing frame, sets the color format, drawing frame base address and X direction resolution to the MDR0, FBR and XRES registers respectively.

The following shows a drawing frame setting program example. Programs and display lists following this program example do not have draw frames set if not particularly necessary, so they are drawn to the drawing frames specified just before.

< Conditions example >

- Drawing frame address: 0x00000000, Resolution X-axis direction: 640. Y-axis direction 480
- Color format is 16 bits/pixel

```
/*-------------------------------------------------------------------------------------------------------------
Drawing frame setting program example
-------------------------------------------------------------------------------------------------------------*/
/*--------------------- Driver function usage example ------------------------------------------------*/
GdcDrawDimension ( GDC_16BPP_FORMAT, 0x00000000, 0x0280, 0x01E0 );       /* Sets drawing frame */
/*--------------------- Display list usage example ---------------------------------------------------*/
F1010108                    /* Type: set register, data count: 1, address: 108 =>420 */
00008000                            /* MDR0 register direct color mode */
F1010110                    /* Type: set register, data count: 1, address: 110 =>440 */
00000000                            /* FBR register draw frame base address: 0x00000000 */
F1010111                    /* Type: set register, data count: 1, address: 111 =>444 */
00000280                            /* XRES register sets draw screen X axis direction resolution */
```

4.2 Clearing Drawing Frame

No particular commands or functions have been prepared for clearing the drawing frame on Orchid. Use the “rectangle region” for the drawing frame with any color to clear the drawing frame.

Also, when clearing the C, W and M layers, filling with a set color as the transparent color will allow you to see the drawing on the lower heirarchy.

4.3 Issuing Flash Commands

When transferring the display list to Orchid, insert a flash command (frame buffer and Z buffer) after the display list. Issuing this command draws the drawing results remaining in the drawing pipeline to VRAM.

```
/* Flash command issuing example /*
        :           }        Stacked display list
    0xF0C10000          /* Frame buffer flash command */
    0xF0C20000          /* Z buffer flash command /*
```

4.4 Setting Clip Frame

Specifying the clip frame in Orchid allows you to set it to not draw outside of the frame. Besides the clip for drawing (rendering), there is also a clip (view volume clip) for various conversions (geometry conversion). The view volume clip is described on another page.

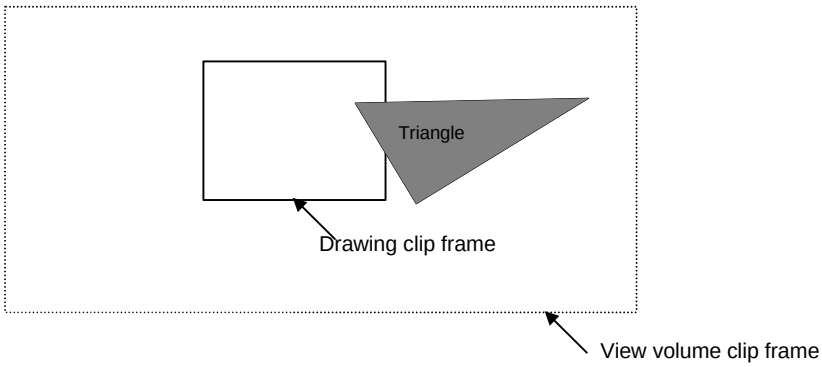
MDR0, CXMIN, CXMAX, CYMIN and CYMAX are the registers to set draw clip processing.

The following page shows a draw clip frame setting program example.

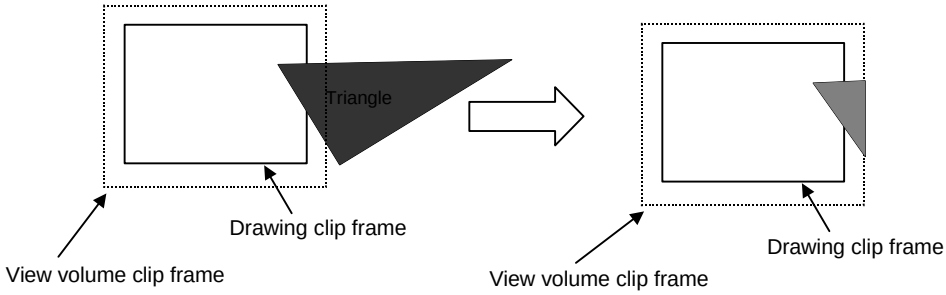
```
/*-------------------------------------------------------------------------------------------------------------------
                Clip frame setting program example
-------------------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function usage example ------------------------------------------------------------*/
GdcDrawClipFrame ( 0x0000, 0x0032, 0x0280, 0x0190 );                    /* Sets clip frame */
GdcSetAttrMisc ( GDC_CLIP, (GDC_CLIP_X_ON | GDC_CLIP_Y_ON));  /* Sets clip processing enabled/disabled */

/*----------------------------------- Display list example ---------------------------------------------------------*/
F1040115                    /*Type: set register, data count: 4, address: 115 =>454 */
00000000                            /* CXMIN register sets clip frame X minimum value */
00000280                            /* CXMAX register sets clip frame X maximum value */
00000032                            /* CYMIN register sets clip frame Y minimum value */
00000190                            /* CYMAX register sets clip frame Y maximum value */
F1010108                    /* Type: set register, data count: 1, address: 108 =>420 */
00008300                            /* MDR0 register sets X, Y direction clip processing */
```

Note that the rendering engine outside of the draw clip frame draws pixels and has a circuit configuration that does not draw according to the clip judgement. For that reason, it is recommended set the view volume clip in a range near the drawing clip and to narrow the pixel range where the rendering engine operates. Refer to ***Section 4.6.2.4*** in the ***MB86292 Specifications*** for details regarding the view volume clip setting.

1. View volume clip frame inappropriately set. Drawing far outside drawing clip frame.



2. Drawing not drawn far outside drawing clip frame by view volume clip
(drawn comparatively faster than 1.)



: Pixel range wherein rendering engine operates.

4.5 Bit Map Drawing (Binary Pattern)

Bit map drawing uses 32 bit unsigned integers as the pattern format where 1 bit corresponds to 1 pixel. When the binary pattern is 1, it is drawn with the set foreground color, when 0, it is drawn with the set background color. It is possible to set not to draw the background using the register setting.

It is possible to set the expand or compress mode with bit map drawing. Set these modes with the MDR0 register's BSH and BSV bits. Also, the foreground and background are set by the FC and BC registers.

The following shows a bit map drawing program example.

<Drawing >

- Bit map foreground color: white; background color: no drawing
- Pattern data: unsigned long type pointer stored in pat_ptr
- Pattern size: 56 × 16

```
/*-----------------------------------------------------------------------------------------------
                    Binary pattern drawing program
-----------------------------------------------------------------------------------------------*/
/*--------------------- Driver function usage example ----------------------------------*/
GdcBitPatternMode ( GDC_BPSCALE_H_EQUIV | GDC_BPSCALE_V_EQUIV );
                                                        /* Sets expand compress mode */
GdcColor ( GDC_WHITE );                                 /* Sets foreground color */
GdcBackColor ( 0x8000 );                                /* Sets background color */
GdcBitPatternDraw ( 320, 240, 56, 16, pat_ptr );        /* Binary pattern drawing */
```

Note: When drawing attributes are not set, those attributes set previously are used.

```
/*------------------ Display list usage example ------------------*/
F1010108            /* Type: set register, data count: 1, address: 108 =>420 */
00008000            /* MDR0 register bit map drawing vertical and horizontal ratio 1 */
F1020120 /* Type: set register, data count: 1, address: 120 =>480 */
00007FFF            /* FC register sets foreground color */
00008000            /* BC register sets background color */
0B430022 /* Type: draw bit map P, Cmd: bit map, data count: 34 */
00F00140            /* Rys, RXs, pattern drawing starting point Y, X coordinates */
00100038            /* Rsize Y, RsizeX pattern data Y, X axes directions widths */
00000000            /* Pattern data */
                                    }  32 pattern data
00000000            /* Pattern data */
```

* 1 bit of the binary pattern format corresponds to the 1 pixel. With 32 bits, it is one pattern format. For that reason, in the above program, the X direction width of the pattern data is 56 pixels so two patterns of data are required per one line. Also, the Y direction axis is 16 pixels, so a total of 32 patterns (16 x 2) are required.

pat_ptr[0]=0xXXXXXXXX, pat_ptr[1]=0xXXXXXXXX <= Two of these for 1 horizontal direction data

↑

pat_ptr [1] lower 1 byte is invalid.

Ex.: If the pattern data X direction width: 120: Y direction width: 20, the necessary data count is 120/32 = 3.75 => 4 data per 1 horizontal direction.

80 are required.

The following shows the pattern data for 56 pixels in the X axis direction width and 16 pixels in the Y axis direction width.

This pattern data is the character pattern written “Fujitsu.”

```
static unsigned long        pat_ptr[] = {
    0x00000000,     0x00000000,     0x00000000,     0x00000000,
    0x00000000,     0x00000000,     0xfe000810,     0x00000000,
    0x80000810,     0x00000000,     0x80000000,     0x10000000,
    0x80000000,     0x10000000,     0x80420810,     0x7e3c4200,
    0xfc420810,     0x10424200,     0x80420810,     0x10404200,
    0x80420810,     0x103c4200,     0x80420810,     0x10024200,
    0x80460810,     0x10424600,     0x803a0810,     0x0e3c3a00,
    0x00000800,     0x00000000,     0x00007000,     0x00000000 };
```

4.6 BLT and Rectangle Drawing

There are types for rectangle drawing and transfer, described below.

1. Rectangle region drawing using a single color
2. Rectangle drawing by transferring between the graphics memory
3. Rectangle drawing by transferring from the host CPU to the drawing frame
4. Data transfer from the host CPU to the Orchid built-in texture memory
5. Data transfer from the graphics memory to the Orchid built-in texture memory

Logic operation processing and transparent processing can be set with the BLT function.

Use the MDR4 register to set the logic operation processing and transparent processing, and use the TColor register to set the transparent color.

4.6.1. Rectangle region drawing using a single color

Clear each drawing frame and buffer using a single color filling.

The following shows a rectangle region filling program example using a single color.

<Drawing >

- Draw color: White
- Draw region: Draws the X, Y directions (60, 40) with (X, Y) = (320, 420) on the upper left

```
/*-------------------------------------------------------------------------------------------------------------------
                Rectangle draw program (fill with single color)
-------------------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function usage example ------------------------------------------------------------*/
GdcSetAttrBlt ( GDC_BLT_TILE, GDC_DISABLE );
                                                    /* Sets filling attributes when using rectangle region drawing */
GdcSetAttrBlt ( GDC_BLEND_MODE, GDC_BLEND_COPY );
                                                    /* Sets filling attributes when using rectangle region drawing */
GdcColor ( GDC_WHITE );                             /* Sets filling color */
GdcBltFill ( 320, 240, 60, 40 );                    /* Rectangle region drawing */
/*------------------ Display list usage example ---------------------------------------------------------------*/
F101010C            /* Type: set register, set data count: 1, address: 10 =>430 */
00000000                        /* Sets MDR4 register */
F1010120            /* Type: set register, set data count: 1, address: 120 =>480 */
00007FFF                        /* FC register sets drawing color */
09410000            /* Type: DrawRectP,Cmd: BltFill*/
00F00140                        /* Rys, RXs, Y, X coordinates starting point to be drawn */
0028003C                        /* RsizeY, RsizeX Y, X direction sizes to be drawn */
```

Note: When drawing attributes are not set, those attributes set previously are used.

4.6.2 Rectangle region transfer drawing between graphics memory

Transfers, copies and draws the rectangle region in the transfer origin to the transfer destination.

By performing transparent processing, it is possible to not draw specified color pixels to the transfer destination.

4.6.2.1 Copying within the same drawing frame

Draws bit maps in the current drawing frame using rectangle transfer.

<Drawing >

- Transfers and copies rectangle regions drawn from the base frame starting point X, Y coordinates (10, 20) to the X, Y direction (60, 40) to the X, Y coordinates (290, 220).
- Blend Mode: logic operation (EQUIV)

```
/*-------------------------------------------------------------------------------------------------------------------
Rectangle transfer drawing program (Copying within the same drawing frame)
-------------------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function usage example ----------------------------------------------*/
GdcSetAttrBlt ( GDC_BLEND_MODE, GDC_BLEND_ROP);      /* Sets blend mode */
GdcSetRop ( GDC_ROP_EQUIV );           /* Sets logic operation mode */
GdcBltCopy ( 10, 20, 290, 220, 60, 40 );    /* Rectangle region drawing */
/*---------------------- Display list usage example ------------------------------------------------*/
F101010C            /* Type: set register, set data count: 1, address: 10C =>430 */
00001300                    /* Sets MDR4 register blend mode */
0D440000            /* Type: BltCopyP, Cmd: TopLeft */
0014000A                    /* SRYs, SRXs, transfer origin region Y, X coordinates */
00DC0122                    /* DRYs, DRXs, transfer destination region Y, X coordinates */
0028003C                    /* BRsizeY, BRsizeX Y, X direction sizes of block to be transferred */
```

Note: When drawing attributes are not set, those attributes set previously are used.

4.6.2.2 Copying with any drawing frame

Draws bit maps between any drawing frames using rectangle block transfer.

<Drawing >

- Perform a transparent copy (transparent color 0x7c00)
- Transfer origin: Drawing frame address: 0x00000000
  Transfer origin rectangle top left X, Y coordinates: (10, 20)
- Transfer destination: Drawing frame address: 0x00096000
  Transfer destination rectangle region top left X, Y coordinates: (30, 50)
- Transfer block size: X direction: 60; Y direction: 40

```
/*-------------------------------------------------------------------------------------------------------
Rectangle transfer drawing program (Copying between any drawing frames)
-------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function usage example ---------------------------------------------*/
GdcSetAttrBlt ( GDC_TRANSPARENT_MODE,GDC_ENABLE );      /* Sets transparent mode */
GdcBltColorTransparent (0x7C00);                        /* Sets logic operation mode */
GdcBltCopyAlt ( 10, 20 , 30, 50, 60, 40, 0, 640,0x96000, 640 );
                                                        /* Copies and draws rectangle region copy and draw */
/*------------------------- Display list usage example ---------------------------------------------*/
F101010C        /* Type: set register, data count: 1, address: 10C =>430 */
00000002            /* MDR4 register sets transparent mode */
F10100A0        /* Type: set register, data count: 1, address: 0A0 =>280 */
00007c00            /* TColor register sets transparent color setting */
0F440000        /* Type: BitCopyAlternateP, Cmd: TopLeft */
00000000            /* SADDR  Transfer origin drawing frame offset address */
00000280            /* SStride  Transfer origin drawing frame X direction memory width */
0014000A            /* SRYs, SRXs, Transfer origin drawing frame rectangle region top left Y, X coordinates */
0012C000            /* DADDR  Transfer destination drawing frame offset address */
00000280            /* DStride  Transfer destination drawing frame X direction memory width */
0032001E            /* DRYs, DRXs, Transfer destination drawing frame rectangle region top left Y, X coordinates */
0028003C            /* BRsizeY, BRsizeX Y, X direction sizes of block to transfer */
```

* At GdcBltCopyAlt, used when the data to draw is not duplicated at the transfer destination and transfer origin frames.

* Color modes for the transfer origin and transfer destination must be the same.

Note: When drawing attributes are not set, those attributes set previously are used.

4.6.3 Transfer, copy and draw to the VRAM from main memory

Transfers and draws rectangle blocks from the main memory to the VRAM (frame buffer).

Performing transparent processing enables not to draw specified color pixels to the transfer destination.

<Drawing >

- Transfer source: Pointer to pattern data on main memory: ptnptr
- Transfer destination: Rectangle region starting point (X and Y) to copy with the current drawing frame: (20, 50)
- Drawing size: X, Y Directions: (128, 128)
- Blend mode: Source copy

```
/*-------------------------------------------------------------------------------------------------------------------
Rectangle transfer drawing program (Copy from the main memory to the current drawing frame)
-------------------------------------------------------------------------------------------------------------------*/
/*-------------------------- Driver function use example ----------------------------------------------------*/
GdcSetAttrBlt ( GDC_BLEND_MODE, GDC_BLEND_COPY );  /* Blend mode setting */
GdcBltDraw ( 20, 50, 128, 128, ptnptr );                    /* Rectangle area transfer and drawing */
/*----------------------- Display list use example ------------------------------------------------------------*/
F101010C        /* Type: Set Register, data count: 1, address: 10C =>430 */
00000000              /* MDR4 register blend mode setting */
0B422002        /* Type: Draw Bit map P, Cmd: Bit Draw, data count: 8194 */
00320014              /* RYs, RXs, Transfer destination drawing frame rectangle area top left Y, X coordinates */
00800080              /* RsizeY, RsizeX Y, X direction sizes of block to transfer */
3E533E53              /* Pattern data */   }
                                             } 8192 (128 × 128/2) patterns data
3E533E53              /* Pattern data */   }
```

Note: When drawing attributes are not set specifically, those attributes set earlier are used as they are.

4.6.4 Transfer of texture and tile patterns from main memory

Transfers texture and tile patterns from the main memory to the Orchid internal TexRAM and frame memory.

4.6.4.1 Transfer to internal TexRAM

The internal TexRAM is 8 KB. Up to one 64 × 64 pattern in 16 bit color format can be stored and up to two 64 × 64 pattern in 8 bit color format (in tiling) can be stored. Also, if the patterns are small, a multiple can be stored by specifying the corresponding addresses when drawing (for example, four with 32 × 32 patterns).

In texture mapping, using the internal TexRAM realizes faster drawing than placing Tex patterns on the frame memory.

Tiling corresponds only to the internal TexRAM.

<Program example>

- Tex pattern: Pointer: ptnptr1 (Size: 32 × 32), ptnptr 2 (Size: 16 × 16)
  ptnptr3 (Size: 16 × 16)

```
/*-------------------------------------------------------------------------------------------------------------------
Stores 3 types of texture (tiles) patterns from the main memory to the internal texture memory.
-------------------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function use example ------------------------------------------------------------*/
GdcTextureLoadInt ( 32*32, ptnptr0, 0x0 );
GdcTextureLoadInt ( 16*16, ptnptr1, 0x800 );
GdcTextureLoadInt ( 16*16, ptnptr2, 0xA00 );
/*---------------------- Display list use example ---------------------------------------------------------------*/
F101011B        /* Type: Set Register, data count: 1, address: 11B =>46C */
00000000                /* TOA Register  Pattern storage address setting */
11490200        /* Type: LoadTextureP, Cmd: LoadTex, data count: 200 */
xxxxxxxx                /* Tex pattern from here */
                                                        } 32 × 32/2 (When 16 BPP) data
xxxxxxxx                /* Tex pattern to here */
F101011B        /* Type: Set Register, data count: 1, address: 11B =>46C */
00000800                /* TOA Register  Pattern storage address setting */
11490080        /* Type: LoadTextureP, Cmd: LoadTex, data count: 80 */]
xxxxxxxx                /* Tex pattern from here */
                                                        } 16 × 16/2 (When 16 BPP) data
xxxxxxxx                /* Tex pattern to here */
F101011B        /* Type: Set Register, data count: 1, address: 11B =>46C */
00000A00                /* TOA Register  Pattern storage address setting */
11490080        /* Type: LoadTextureP, Cmd: LoadTex, data count: 80 */
xxxxxxxx                /* Tex pattern from here */
                                                        } 16 × 16/2 (When 16 BPP) data
xxxxxxxx                /* Tex pattern to here */
```

4.6.4.2 Transfer to frame buffer

A maximum size of 256 × 256 texture pattern can be used by placing the Tex pattern on the frame buffer.

Transferring texture patterns to the frame buffer uses the same method as described in ***Section 4.6.3***. For that reason, after completing the transfer, it is necessary to return the FBR and XRES registers to their original settings.

<Program example>

- Tex pattern: Pointer: ptnptr1 (Size: 256 × 256)

```
/*-------------------------------------------------------------------------------------------------------
Stores texture patterns from main memory to frame memory.
-------------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function use example ----------------------------------------------*/
GdcTextureDimension(0x00f00000,256,256);
GdcTextureLoadExt ( 256*256, ptnptr0, 0x0f000000 );
                Note: Before using the GdcTextureLoadExt function, set the size to transfer with the
                      GdcTextureLoadExt ( 256 × 256, ptnptr0, 0x0f000000 ); GdcTextureDimension functions.
/*---------------------------- Display list use example -------------------------------------------*/
F1010113        /* Type: Set Register, data count: 1, address: 113 =>44C */
00000000                /* TBR Register  Pattern storage address setting */
F1010119        /* Type: Set Register, data count: 1, address: 119 =>464 */
01000100                /* TXS Register  Tex pattern size setting */
F1010110        /* Type: Set Register, data count: 1, address: 110 =>440 */
0F000000                /* FBR Register  Pattern storage address setting */
F1010111        /* Type: Set Register, data count: 1, address: 111 =>444 */
00000000                /* XRES Register  X axis direction resolution setting */
0B428000        /* Type: Draw Bit map P, Cmd: Bit Draw, data count: 0x8000 */
00000000                /* Rys, RXs, Draw start coordinates setting of Y, X */
01000100                /* RsizeY, RsizeX Y, X drawing size setting */
xxxxxxxx                /* Tex pattern from here */
                                                        } 256 × 256/2 data
xxxxxxxx                /* Tex pattern to here */
```

4.6.5 Transfer and copy from texture pattern graphics memory to internal texture memory

Temporarily stores texture patterns on the graphics memory. When using those texture patterns, drawing is faster by transfer copying them to the Orchid internal texture memory.

Drawing attribute can not be set particularly.

The following shows an example program.

<Transfer >

- Transfer source: Stored in the graphics memory address “0000”.
- Transfer destination: Internal texture memory address “0x800”

```
/*-----------------------------------------------------------------------------------------------------------
Rectangle transfer program
(Copy from texture pattern graphics memory to the internal texture memory)
-----------------------------------------------------------------------------------------------------------*/
/*--------------------- Driver function use example ----------------------------------------------*/
GdcBltTexture ( 0x000000, 32, 0, 0, 32, 32, 0x800*);
/*---------------------- Display list use example ------------------------------------------------*/
13490000          /*Type: BltTextureP,Cmd: LoadTile*/
00000000                  /* SrcADDR  Transfer source frame origin address */
00000020                  /* SrcStride  Transfer source frame X direction memory width */
00000000                  /* SrcRectYs, Xs,  Y, X coordinates from transfer source rectangle area top left origin */
00200020                  /* BrsizeY, X  Y, X direction sizes of the rectangle area */
00000800*                 /* DADDR  Texture pattern storage destination offset address */
```

4.7 Erasing Hidden Planes

To erase hidden planes using the Z comparison on three-dimensional object drawing, use one frame memory as the Z buffer. 2 bytes of Z buffer area are required each 1 pixel of the same shape as the drawing frame.

The Z value comparison method can be selected by setting the comparison calculation equation for the MDR1 and MDR2 registers.

Note that a Z buffer clear command is not available as a display list. For that reason, use the BltFill command to clear the Z buffer.

<Program example>

- Memory address to use as the Z buffer: 0x00384000
- Drawing frame size: 640 × 480

```
/* When using driver function */
    GdcBufferCreateZ(0x00384000);
    GdcBufferClearZ ();

/*------------------ When using display list ------------------*/
    0xF1010112          /* Type: Set Register, data count: 1, address: 112 =>448 */
    0x0038400                   /* ZBR  Z buffer base address setting */
    0xF1010120          /* Type: Set Register, data count: 1, address: 120 =>480 */
    0x0000FFFF                  /* FC  Foreground color setting */
    0xF1010110          /* Type: Set Register, data count: 1, address: 110 =>440 */
    0x00348000                  /* FBR Drawing frame base set to Z buffer */
    0x94100000          /* Type: DrawRectP, Cmd: BltFill */
    0x00000000                  /* RYs, RXs   Y, X coordinates setting */     } Executes clear
    0x01E00280                  /* RsizeY, RsizeX  Y, X area setting */
```

4.8 Drawing Using Geometry Commands

Orchid integrates a geometry engine that transforms vertex coordinates into the drawing (device) coordinates from object coordinates.

4.8.1 Initializing the geometry engine

For the first time of geometry process initialize the geometry engine after starting Orchid.

The geometry engine is initialized by issuing a command.

```
/*------------------When using driver function------------------*/
GdcGeoInitialize();
/*-------------------- When using display list -------------------*/
0x40000000    /*Type: G_Init */
```

4.8.2 Various parameters settings

To use the geometry engine for drawing, it is necessary to set the following parameters in advance.

4.8.2.1 Geometry attribute settings

Sets the parameter attributes to process with the geometry engine.
Corresponds to Orchid registers GMDR0 to 2.
Parameters to be set are shown below.

- GMDR0 register (Vertex geometry attribute setting)

  Sets using the driver function GdcGeoSetAttrMisc() and ***SetRegister*** command.

  Input data format

  Vertex color presence

  Z coordinates presence

  Texture coordinates presence

- GMDR1 register (Line drawing geometry attribute setting)

  Sets using the driver function GdcSetAttrLine and ***SetRegister*** command.

  Broken line pattern reference position

  Ending point drawing mode setting

  Anti-alias mode setting

- GMDR2 register (Triangle drawing geometry attribute setting)

  Mode that can define triangle surface direction and does not draw back plane (back face curling).

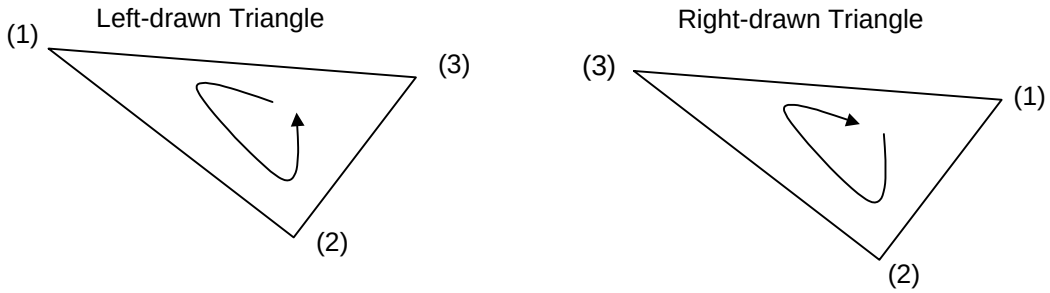
  This function is enabled only in 3D.

  Sets using the driver function GdcGeoSetAttrSurf() and ***SetRegister*** command.

  Surface definition setting

  Back plane drawing setting

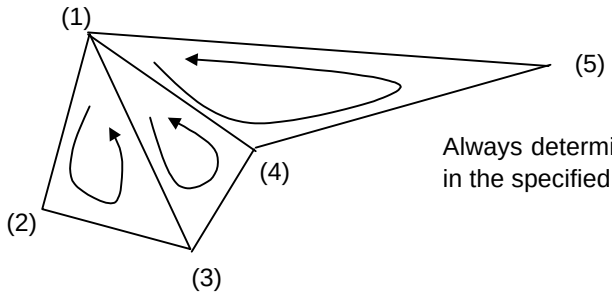
  Back face curling determines the front and back planes in the order of the vertex to be specified.



* Circled numbers show the specifying order of the vertex.

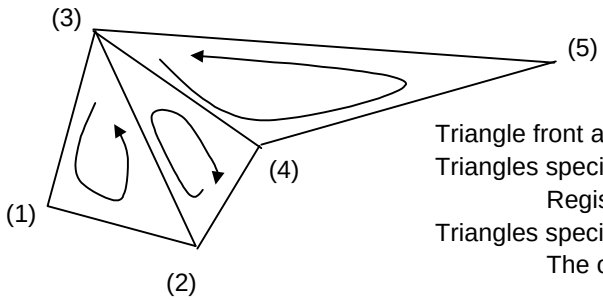
Also, the method used to decide differs according to the ***Traingle_Fan*** and the ***Triangle_Strip*** commands.

For ***Triangle_Fan*** (When front plane is set to be counterclockwise)



Always determines front and back planes
in the specified triangle direction

For ***Triangle_Strip*** (When front plane is set to be counterclockwise)



Triangle front and back planes are determined by:
Triangles specified by an odd number:
Register setting is front
Triangles specified by an even number:
The opposite of the register setting is front

4.8.2.2 Rendering attribute settings

Sets the object attributes to be rendered.

Corresponds to MDR 1 to 3.

Parameters to be set are shown below.

Note that the geometry and rendering attributes must both be set to the appropriate values. Refer to ***MB86291 Specifications*** for details.

- MDR1 register (Line and point drawing attribute setting)
  Sets using the driver function GdcSetAttrLine() and ***SetRegister*** command.
- MDR2 register (Triangle drawing attribute setting)
  Sets using the driver function GdcSetAttrSurf() and ***SetRegister*** command.
- MDR3 register (Texture mapping drawing attribute setting)
  Sets using the driver function GdcSetAttrTexture() and ***SetRegister*** command.

4.8.2.3 4 × 4 Queue setting

Sets the 4 × 4 queue to perform an MVP transformation (from an object coordinates to a clip coordinates).

<Program example>

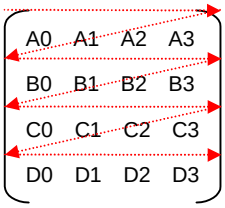
- 4 × 4 Queue is stored in the float type pointer *mat.

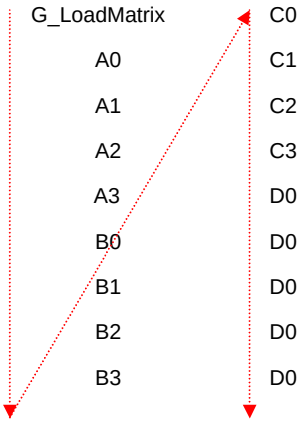
```
/*----------------- When using driver function --------------*/
        GdcGeoLoadMatrixf(mat);
/*------------------ When using display list ------------------*/
        0x43000000          /*Type: G_LoadMatrix */
        0xXXXXXXXX          /*Matrix a0*/
        0xXXXXXXXX          /*Matrix a1*/

        0xXXXXXXXX          /*Matrix d2*/
        0xXXXXXXXX          /*Matrix d3*/
```

A separate program, “3D Graphics Library” is available to generate queues for MVP transformation.

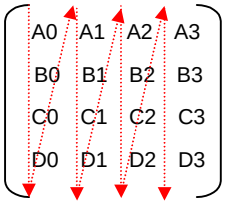
The pointer corresponding to the queue setting function GdcGeoLoadMatrix in the MB86290 Series Graphics Driver and the display list to load the queue to Orchid differ with regard to the lining up of the queue.

Lineup of queue and display list when loading the display list to ***Scarlet***:





Lineup of queue pointer to be passed to the driver function GdcGeoLoadMatrix()



Driver use example

```
Flaot Mat[] = {   A0,B0,C0,D0, A1,B1,C1,D1, A2,B2,C2,D2,
                  A3,B3,C3,D3}
GdcGeoLoadMatrix(Mat);
```

* Driver automatically changes lineup of queue when loading queue to ***Scarlet***.

4.8.2.4 View volume clip setting

Sets the clip boundary value in the view volume clipping to the X, Y, Z and W coordinates as required.

Setting the view volume clip properly can reduce pixel drawing of no use and make drawing speed fast. See ***Section 4.4*** and refer to ***MB86291 Specifications*** for details.

Note that view volume clip can be turned ON and OFF. See section ***5.1*** and refer to ***MB86291 Specifications*** for details.

<Program example>

```
/*------------------When using driver function----------------*/
        GdcGeoViewVolumeXYClip (xmin, xmax, ymin, ymax);
        GdcGeoViewVolumeZClip (zmin, zmax);
        GdcGeoViewVolumeWClip (wmin);
/*------------------ When using display list ------------------*/
        0x44000000          /* Type: G_ViewVolumeXYClip */
        0xXXXXXXXX                  /*XMin*/
        0xXXXXXXXX                  /*XMax*/
        0xXXXXXXXX                  /*YMin*/
        0xXXXXXXXX                  /*YMax*/
        0x45000000          /* Type: G_ViewVolumeZClip */
        0xXXXXXXXX                  /*ZMin*/
        0xXXXXXXXX                  /*ZMax*/
        0x46000000          /*Type: G_ViewVolumeWClip */
        0xXXXXXXXX                  /*WMin*/
```

4.8.2.5 View port and depth range settings

Sets the view port and depth range for transformation from the regular device coordinates to the drawing device coordinates as required.

View port and depth range transformation transforms normalized coordinates (queue transformation, clip, 3D-2D post conversion) to draw (device) coordinates.

The following shows a detailed description of the calculation method.

Xdc = Xndc * Xscale + Xoffset
Ydc = Yndc * Yscale + Yoffset
Zdc = Zndc * Zscale + Zoffset

* Xdc to Zdc: Draw (device) coordinates
Xndc to Zndc: Normalized device coordinates
Xscale to Zscale: Scale values of X, Y and Z
Xoffset to Zoffset: Offset values of X, Y and Z

<Program example>

```
/*------------------ When using driver function --------------*/
        GdcGeoNdcDcViewportCoeff (scalex, offsetx, scaley, offsety);
        GdcGeo NdcDcDepthCoef (scalez, offsetz);
/*------------------- When using display list -------------------*/
        0x41000000          /* Type: G_Viewport Command*/
        0xXXXXXXXX                  /*X  Scale*/
        0xXXXXXXXX                  /*X  Offset*/
        0xXXXXXXXX                  /*Y  Scale*/
        0xXXXXXXXX                  /*Y  Offset*/
        0x42000000          /*Type: G_DepthRange Command*/
        0xXXXXXXXX                  /*Z  Scale*/
        0xXXXXXXXX                  /*Z  Offset*/
```

4.8.3 Point drawing

Issues point drawing command with the various settings to perform MVP transformation, geometry and rendering attributes set.

<Program example>

- Drawing color: White
- Format: Flowing decimal

```
/* When using driver function */
        GdcColor(0x7fff);
        GdcGeoPrimType (GDC_POINTS);
        GdcGeoDrawVertex2Df( x, y);
        GdcGeo PrimEnd();
/*------------------ When using display list ------------------*/
        0xF1010120      /* Type: Set Register, data count: 1, address: 480 =>FC */
        0x00007FFF              /* Drawing color setting */
        0x21000000      /*Type:G_Begin Command:Points*/
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float X */
        0xXXXXXXXX              /*float Y */
        0x23000000      /*Type:G_End */
```

4.8.4 Line drawing

Issues the line drawing command with the various settings to perform MVP transformation, geometry and rendering attributes set.

There are independent lines and continuous lines for line strip drawing commands for line drawing.

Line geometry and drawing attributes are set with the GMDR0 and 1 registers and MDR1 register.

4.8.4.1 Multiple line drawing

<Program example>

Draws broken lines as continuous lines with the following attributes.

- Drawing color: White
- Line width: 5 pixels
- Format: Flowing decimal
- Line pattern: Ipat
- Draws line portion (x0,y0,z0)=>(x1,y1,z1) and line portion (x2,y2,z2)=>(x3,y3,z3)

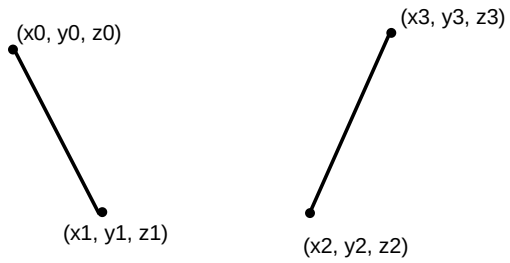


Fig. Drawing Example

```
/* When using driver function */
        GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_ENABLE);
        GdcColor(0x7FFF);
        GdcSetLinePattern(lpat);
        GdcSetAttrLine(GDC_DEPTH_TEST, GDC_ENABLE);
        GdcSetAttrLine(GDC_DEPTH_FUNC, GDC_DEPTH_LEQUAL);
        GdcSetAttrLine(GDC_BROKEN_LINE, GDC_ENABLE);
        GdcSetAttrLine(GDC_ LINE_WIDTH, GDC_LINES_WIDTH_5);
        GdcGeoPrimType (GDC_LINES);
        GdcGeoDrawVertex3Df( x0, y0, z0);
        GdcGeoDrawVertex3Df( x1, y1, z1);
        GdcGeoDrawVertex3Df( x2, y2, z2);
        GdcGeoDrawVertex3Df( x3, y3, z3);
        GdcGeo PrimEnd();

/* When using display list */
        0xF1012040          /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
        0x00000004                  /* Vertex parameter setting */
        0xF1010120          /* Type: Set Register, data count: 1, address: 480 =>FC */
        0x00007FFF                  /* Drawing color setting */
        0xF1010123          /* Type: Set Register, data count: 1, address: 48C =>BLP */
```

```
0xXXXXXXXX          /* Broken line pattern setting */
0xF1010109      /* Type: Set Register, data count: 1, address: 424 =>MDR1 */
0x0408061C          /* Line drawing mode setting */
0x21010000      /*Type:G_Begin Command:Lines*/
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x0 */
0xXXXXXXXX          /*float y0 */
0xXXXXXXXX          /*float z0 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x1 */
0xXXXXXXXX          /*float y1 */
0xXXXXXXXX          /*float z1 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x2 */
0xXXXXXXXX          /*float y2 */
0xXXXXXXXX          /*float z2 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x3 */
0xXXXXXXXX          /*float y3 */
0xXXXXXXXX          /*float z3 */
0x23000000      /*Type:G_End */
```

Terminate drawing with the ***G_End*** command temporarily when the vertex (x1, y1) has been sent in order to change color attributes with the line portion (x0,y0)=>(x1,y1) and the line portion (x2,y2,)=>(x3,y3). Then, draw the line portion (x2,y2,)=>(x3,y3) with ***G_BeginCont*** to process at high speed.

4.8.4.2 Consecutive line drawing

<Program example>

Draws broken lines as consecutive lines with the following attributes.

- Drawing color: White
- Line width: 5 pixels
- Format: Floating point
- Line pattern: Ipat
- Draws line portion (x0,y0)=>(x1,y1)=>(x2,y2,)

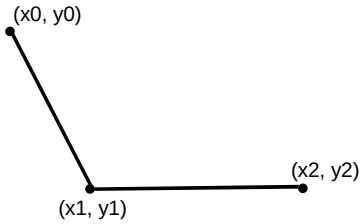


Fig. Drawing Example

```
/* When using driver function */
        GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_DISABLE);
        GdcColor(0x7FFF);
        GdcSetLinePattern(lpat);
        GdcSetAttrLine(GDC_DEPTH_TEST, GDC_DISABLE);
        GdcSetAttrLine(GDC_ LINE_WIDTH, GDC_LINES_WIDTH_5);
        GdcGeoPrimType (GDC_POLYLINE);
        GdcGeoDrawVertex3Df( x0, y0);
        GdcGeoDrawVertex3Df( x1, y1);
        GdcGeoDrawVertex3Df( x2, y2);
        GdcGeo PrimEnd();

/* When using display list */
0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
0x00000000              /* Vertex Parameter Setting */
0xF1010120      /* Type: Set Register, data count: 1, address: 480 =>FC */
0x00007FFF              /* Drawing color setting */
0xF1010123      /* Type: Set Register, data count: 1, address: 48C =>BLP */
0xXXXXXXXX              /* Broken line pattern setting */
0xF1010109      /* Type: Set Register, data count: 1, address: 424 =>MDR1 */
0x04080600              /* Line drawing mode setting */
0x21050000      /*Type:G_Begin Command:Line_Strip*/
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x0 */
0xXXXXXXXX               /*float y0 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x1 */
0xXXXXXXXX              /*float y1 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x2 */
0xXXXXXXXX              /*float y2 */
0x23000000      /*Type:G_End */
```

4.8.5 Triangle drawing

Issues triangle drawing command with the various settings to perform MVP transformation, geometry and rendering attributes set.
There are triangle, triangle fan and triangle strip drawing commands for triangle drawing.
Triangle geometry and drawing attributes are set with the GMDR0 and 1 registers and MDR2 register.

- Performing tiling

  For tiling, Pattern data can be used only with the built-in TexRAM.

  Set to correspond to the TIS and TOA registers.

- When texture mapping is used:

  Set the MDR3 and TXS registers, and set the TOA register when using the built-in TexRAM and the TBR register when placing the Tex pattern in the frame buffer.

See ***4.6.4*** and ***4.6.5*** for details regarding how to transfer Tex (tile) patterns.

4.8.5.1 Multiple drawing of independent triangles

<Program example>

Draws triangles having drawing attributes with the following attributes.

Triangle 1:

- Format: Floating point
- Vertex: (x0,y0,z0),(x1,y1,z1),(x2,y2,z2)
- Tiling: (Pointer ptnptr0 (Size: 64 x 64)

Triangle 2:

- Format: Floating point
- Vertex: (x3,y3,z3),(x4,y4,z4),(x5,y5,z5)
- Vertex color: (R0,G0,B0), (R1,G1,B1), (R2,G2,B2)
- Texture mapping: (Modulate, Pointer: ptnptr1 (Size: 32 × 32)
- Texture coordinates: (s0,t0,), (s1,t1), (s2,t2)

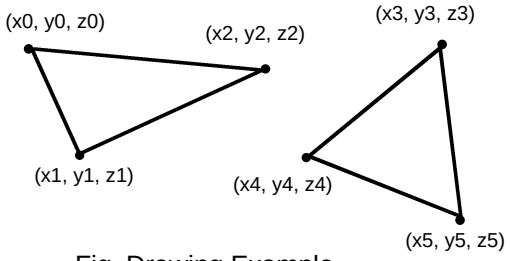


Fig. Drawing Example

```
/* When using driver function */
Triangle 1 geometry attribute settings
    GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_ENABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
    GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_FLAT);
    GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_ENABLE);
    GdcSetAttrSurf(GDC_DEPTH_FUNC, GDC_DEPTH_LEQUAL);
    GdcSetAttrSurf(GDC_BLEND_MODE, GDC_BLEND_COPY);
    GdcSetAttrSurf(GDC_ TEXTURE_SELECT, GDC_SELECT_TILE);
    GdcTextureLoadInt(64*64,ptnptr0,0);
    GdcTextureMemoryMode(GDC_TEX_MEM_MODE_INT);
    GdcTextureDimension(0,64,64);
    GdcGeoPrimType (GDC_TRIANGLES);
    GdcGeoDrawVertex3Df( x0, y0, z0);
    GdcGeoDrawVertex3Df( x1, y1, z1);
    GdcGeoDrawVertex3Df( x2, y2, z2);
    GdcGeo PrimEnd();

    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_DISABLE);
    GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_SMOOTH);
    GdcSetAttrSurf(GDC_ TEXTURE_SELECT, GDC_SELECT_TEXTURE);
    GdcSetAttrTexture(GDC_ TEXTURE_PERSPECTIVE, GDC_ENABLE);
    GdcSetAttrTexture(GDC_ TEXTURE_BLEND, GDC_TEXTURE_MODULATE);
    GdcTextureLoadInt(32*32,ptnptr1,0);
    GdcTextureDimension(0,32,32);
```

Triangle 1 geometry attribute settings

Triangle 1 drawing

Set part differing to triangle 1 drawing attributes

```
GdcGeoPrimType (GDC_TRIANGLES);
GdcGeoTexCoord2Df(s0,t0);
GdcVertexColor3f(R0,G0,B0);
GdcGeoDrawVertex3Df( x3, y3, z3);
GdcGeoTexCoord2Df(s1,t1);
GdcVertexColor3f(R1,G1,B1);
GdcGeoDrawVertex3Df( x4, y4, z4);
GdcGeoTexCoord2Df(s2,t2);
GdcVertexColor3f(R2,G2,B2);
GdcGeoDrawVertex3Df( x5, y5, z5);
GdcGeo PrimEnd();
```

Triangle 2 drawing

```
/* When using display list */
0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
0x00000005              /* Vertex parameter setting */
0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
0x1000061C              /* Triangle drawing attributes setting */
0xF101011B      /* Type: Set Register, data count: 1, address: 468 =>TOA */
0x00000000              /* Built-in TexRAM address setting */
0x11480800      /* Type: Load TextureP, Cmd: LoadTexture data count: 0x800 */
0xXXXXXXXX              /* Tex pattern from here */
      ⋮
0xXXXXXXXX              /* Tex pattern to here */
0xF101011A      /* Type: Set Register, data count: 1, address: 468 =>TIS */
0x00400040              /* Tile size setting */
0x21030000      /*Type:G_Begin Command:Triangles*/
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x0 */
0xXXXXXXXX              /*float y0 */
0xXXXXXXXX              /*float z0 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x1 */
0xXXXXXXXX              /*float y1 */
0xXXXXXXXX              /*float z1 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX              /*float x2 */
0xXXXXXXXX              /*float y2 */
0xXXXXXXXX              /*float z2 */
0x23000000      /*Type:G_End */
0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
0x0000000f              /* Vertex parameter setting */
0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
0x2000061D              /* Triangle drawing attributes setting */
0xF101010B      /* Type: Set Register, data count: 1, address: 42C =>MDR3 */
0x00010009              /* Texture mapping drawing attributes setting */
0xF101011B      /* Type: Set Register, data count: 1, address: 468 =>TOA */
0x00000000              /* Built-in TexRAM address setting */
0x11480200      /* Type: Load TextureP, Cmd: LoadTexture data count: 0x200 */
0xXXXXXXXX              /* Tex pattern from here */
      ⋮
0xXXXXXXXX              /* Tex pattern to here */
0xF1010119      /* Type: Set Register, data count: 1, address: 464 =>TXS */
```

64*64/2

Triangle 1 drawing

32*32/2

```
0x00200020          /* Texture size setting */
0x21030000      /*Type:G_Begin Command:Triangles*/
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x3 */
0xXXXXXXXX          /*float y3 */
0xXXXXXXXX          /*float z3 */
0xXXXXXXXX          /*float R0 */
0xXXXXXXXX          /*float G0 */
0xXXXXXXXX          /*float B0 */
0xXXXXXXXX          /*float S0 */
0xXXXXXXXX          /*float T0 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x4 */
0xXXXXXXXX          /*float y4 */
0xXXXXXXXX          /*float z4 */
0xXXXXXXXX          /*float R1 */
0xXXXXXXXX          /*float G1 */
0xXXXXXXXX          /*float B1 */
0xXXXXXXXX          /*float S1 */
0xXXXXXXXX          /*float T1 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x5 */
0xXXXXXXXX          /*float y5 */
0xXXXXXXXX          /*float z5 */
0xXXXXXXXX          /*float R2 */
0xXXXXXXXX          /*float G2 */
0xXXXXXXXX          /*float B2 */
0xXXXXXXXX          /*float S2 */
0xXXXXXXXX          /*float T2 */
0x23000000      /*Type:G_End */
```

4.8.5.2 Triangle strip drawing

This draws by connecting the vertices (n-2), (n-1) and (n) specified by the triangle fan.

Drawing is executed by transferring only the vertex coordinates at any time so the display list is greatly reduced.

<Program example>

- Format: Floating point
- Vertex: (x0,y0,z0),(x1,y1,z1),(x2,y2,z2), (x3,y3,z3)
- α Blend

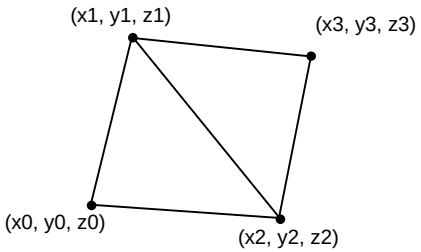


Fig. Drawing Example

```
/* When using driver function */
    GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_ENABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
    GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_FLAT);
    GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_ENABLE);
    GdcSetAttrSurf(GDC_DEPTH_FUNC, GDC_DEPTH_LEQUAL);
    GdcSetAttrSurf(GDC_BLEND_MODE, GDC_BLEND_ALPHA);
    GdcSetAttrSurf(GDC_ TEXTURE_SELECT, GDC_SELECT_PLAIN);
    GdcSetAlpha(0x80);
    GdcGeoPrimType (GDC_TRIANGLE_STRIP);
    GdcGeoDrawVertex3Df( x0, y0, z0);
    GdcGeoDrawVertex3Df( x1, y1, z1);
    GdcGeoDrawVertex3Df( x2, y2, z2);
    GdcGeoDrawVertex3Df( x2, y2, z2);

/* When using display list */
    0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
    0x00000004        /* Peak Parameter Setting */
    0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
    0x0000069C        /* Triangle Draw Attributes Setting */
    0xF1010122      /* Type: Set Register, data count: 1, address: 488 =>ALF */
    0x00000080        /* Blend Coefficient Setting */
    0x21070000      /*Type:G_Begin Command:Triangle_Strip*/
    0x30000000      /*Type:G_Vertex */
    0xXXXXXXXX          /*float x0 */
    0xXXXXXXXX          /*float y0 */
    0xXXXXXXXX          /*float z0 */
    0x30000000      /*Type:G_Vertex */
    0xXXXXXXXX          /*float x1 */
    0xXXXXXXXX          /*float y1 */
```

```
0xXXXXXXXX          /*float z1 */
0x30000000      /*Type:G_Vertex */
0xXXXXXXXX          /*float x2 */
0xXXXXXXXX          /*float y2 */
0xXXXXXXXX          /*float z2 */
0x23000000      /*Type:G_End */
```

4.8.5.3 Triangle fan drawing

This draws by connecting the vertices (n-1) and (n) centering on the vertex 0 specified by the triangle fan.

Drawing is executed by transferring only the peak coordinates at any time so the display list is greatly reduced.

<Program example>

- Format: Floating point
- Vertex: (x0,y0),(x1,y1),(x2,y2), (x3,y3)
- Logic Operation: OR

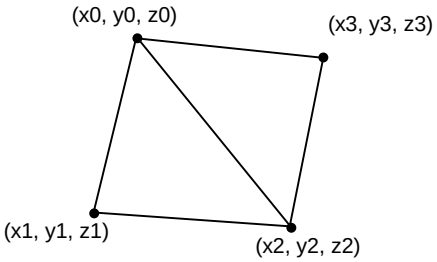


Fig. Drawing Example

```
/* When using driver function */
    GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_ENABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
    GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_FLAT);
    GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_ENABLE);
    GdcSetAttrSurf(GDC_DEPTH_FUNC, GDC_DEPTH_LEQUAL);
    GdcSetAttrSurf(GDC_BLEND_MODE, GDC_BLEND_ROP);
    GdcSetAttrSurf(GDC_ TEXTURE_SELECT, GDC_SELECT_PLAIN);
    GdcSetRop(GDC _ROP_OR);
    GdcGeoPrimType (GDC_TRIANGLE_FAN);
    GdcGeoDrawVertex3Df( x0, y0, z0);
    GdcGeoDrawVertex3Df( x1, y1, z1);
    GdcGeoDrawVertex3Df( x2, y2, z2);
    GdcGeoDrawVertex3Df( x2, y2, z2);
    GdcGeo PrimEnd();
```

```
/* When using display list */
    0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
    0x00000004          /* Vertex parameter setting */
    0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
    0x00000E1C          /* Triangle drawing attributes setting */
    0x21080000      /*Type:G_Begin Command:Triangle_Fan*/
    0x30000000      /*Type:G_Vertex */
    0xXXXXXXXX          /*float x0 */
    0xXXXXXXXX          /*float y0 */
    0xXXXXXXXX          float z0 */
    0x30000000      /*Type:G_Vertex */
    0xXXXXXXXX          /*float x1 */
    0xXXXXXXXX          /*float y1 */
    0xXXXXXXXX          /*float z1 */
    0x30000000      /*Type:G_Vertex */
    0xXXXXXXXX          /*float x2 */
    0xXXXXXXXX          /*float y2 */
    0xXXXXXXXX          /*float z2 */
    0x23000000      /*Type:G_End */
```

4.8.6 Polygon (Any Polygon) drawing

Issues triangle drawing command with the various settings to perform MVP transformation, geometry and rendering attributes set.

Polygon drawing is a function for drawing any closed polygons including concave shapes comprising vertices. (However, this does not support polygons having intersected sides.)

The polygon drawing flag buffer requires “drawing frame (1 pixel = 1 bit) + X resolution (1 pixel = 1 bit) × 2” bits of memory on the graphics memory as the work area for drawing. It is necessary to set the buffer address (PFBR register) to leave that amount free for X resolution. Refer to the product specifications for details.

Note that it is not possible to delete hidden plane erasing, texture mapping or gouraud shading that use the Z buffer.

<Program example>

- Format: Floating point
- Vertex: (x0,y0) to (x7,y7)
- α Blend

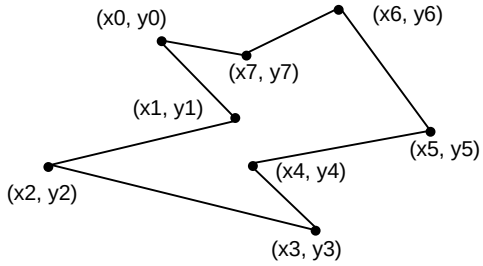


Fig. Drawing Example

```
/* When using driver function */
        GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
        GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_FLAT);
        GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_DISABLE);
        GdcSetAttrSurf(GDC_BLEND_MODE, GDC_BLEND_ALPHA);
        GdcSetAttrSurf(GDC_ TEXTURE_SELECT, GDC_SELECT_PLAIN);
        GdcSetAlpha(0x80);
        GdcGeoPrimType (GDC_POLYGON);
        GdcGeoDrawVertex2Df( x0, y0);
        GdcGeoDrawVertex2Df( x1, y1);
        GdcGeoDrawVertex2Df( x2, y2);
        GdcGeoDrawVertex2Df( x2, y2);
        GdcGeoDrawVertex2Df( x3, y3);
        GdcGeoDrawVertex2Df( x4, y4);
        GdcGeoDrawVertex2Df( x5, y5);
        GdcGeoDrawVertex2Df( x6, y6);
        GdcGeoDrawVertex2Df( x7, y7);
        GdcGeo PrimEnd();
```

```
/* When using display list */
        0xF1012040      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
        0x00000000              /* Vertex parameter setting */
        0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
        0x00000680              /* Triangle drawing attributes setting */
        0xF1010122      /* Type: Set Register, data count: 1, address: 488 =>ALF */
        0x00000080              /* α coefficient setting */
        0x21020000      /*Type:G_Begin Command:Polygon*/
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x0 */
        0xXXXXXXXX              /*float y0 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x1 */
        0xXXXXXXXX              /*float y1 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x2 */
        0xXXXXXXXX      /*float y2 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x3 */
        0xXXXXXXXX              /*float y3 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x4 */
        0xXXXXXXXX               /*float y4 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x5 */
        0xXXXXXXXX              /*float y5 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x6 */
        0xXXXXXXXX              /*float y6 */
        0x30000000      /*Type:G_Vertex */
        0xXXXXXXXX              /*float x7 */
        0xXXXXXXXX              /*float y7 */
        0x23000000      /*Type:G_End */
```

4.9 Drawing Using the Rendering Commands

It is possible to directly specify the drawing device coordinates to draw as the Cremson upper compatible LSI.

It is necessary to set as described in ***Sections 4.8.2.2*** and ***4.7*** for proper handling of the various drawing attributes and Z buffer.

4.9.1 Point drawing

<Drawing>

- Point color: White
- Point drawing coordinates: (X, Y) = (100, 150)
- Figure drawing attributes: Alpha blending (α coefficient 0x80)

```
/*-------------------------------------------------------------------------------------------------------
        Point drawing program
-------------------------------------------------------------------------------------------------------*/
/*--------------------- Driver function use example ------------------------------------------*/
GdcSetAttrLine ( GDC_BLEND_MODE, GDC_BLEND_ALPHA );    /* Blend mode setting */
GdcSetAlpha ( 0x80 );                                  /* Alpha blend coefficient setting */
GdcPrimType ( GDC_POINTS );                            /* Drawing primitive type setting */
GdcColor ( 0x7FFF );                                   /* 16 bit point color setting */
GdcDrawVertex2D ( 0x00640000, 0x00960000 );            /* Vertex coordinates setting */
GdcPrimEnd ( );                                        /* Point drawing end */

/*---------------------- Display list example ------------------------------------------------------*/
F1010109        /* Type: Set Register, data count: 1, address: 109 =>424 */
00000680                /* MDR1 register blend mode setting */
F1010122        /* Type: Set Register, data count: 1, address: 122 =>488 */
00000080                /* ALF register alpha blending coefficient setting */
F1010120        /* Type: Set Register, data count: 1, address: 120 =>480 */
00007FFF                /* FC register foreground color setting */
00000000        /*Type:  DrawPixel, Command: Pixel */
00648000                /* X coordinates setting */
00968000              /* Y coordinates setting */
```

4.9.2 Line drawing

<Drawing>

- Line color: White
- Line drawing coordinates: (X, Y, Z) = (20, 16, 10) , (150, 250, 160)
- Z value comparison method: Draw if Z value is smaller than Z buffer.
- Figure drawing attributes: Broken line, line width 15 pixels, anti-alias processing

```
/*-------------------------------------------------------------------------------------------------------
        3D Line drawing program example
-------------------------------------------------------------------------------------------------------*/
/*--------------------- Driver function use example ----------------------------------------*/
GdcSetAttrLine ( GDC_DEPTH_TEST, GDC_ENABLE );              /* Z value comparison mode setting */
GdcSetAttrLine ( GDC_DEPTH_FUNC, GDC_DEPTH_LESS );          /* Z comparison equation setting */
GdcSetAttrLine ( GDC_DEPTH_WRITE_MASK, GDC_DISABLE );       /* Write Z value to Z buffer */
GdcSetAttrLine ( GDC_BLEND_MODE, GDC_BLEND_COPY );          /* Set drawing blend mode to logic operation */
GdcSetAttrLine ( GDC_BROKEN_LINE, GDC_ENABLE );             /* Draw broken line */
GdcSetAttrLine ( GDC_LINE_WIDTH, GDC_LINE_WIDTH_15 );       /* Set line width to 15 pixels */
GdcSetAttrLine ( GDC_ANTI_ALIAS, GDC_ENABLE );              /* Anti-alias process setting */
GdcSetLinePattern ( 0xFFFF0000 );                           /* Broken line pattern setting */
GdcPrimType ( GDC_LINES );                                  /* Drawing primitive type setting */
GdcColor ( GDC_RED);                                        /* 16 bit line color setting */
GdcDrawVertex3D ( 0x000A0000, 0x000A0000, 0x000A );         /* Vertex coordinates setting */
GdcDrawVertex3D ( 0x00F60000, 0x00FA0000, 0x00A0 );         /* Vertex coordinates setting */
/*---------------------- Display list example ------------------------------------------------------*/
F1010109        /* Type: Set Register, data count: 1, address: 109 =>424 */
0E080014            /* MDR1 register   Line drawing mode setting */
F1010123        /* Type: Set Register, data count: 1, address: 123 =>48C */
FFFF0000            /* BLP register Broken line pattern setting */
F1010120        /* Type: Set Register, data count: 1, address: 120 =>480 */
00007C00            /* FC register drawing color setting */
F1020055        /* Type: Set Register, data count: 1, address: 55 =>154 */
00050000            /* LZs   Line position drawing starting point Z coordinates setting */   } Z component settings
00005000/*          LZde  Z incline value setting */                                          }
022D0000        /*Type: DrawLine,Cmd: AntiYvectorBlpClear*/                                  }
00F00000            /* LPN   Principal axis direction pixel count */                        }
00030000            /* LXs   Line position drawing starting point X coordinates setting */  }
0000FBBB            /* Lxde   X incline value setting */                                    } Line drawing
000A0000            /* LYs   Line position drawing starting point Y coordinates setting */  }
00010000            /* LYde   Y incline value setting */                                    }
```

4.9.3 Triangle drawing

< Drawing >

- Triangle drawing coordinates: (X, Y, Z) = (100, 100, 255) , (150, 250, 0) , (100, 200, 255)
- Figure drawing attribute: Texture mapping, GURO shading
- Texture attributes: Perspective collect, bi-linear sampling, texture wrap repeat, texture blend module
- Texture pattern: Pattern stored in COL16 type pointer movie01
- Texture size: 128 × 128 pixels
- Texture memory: Loaded to the graphics memory address 0x500000.

```
/*-----------------------------------------------------------------------------------------------------
Triangle drawing program using texture mapping
(With GURO shading)
-----------------------------------------------------------------------------------------------------*/
/*---------------------- Driver function use example -------------------------------------------*/
GdcSetAttrSurf ( GDC_SHADE_MODE, GDC_SHADE_SMOOTH );  /* Shading mode setting
GdcSetAttrSurf ( GDC_DEPTH_TEST, GDC_ENABLE );              /* Z value comparison mode setting */
GdcSetAttrSurf ( GDC_DEPTH_FUNC, GDC_DEPTH_LESS );          /* Z value comparison equation setting */
GdcSetAttrSurf ( GDC_DEPTH_WRITE_MASK, GDC_DISABLE );     /* Z value write enable mode setting */
GdcSetAttrSurf ( GDC_BLEND_MODE, GDC_BLEND_COPY );          /* Blend mode setting */
GdcSetAttrSurf ( GDC_TEXTURE_SELECT, GDC_SELECT_TEXTURE ); /* Texture mode setting */
GdcSetAttrTexture ( GDC_TEXTURE_PERSPECTIVE, GDC_ENABLE );
                                                  /* Texture coordinates correct mode setting */
GdcSetAttrTexture ( GDC_TEXTURE_FILTER, GDC_TEXTURE_BILINEAR );
                                                  /* Texture coordinates correct mode setting */
GdcSetAttrTexture ( GDC_TEXTURE_WRAP_S, GDC_TEXTURE_REPEAT );
                                                  /* Texture S coordinates wrap method setting */
GdcSetAttrTexture ( GDC_TEXTURE_WRAP_T, GDC_TEXTURE_REPEAT );
                                                  /* Texture T coordinates wrap method setting */
GdcSetAttrTexture ( GDC_TEXTURE_BLEND, GDC_TEXTURE_MODULATE );
                                                  /* Texture color and polygon color blend setting */
GdcTextureLoadExt ( (128*128), movie01, 0x500000 );  /* Load texture pattern to graphics memory */
GdcTextureMemoryMode ( GDC_TEX_MEM_MODE_EXT );
                                    /* Set texture pattern reference destination memory to internal or external */
GdcTextureDimension ( 0x500000, 128, 128);   /* Texture size and address to reference texture setting */
GdcPrimType ( GDC_TRIANGLES );                                  /* Drawing primitive setting */
GdcColor ( GDC_GRAY );                                          /* Vertex color setting */
GdcTexCoord3DNf ( 0.0f, 0.0f, 0.5f );                           /* Vertex texture coordinates setting */
GdcDrawVertex3D ( 0x00640000, 0x00640000, 0xFF );               /* Vertex coordinates setting */
GdcTexCoord3DNf ( 0.0f, 1.0f, 0.5f );                           /* Vertex texture coordinates setting */
GdcDrawVertex3D ( 0x00640000, 0x00C80000, 0xFF);                /* Vertex coordinates setting */
GdcColor ( GDC_YELLOW );                                        /* Vertex color setting */
GdcTexCoord3DNf ( 1.0f, 0.5f, 1.0f );                           /* Vertex texture coordinates setting */
GdcDrawVertex3D ( 0x00FA0000, 0x00960000, 0x00);                /* Vertex coordinates setting */
GdcPrimEnd ( );                                                 /* Drawing primitive end */
```

```
/*------------------ Display list example ------------------*/
F101010A          /* Type: Set Register, data count: 1, address: 10A =>428 */
20000615                /* MDR2 register   Window drawing mode setting */
F101010B          /* Type: Set Register, data count: 1, address: 10B =>42C */
00010528                /* MDR3 register   Texture mapping process mode setting */
F1010113          /* Type: Set Register, data count: 1, address: 113 =>44C */
00500000                /* TBR Register   Texture memory base setting */
F101010C (Note) /* Type: Set Register, data count: 1, address: 10 =>430 */
00000700                /* MDR4 register   BLT mode setting */
F1010110          /* Type: Set Register, data count: 1, address: 110 =>440 */
00500000                /* FBR register   Drawing frame base address setting */
F1010111          /* Type: Set Register, data count: 1, address: 111 =>440 */
00000080                /* XRES Register   Drawing window X axis direction resolution setting */
0B422002          /* Type: Draw Bit map P, Cmd: BltDram, data count: 8194 */
00000000          /* RYs, RXs, Drawing starting point coordinates setting of Y, X */
00800080          /* RsizeY, RsizeX Y, X direction drawing size setting */
3E533E53          /* Texture pattern */
    ⋮                                                    8192 texture patterns
3E533E53                /* Texture pattern */
F101010C (Note) /* Type: Set Register, set data count: 1, address: 10C =>430 */
00000000                /* MDR4 register   BLT mode setting */
F1010110          /* Type: Set Register, data count: 1, address: 110 =>440 */
00000000                /* FBR register   Drawing frame base address setting */
F1010111          /* Type: Set Register, data count: 1, address: 111 =>440 */
00000280                /* XRES Register   Drawing window X axis direction resolution setting */
F1010120          /* Type: Set Register, data count: 1, address: 120 =>480 */
00004210                /* FC register   Foreground color for drawing setting */
F1010119          /* Type: Set Register, data count: 1, address: 119 =>464 */
00800080                /* TXS register   Texture size setting */
F1030020          /* Type: Set Register, data count: 3, address: 20 =>80 */
007F8000                /*Zs   Long edge starting Z coordinates relating to Ys setting */
FFFF2667                /* dZdx   Z add value of the X axis direction setting */
00000000                /* dZdx   Z add value of the long edge direction */
F1090010          /* Type: Set Register, data count: 9, address: 10 =>40 */
00800000                /* Rs   Long edge starting color value R relating to Ys setting */
0000CCCC                /* dRdx   R add value of the X axis direction setting */
00000000                /* dRdx   R add value of the long edge direction setting */
00800000                /* Gs    Long edge starting color value G relating to Ys setting */
0000CCCC                /* dGdx   G add value of the X axis direction setting */
00000000                /* dGdy   G add value of the long edge direction setting */
00800000                /* Bs   Long edge starting color value B relating to Ys setting */
FFFF258C                /* dBdx   B add value of the X axis direction setting */
00000000          /* dBdy   B add value of the long edge direction setting */
```

Recovery of original setting value

Z component settings

Color component setting

```
F1090030        /* Type: Set Register, data count: 9, address: 30 =>C0 */
00000000            /* Ss   Long edge starting S texture coordinates relating to Ys setting */
0000DA74            /* dSdx   S add value of the X axis direction setting */
00000000            /* dSdx   S add value of the long edge direction */
00000000            /* Ts   Long edge starting T texture coordinates relating to Ys setting */
0000369D            /* dTdx   T add value of the X axis direction setting */
0000A3D7            /* dTdy   T add value of the long edge direction */
00800000            /* Qs   Long edge starting depth correction value setting relating to Ys */
0000DA74            /* dQdx   W add value of the X axis direction setting */
00000000            /* dQdy   Q add value of the long edge direction */
05600000        /* Type: DrawTrap,Cmd: TrapRight*/
00640000            /* Ys   Long edge starting Y coordinates setting */
00640000            /* Xs   Long edge starting X coordinates relating t Ys setting */
00000000            /* DXdy   X add value of the long edge direction */
0063FFFF            /* XUs   Upper edge starting X coordinates setting */
00030000            /* DXUdy   X add value of the upper edge direction */
00F9FFFE            /* XLs   Lower edge starting X coordinates setting */
FFFD0001            /* DXLdy   X add value of the lower edge direction */
00320000            /* USN   Sets upper triangle span count */
00320000            /* LSN   Sets lower triangle span count */
```

Texture component setting

Draw triangle

5. CREATING APPLICATION PROGRAMS

This section describes program example that makes the best use of Orchid's performance and functions.

Review this section when creating applications.

5.1 Executing Only the Setup Process Using Orchid

The following example shows a program example that executes the setup with Orchid by passing the drawing device coordinates directly to Orchid.

Set the parameters as shown below. Set the rendering attributes to their appropriate values.

Note that view clip settings are off in the example below, but if the object coordinates applied to Orchid are outside of the draw frame excessively, set the X, Y and Z view clip frame properly. See ***Sections 4.4*** and ***4.6.2.4***.

- Queue: Increment Queue

$$\begin{pmatrix} 1.0f & 0.0f & 0.0f & 0.0f \\ 0.0f & 1.0f & 0.0f & 0.0f \\ 0.0f & 0.0f & 1.0f & 0.0f \\ 0.0f & 0.0f & 0.0f & 1.0f \end{pmatrix}$$

- View Port: Offset is 0, Scale is 1 magnification.
- View Clip: Off

  min = -3.40282347e+38(0xff7fffff) (Single precision Float minimum value)

  max = 3.40282347e+38(0x7f7fffff) (Single precision Float maximum value)

  wmin = 1.1754944e-38(0x00800000) (Single precision Float positive minimum value)
- GMDR0 register: FrustumBit (Bit0) = 0 (When using the driver, it is automatically set by setting the increment queue.

<Program example>

- Draws triangles (100.0f,100.0f,100.0f), (100.0f,200.0f,100.0f), (200.0f,200.0f,100.0f) with the color parameters (1.0f, 0, 0), (0, 1.0f, 0), (0, 0, 1.0f).

```
/* When using driver function */
float mat[] = {1,0,0,0,  0,1,0,0,  0,0,1,0,  0,0,0,1}; // increment
float min = -3.40282347e+38; float max = 3.40282347e+38; float wmin = 1.1754944e-38;
GdcGeoLoadMatrixf(mat);
GdcGeoNdcDcViewportCoeff( 1.0f, 0, 1.0f, 0);
GdcGeoNdcDcDepthCoeff( 1.0f, 0);
GdcGeoViewVolumeXYClip( min, max, min, max);
GdcGeoViewVolumeZClip( min, max);
GdcGeoViewVolumeWminClip( wmin);
GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_ENABLE);
GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_ENABLE);
GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_ENABLE);
GdcSetAttrSurf(GDC_DEPTH_FUNC, GDC_DEPTH_LEQUAL);
GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_SMOOTH);
GdcGeoPrimType (GDC_TRIANGLES);
GdcVertexColor3f(1.0f, 0.0f, 0.0f);
GdcGeoDrawVertex3Df( 100.0f, 100.0f, 100.0f);
GdcVertexColor3f(0.0f, 1.0f, 0.0f);
GdcGeoDrawVertex3Df( 100.0f, 200.0f, 100.0f);
GdcVertexColor3f(0.0f, 0.0f, 1.0f);
GdcGeoDrawVertex3Df( 200.0f, 200.0f, 100.0f);
GdcGeo PrimEnd

/* When using display list */
0x43000000          /* Type:G_LoadMatrix */
0x3F800000                  /*Matrix_a0 */
0x00000000                  /*Matrix_a1 */
0x00000000                  /*Matrix_a2 */
0x00000000                  /*Matrix_a3 */
0x00000000                  /*Matrix_b0 */
0x3F800000                  /*Matrix_b1 */
0x00000000                  /*Matrix_b2 */
0x00000000                  /*Matrix_b3 */
0x00000000                  /*Matrix_c0 */
0x00000000                  /*Matrix_c1 */
0x3F800000                  /*Matrix_c2 */
0x00000000                  /*Matrix_c3 */
0x00000000                  /*Matrix_d0 */
0x00000000                  /*Matrix_d1 */
0x00000000                  /*Matrix_d2 */
0x3F800000                  /*Matrix_d3 */
0x41000000          /* Type:G_Viewport */
0x3F800000                  /*X_Scaling */
0x00000000                  /*X_Offset */
0x3F800000                  /*Y_Scaling */
0x00000000                  /*Y_Offset */
0x42000000          /* Type:G_DepthRange */
0x3F800000                  /*Z_Scaling */
0x00000000                  /*Z_Offset */
```

Queue setting

```
0x44000000      /* Type:G_ViewVolumeXYClip */
0xFF7FFFFF          /*XMIN */
0x7F7FFFFF          /*XMAX */
0xFF7FFFFF          /*YMIN */
0x7F7FFFFF          /*YMAX */
0x45000000      /* Type:G_ViewVolumeZClip */
0xFF7FFFFF          /*ZMIN */
0x7F7FFFFF          /*ZMAX */
0x46000000      /* Type:G_ViewVolumeWClip */
0x00800000          /*WMIN */
0xF1012010      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
0x00000006          /* Peak Parameter Setting */
0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
0x0000061D          /* Triangle rendering mode setting */
0x21030000      /*Type:G_Begin Command:Triangles*/
0x30000000      /*Type:G_Vertex */
0x42C80000          /*float x0 */
0x42C80000          /*float y0 */
0x42C80000          /*float z0 */
0x3F800000          /*float r0 */
0x00000000          /*float g0 */
0x00000000          /*float b0 */
0x30000000      /*Type:G_Vertex */
0x42C80000          /*float x1 */
0x43480000          /*float y1 */
0x42C80000          /*float z1 */
0x00000000          /*float r1 */
0x3F800000          /*float g1 */
0x00000000          /*float b1 */
0x30000000      /*Type:G_Vertex */
0x43480000              /*float x2 */
0x43480000              /*float y2 */
0x42C80000              /*float z2 */
0x00000000              /*float r2 */
0x00000000              /*float g2 */
0x3F800000              /*float b2 */
0x23000000      /*Type:G_End */
```

5.2 Executing 2D Transformation Using Orchid

The following shows a program that transforms from object coordinates to the 2D coordinates to execute drawing.

When the GMDR0 register Frustum mode bit is “0” and the Z data enable bit is “0”, it ignores Z and calculates as the 3×2 queue but division using Wcc is not performed.

The following shows a description of the method of calculation. Refer to ***Sections 7.1.2*** and ***7.1.3*** in the specifications for Orchid for details regarding 3D conversion calculations.

To transform 2D coordinates with Orchid, set the queue to load to Orchid as the following.

Calculations using Orchid:

Queue to load to Orchid

$$\begin{pmatrix} Xcc \\ Ycc \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} MA0 & MA1 & 0 & MA2 \\ MB0 & MB1 & 0 & MB2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Xoc \\ Yoc \\ 0 \\ 1 \end{pmatrix}$$

Xcc = MA0*Xoc + MA1*Yoc + MC0

Ycc = MB0*Xoc + MB1*Yoc + MC1

<Program Example>

Draws the triangle with object coordinates (-1.0f, 1.0f), (-1.0f, -1.0f) and (1.0f, -1.0f) by view port center (320, 240), view port scale (50, 50), view clop off and moving X by 3 with transformation queue.

```
/* When using driver function */
    float mat[] = {1,03.0,0,  0,1,0,0,  0,0,1,0,  0,0,0,1}; // increment
    float min = -3.40282347e+38; float max = 3.40282347e+38; float wmin = 1.17549435e-38;
    GdcGeoLoadMatrixf(mat);
    GdcGeoNdcDcViewportCoeff( 50.0f, 320.0f, 50.0f, 240.0f);
    GdcGeoViewVolumeXYClip( min, max, min, max);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_DISABLE);
    GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_DISABLE);
    GdcGeoPrimType (GDC_TRIANGLES);
    GdcGeoDrawVertex2Df( -1.0f, 1.0f);
    GdcGeoDrawVertex2Df( -1.0f, -1.0f);
    GdcGeoDrawVertex2Df( 1.0f, -1.0f);
    GdcGeo PrimEnd();

/* When using display list */
    0x43000000    /* Type:G_LoadMatrix */
    0x3F800000        /*Matrix_a0 */
    0x00000000        /*Matrix_a1 */
    0x00000000        /*Matrix_a2 */
    0x40400000        /*Matrix_a3 */
    0x00000000        /*Matrix_b0 */
    0x3F800000        /*Matrix_b1 */
    0x00000000        /*Matrix_b2 */
    0x00000000        /*Matrix_b3 */
    0x00000000        /*Matrix_c0 */
    0x00000000        /*Matrix_c1 */
    0x3F800000        /*Matrix_c2 */
    0x00000000        /*Matrix_c3 */
    0x00000000        /*Matrix_d0 */
    0x00000000        /*Matrix_d1 */
    0x00000000        /*Matrix_d2 */
    0x3F800000        /*Matrix_d3 */
    0x41000000    /* Type:G_Viewport */
    0x42480000        /*X_Scaling */
    0x43A00000        /*X_Offset */
    0x42480000        /*Y_Scaling */
    0x43700000        /*Y_Offset */
    0x44000000    /* Type:G_ViewVolumeXYClip */
    0xFF7FFFFF        /*XMIN */
    0x7F7FFFFF        /*XMAX */
    0xFF7FFFFF        /*YMIN */
    0x7F7FFFFF        /*YMAX */
    0xF1012040    /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
    0x00000000        /* Vertex parameter setting */
    0xF101010A    /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
    0x00000600        /* Triangle rendering mode setting */
    0x21030000    /*Type:G_Begin Command:Triangles*/
    0x30000000    /*Type:G_Vertex */
    0xBF800000        /*float x0 */
    0x3F800000        /*float y0 */
    0x30000000    /*Type:G_Vertex */
    0xBF800000        /*float x1 */
    0xBF800000        /*float y1 */
```

Queue setting

```
0x30000000      /*Type:G_Vertex */
0x3F800000          /*float x2 */
0xBF800000          /*float y2 */
0x23000000      /*Type:G_End */
```

5.3 Drawing a Bird's-eyes View Image Using Orchid

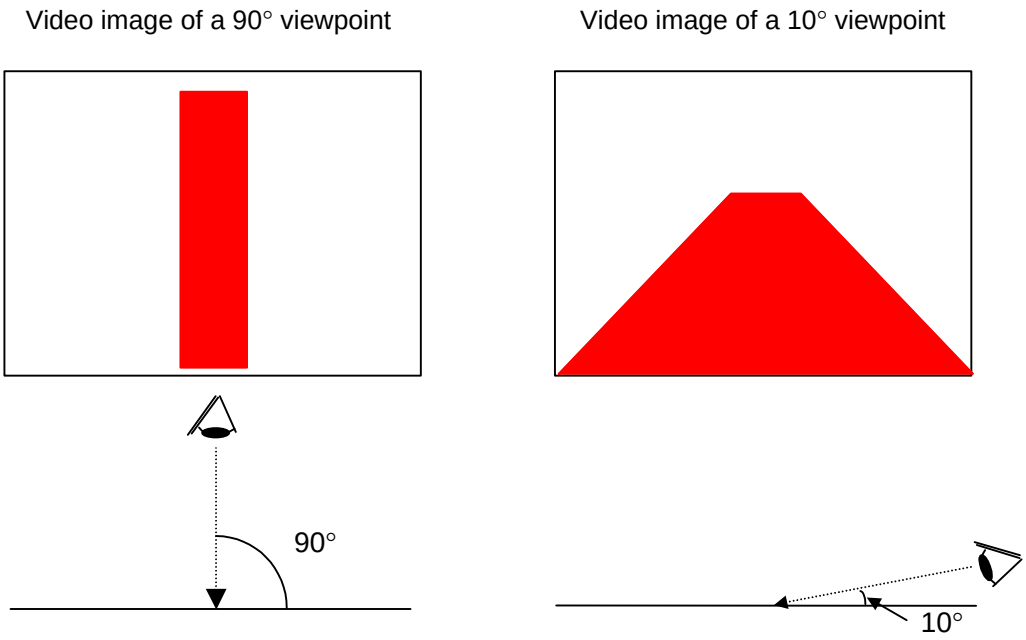
The following shows an example in which the queue operation is performed by 4 × 4 and drawing is executed without Z buffer.

This method is effective to incline a 2D map and draw a bird's-eye view of the map.

When the GMDR0 register Frustum mode bit is "1" and the Z data enable bit is "0", the geometry engine calculates the coordinates transformation of the 4 × 4 queue as Z = 0; with the rendering engine, it issues a command with no Z buffer to draw.

<Program example>

- Transforms queues and draws by setting object coordinates (-1.0f, 5.0f), (-1.0f, -5.0f), (1.0f, 5.0f) and (1.0f, -5.0f) with triangle strip to view port center (320, 240), view port scale (200.0f, 200.0f), view clip xmin = -2.0f, xmax = 2.0f, ymin = -1.25f, ymax = 1.25f, zmin = -1.0f, zmax = 1.0f, and wmin = 0.000000001f.



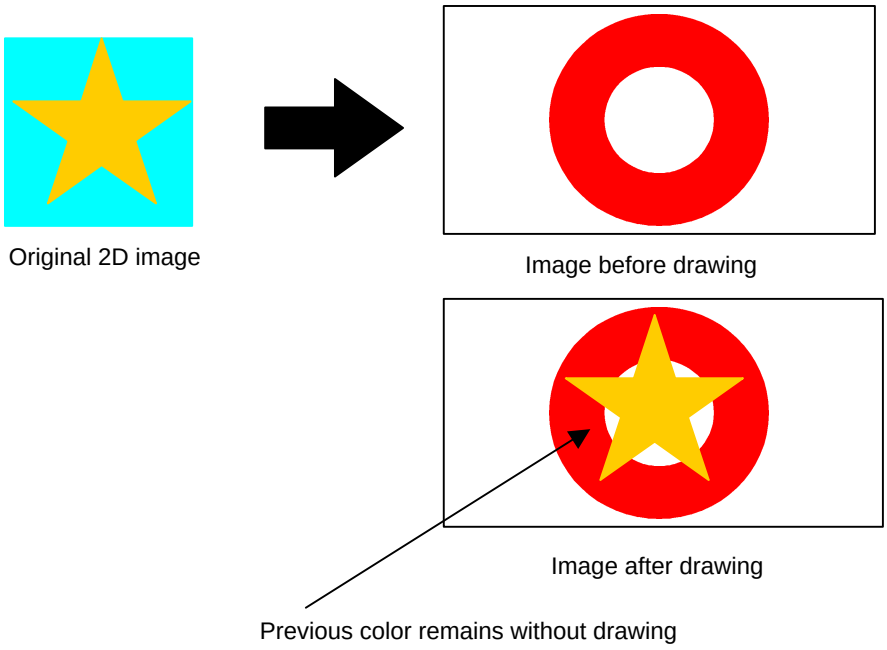
```
/* When using driver function */
        /* Matrix of incline of 10° */
        float mat10[] = { -1.0f, 0.f, 0.f, 0.f,  0.f,-0.173648f,0.f, 0.984808f,
                0.f, -0.984808f, 0.f, -0.173648f,  0.f, 0.f, 0.1f, 5.0f};
        /* Matrix of incline of 90° */
        float mat90[] = { -1.0f, 0.f, 0.f, 0.f,  0.f,-1.0f,0.f, 0. f,
                0.f, 0.f, 0.f, -1.0f,  0.f, 0.f, 0.1f, 5.0f};
        GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT, GDC_GEO_FLOAT_INPUT);
        GdcGeoSetAttrSurf(GDC_GEO_FACE_CULL, GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_VTX_Z, GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_VTX_COL, GDC_DISABLE);
        GdcGeoSetAttrMisc(GDC_GEO_VTX_ST, GDC_DISABLE);
        GdcGeoNdcDcViewportCoeff(200.0f, 320.0f, 200.0f, 240.0f);
        GdcGeoNdcDcDepthCoeff(100.f, 32768.0f);
        GdcGeoViewVolumeXYClipf(-2.0f, 2.0f, -1.25f, 1.25f);
        GdcGeoViewVolumeZClipf(-1.0f, 1.0f);
        GdcGeoViewVolumeWminClipf(wmin);
        GdcDrawClipFrame(0, 0, 640, 480);
        GdcSetAttrMisc(GDC_CLIP, (GDC_CLIP_X_ON|GDC_CLIP_Y_ON));
        GdcSetAttrSurf(GDC_DEPTH_TEST, GDC_DISABLE);
        GdcSetAttrSurf(GDC_SHADE_MODE, GDC_SHADE_FLAT);
        GdcSetAttrSurf(GDC_TEXTURE_SELECT,GDC_SELECT_PLAIN);
        GdcGeoLoadMatrixf(fMat10);
        GdcColor(0x7c00);
        GdcGeoPrimType(GDC_TRIANGLE_STRIP);
        GdcGeoDrawVertex2Df(-1.0f, 5.0f);
        GdcGeoDrawVertex2Df(-1.0f,-5.0f);
        GdcGeoDrawVertex2Df( 1.0f, 5.0f);
        GdcGeoDrawVertex2Df( 1.0f,-5.0f);
        GdcGeoPrimEnd();

/* When using display list */
        0x41000000      /* Type:G_Viewport */
        0x42480000              /*X_Scaling */
        0x43A00000              /*X_Offset */
        0x42480000              /*Y_Scaling */
        0x43700000              /*Y_Offset */
        0x41000000      /* Type:G_DepthRange */
        0x42C80000              /*Z_Scaling */
        0x47000000              /*Z_Offset */
        0x44000000      /* Type:G_ViewVolumeXYClip */
        0xC0000000              /*XMIN */
        0x40000000              /*XMAX */
        0xBFA00000              /*YMIN */
        0x3FA0000               /*YMAX */
        0x45000000      /*G_ViewClipZ*/
        0xBF800000              /*ZMIN */
        0x3F800000              /*ZMAX */
        0x46000000      /*G_ViewClipW*/
        0x3089705F              /*WMIN */
        0xF1040115      /* Type: Set Register, data count: 4, address: 454 =>CXMIN */
        0x00000000              /* Rendering clip XMIN setting */
        0x00000280              /* Rendering clip XMAX setting */
        0x00000000              /* Rendering clip YMIN setting */
```

```
0x000001E0          /* Rendering clip YMAX setting */
0xF1010108      /* Type: Set Register, data count: 1, address: 420 =>MDR0 */
0x00008300          /* Rendering clip on setting */
0xF1012010      /* Type: Set Register, data count: 1, address: 8040 =>GMDR0 */
0x00000001      /* Vertex parameter setting */
0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
0x00000600      /* Triangle rendering mode setting */
0x43000000      /* Type:G_LoadMatrix */
0xBF800000          /*Matrix_a0 */
0x00000000          /*Matrix_a1 */
0x00000000          /*Matrix_a2 */
0x00000000          /*Matrix_a3 */
0x00000000          /*Matrix_b0 */
0xBE31D0C8          /*Matrix_b1 */
0xBF7C1C61          /*Matrix_b2 */
0x00000000          /*Matrix_b3 */
0x00000000          /*Matrix_c0 */
0x00000000          /*Matrix_c1 */
0x00000000          /*Matrix_c2 */
0x3DCCCCCD          /*Matrix_c3 */
0x00000000          /*Matrix_d0 */
0x3F7C1C61          /*Matrix_d1 */
0xBE31D0C8          /*Matrix_d2 */
0x40A00000          /*Matrix_d3 */
0xF1010120      /* Type: Set Register, data count: 1, address: 480 =>FC */
0x00007C00          /* Foreground color setting */
0x21070000      /*Type:G_Begin Command:Triangle_Strip*/
0x30000000      /*Type:G_Vertex */
0xBF800000          /*float x0 */
0xC0A00000          /*float y0 */
0x30000000      /*Type:G_Vertex */
0x3F800000          /*float x1 */
0x40A00000          /*float y1 */
0x30000000      /*Type:G_Vertex */
0x3F800000          /*float x2 */
0xC0A00000          /*float y2 */
0x23000000      /*Type:G_End */
```

5.4 Drawing Sprite Image Using Orchid

The following describes how to draw sprite images, which are image expression techniques often used in amusement systems, like games, using Orchid.

The drawing of sprite images means that only the desired color is drawn without drawing a color of the 2D image.



There are two drawing methods for sprite images.

1. Drawing using BLT function
2. Drawing using texture mapping function

5.4.1 Drawing using BLT function

This is used to draw the original 2D image without rotating or zooming in or out, without drawing the background color of the original data to draw the desired image.

See ***Section 4.6.2.2*** for details regarding how to draw.

5.4.2 Drawing using texture mapping function

This is used to draw the original 2D image by rotating or zooming in or out, without drawing the background color of the original data to draw the desired image. Refer to ***8.5.2*** in the ***MB86291 <Orchid> Specifications*** for details regarding the “stencil” method.

Note that when the size of the original 2D image exceeds 64 × 64 pixels, arrange to the size of the sprite image to draw.

- Draws referring to the texture data on the graphics memory as it is.
- Separates sprite images and loads the texture data on the graphics memory to the ***Orchid*** internal TexRAM. While referencing the internal, TexRam the application controls the two drawing methods to allow high speed drawing.

See “XXXX” and “XXXX” for details regarding drawing performance.

5.4.2.1 Drawing by referring to texture data on graphics memory

<Program example>

- 256 × 256 2D image has been stored on the graphics memory (0X12C000) on ***Orchid***.
  (The background color bit15 is “0”, other pixel color bit15 are “1”.)
- Drawing 256 × 256 sprites

```
/* When using driver function */
    float fMat[] = { 1.0f, 0.f, 0.f, 0.f,  0.f, 1.0f,0.f, 0. f,
                     0.f, 0.f, 1.0f, 0.0f,  0.f, 0.f, 0.f, 1.0f};
    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_ENABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
    GdcSetAttrSurf(GDC_TEXTURE_SELECT,GDC_SELECT_TEXTURE);
    GdcSetAttrSurf(GDC_DEPTH_TEST,GDC_DISABLE);
    GdcSetAttrSurf(GDC_DEPTH_FUNC,GDC_DEPTH_LEQUAL);
    GdcSetAttrSurf(GDC_SHADE_MODE,GDC_SHADE_FLAT);
    GdcSetAttrSurf(GDC_BLEND_MODE,GDC_BLEND_ALPHA);
    GdcSetAttrTexture(GDC_TEXTURE_PERSPECTIVE,GDC_DISABLE);
    GdcSetAttrTexture(GDC_TEXTURE_FILTER,GDC_TEXTURE_POINT);
    GdcSetAttrTexture(GDC_TEXTURE_WRAP_S,GDC_TEXTURE_CLAMP);
    GdcSetAttrTexture(GDC_TEXTURE_WRAP_T,GDC_TEXTURE_CLAMP);
    GdcSetAttrTexture(GDC_TEXTURE_BLEND,GDC_TEXTURE_STENCIL);
    GdcSetAttrTexture(GDC_TEXTURE_ALPHA,GDC_TEXTURE_ALPHA_STENCIL);
    GdcGeoLoadMatrixf(fMatP);
    GdcGeoNdcDcViewportCoeff(1.0f, 320.0f, 1.0f, 240.0f);
    GdcGeoViewVolumeXYClipf(min, max, min, max);
    GdcTextureMemoryMode(GDC_TEX_MEM_MODE_EXT);
    GdcTextureDimension(0x12C000,256,256);
    GdcGeoPrimType(GDC_TRIANGLE_FAN);
    GdcGeoTexCoord2DNf(0.0f,0.0f);
    GdcGeoDrawVertex2Df(-128.0f, -128.0f);
    GdcGeoTexCoord2DNf(0.0f,1.0f);
    GdcGeoDrawVertex2Df(-128.0f, 128.0f);
    GdcGeoTexCoord2DNf(1.0f, 1.0f);
    GdcGeoDrawVertex2Df(128.0f, 128.0f);
```

```
        GdcGeoTexCoord2DNf(1.0f,0.0f);
        GdcGeoDrawVertex2Df( 128.0f,  -128.0f);
        GdcGeoPrimEnd();

/* When using display list */
        0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
        0x20000698          /* Polygon drawing mode setting */
        0xF101010B      /* Type: Set Register, data count: 1, address: 42C =>MDR3 */
        0x00120000          /* Texture mode setting */
        0xF1012010      /* Type: Set Register, data count: 1, address: 40 =>GMDR0 */
        0x00000008          /* Geometry mode setting */
        0x43000000      /* Type:G_LoadMatrix */
        0x3F800000          /*Matrix_a0 */
        0x00000000          /*Matrix_a1 */
        0x00000000          /*Matrix_a2 */
        0x00000000          /*Matrix_a3 */
        0x00000000          /*Matrix_b0 */
        0x3F800000          /*Matrix_b1 */
        0x00000000          /*Matrix_b2 */
        0x00000000          /*Matrix_b3 */
        0x00000000          /*Matrix_c0 */
        0x00000000          /*Matrix_c1 */
        0x00000000          /*Matrix_c2 */
        0x3F800000          /*Matrix_c3 */
        0x00000000          /*Matrix_d0 */
        0x00000000          /*Matrix_d1 */
        0x00000000          /*Matrix_d2 */
        0x3F800000          /*Matrix_d3 */
        0x41000000        /* Type:G_Viewport */
        0x3F800000          /*X_Scaling */
        0x43A00000          /*X_Offset */
        0x3F800000          /*Y_Scaling */
        0x43700000          /*Y_Offset */
        0x44000000      /* Type:G_ViewVolumeXYClip */
        0xFF7FFFFF          /*XMIN */
        0x7F7FFFFF          /*XMAX */
        0xFF7FFFFF          /*YMIN */
        0x7F7FFFFF          /*YMAX */
        0xF1010113      /* Type: Set Register, data count: 1, address: 44C =>TBR */
        0x0012C000          /* Texture memory address setting */
        0xF1010119      /* Type: Set Register, data count: 1, address: 464 =>TXS */
        0x01000100          /* Texture size setting */
        0x21180000      /*Type:G_Begin Command:Triangle_Fan.Int*/
        0x30000000      /*Type:G_Vertex */
        0xC3000000          /*float x0 */
        0xC3000000          /*float y0 */
        0x00000000          /*float s0 */
        0x00000000          /*float t0 */
        0x30000000      /*Type:G_Vertex */
        0xC3000000          /*float x1 */
        0x43000000          /*float y1 */
        0x00000000          /*float s1 */
        0x3F800000      /*float t1 */
        0x30000000      /*Type:G_Vertex */
        0x43000000          /*float x2 */
```

```
0x43000000          /*float y2 */
0x3F800000          /*float s2 */
0x3F800000          /*float t2 */
0x30000000      /*Type:G_Vertex */
0x43000000          /*float x3 */
0xC3000000          /*float y3 */
0x3F800000          /*float s3 */
0x00000000          /*float t3 */
0x23000000      /*Type:G_End */
```

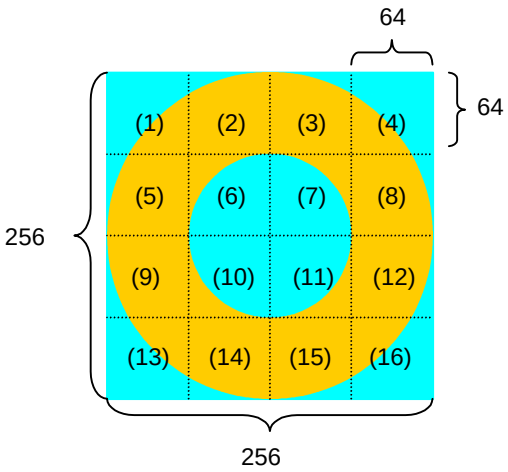
5.4.2.2 Drawing by separating sprite images and loading texture data to internal TexRAM

If the 2D image size exceeds 64 × 64, it divides the sprite image, loads the texture data on the graphics memory to the ***Orchid*** internal TexRAM and while referencing the internal TexRAM, it draws, which achieves high speed processing, based on the sprint image size.

Specifically, a 256 × 256 2D image is divided into 16 and into 16 for sprite images to be drawn.

1 is loaded to the internal TexRAM, draws A, 2 is loaded to the internal TexRAM, draws B, etc.

And 16 is loaded to the internal TexRAM, draws P.



Original 2D image

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Spring image to be drawn

<Program example>

- A 256 × 256 2D image is stored on the graphics memory (0X12C000) on ***Orchid***.
  (The background color bit15 is “0”, other pixel color bit 15 are “1”.)
- Draws 256 × 256 sprites
- Tex data is loaded to the internal TexRAM and draws by referring the internal TexRAM.

```
/* When using driver function */
    float fMat[] = { 1.0f, 0.f, 0.f, 0.f,  0.f, 1.0f,0.f, 0. f,
                     0.f, 0.f, 1.0f, 0.0f,  0.f, 0.f, 0.f, 1.0f};
    GdcGeoSetAttrMisc(GDC_GEO_VTX_ST,GDC_ENABLE);
     GdcGeoSetAttrMisc(GDC_GEO_VTX_COL,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_VTX_Z,GDC_DISABLE);
    GdcGeoSetAttrMisc(GDC_GEO_IN_FORMAT,GDC_GEO_FLOAT_INPUT);
    GdcSetAttrSurf(GDC_TEXTURE_SELECT,GDC_SELECT_TEXTURE);
    GdcSetAttrSurf(GDC_DEPTH_TEST,GDC_DISABLE);
    GdcSetAttrSurf(GDC_DEPTH_FUNC,GDC_DEPTH_LEQUAL);
    GdcSetAttrSurf(GDC_SHADE_MODE,GDC_SHADE_FLAT);
    GdcSetAttrSurf(GDC_BLEND_MODE,GDC_BLEND_ALPHA);
    GdcSetAttrTexture(GDC_TEXTURE_PERSPECTIVE,GDC_DISABLE);
    GdcSetAttrTexture(GDC_TEXTURE_FILTER,GDC_TEXTURE_POINT);
    GdcSetAttrTexture(GDC_TEXTURE_WRAP_S,GDC_TEXTURE_CLAMP);
    GdcSetAttrTexture(GDC_TEXTURE_WRAP_T,GDC_TEXTURE_CLAMP);
    GdcSetAttrTexture(GDC_TEXTURE_BLEND,GDC_TEXTURE_STENCIL);
    GdcSetAttrTexture(GDC_TEXTURE_ALPHA,GDC_TEXTURE_ALPHA_STENCIL);
    GdcGeoLoadMatrixf(fMatP);
```

```
        GdcGeoNdcDcViewportCoeff(1.0f, 320.0f, 1.0f, 240.0f);
        GdcGeoViewVolumeXYClipf(min, max, min, max);
        GdcTextureMemoryMode(GDC_TEX_MEM_MODE_INT);
        GdcTextureDimension(0,64,64);
        for(i=0;i<4;i++){
            for(j=0;j<4;j++){
                X0 = -128.0f + 64*i;
                Y0 = -128.0f + 64*j;
                GdcBltTexture(TEXADR,256,(64*i),(64*j),64,64, 0);
                GdcGeoPrimType(GDC_TRIANGLE_FAN);
                GdcGeoTexCoord2DNf(0.0f,0.0f);
                GdcGeoDrawVertex2Df(X0, Y0);
                GdcGeoTexCoord2DNf(0.0f,s);
                GdcGeoDrawVertex2Df(X0,(Y0+64.0f));
                GdcGeoTexCoord2DNf(s,s);
                GdcGeoDrawVertex2Df((X0+64.0f),(Y0+64.0f));
                GdcGeoTexCoord2DNf(s,0.0f);
                GdcGeoDrawVertex2Df( (X0+64.0f),  Y0);
                GdcGeoPrimEnd();
            }
        }

/* When using display list */
        0xF101010A      /* Type: Set Register, data count: 1, address: 428 =>MDR2 */
        0x20000698          /* Polygon drawing mode setting */
        0xF101010B      /* Type: Set Register, data count: 1, address: 42C =>MDR3 */
        0x00120000          /* Texture mode setting */
        0xF1012010      /* Type: Set Register, data count: 1, address: 40 =>GMDR0 */
        0x00000008          /* Geometry mode setting */
        0x43000000      /* Type:G_LoadMatrix */
        0x3F800000          /*Matrix_a0 */
        0x00000000          /*Matrix_a1 */
        0x00000000          /*Matrix_a2 */
        0x00000000          /*Matrix_a3 */
        0x00000000          /*Matrix_b0 */
        0x3F800000          /*Matrix_b1 */
        0x00000000          /*Matrix_b2 */
        0x00000000          /*Matrix_b3 */
        0x00000000          /*Matrix_c0 */
        0x00000000          /*Matrix_c1 */
        0x00000000          /*Matrix_c2 */
        0x3F800000          /*Matrix_c3 */
        0x00000000          /*Matrix_d0 */
        0x00000000          /*Matrix_d1 */
        0x00000000          /*Matrix_d2 */
        0x3F800000          /*Matrix_d3 */
        0x41000000      /* Type:G_Viewport */
        0x3F800000          /*X_Scaling */
        0x43A00000          /*X_Offset */
        0x3F800000          /*Y_Scaling */
        0x43700000          /*Y_Offset */
        0x44000000      /* Type:G_ViewVolumeXYClip */
        0xFF7FFFFF          /*XMIN */
        0x7F7FFFFF          /*XMAX */
```

```
0xFF7FFFFF          /*YMIN */
0x7F7FFFFF          /*YMAX */
0xF101010B      /* Type: Set Register, data count: 1, address: 42C =>MDR3 */
0x00120001      /* Texture mode setting */
0xF1010119      /* Type: Set Register, data count: 1, address: 464 =>TXS */
0x00400040      /* Texture size setting */
0xF101011B      /* Type: Set Register, data count: 1, address: 46C =>TOA */
0x00000000      /* Texture memory address setting */
0x13480000  /*Type:BltTexureP Command:LoadTexture*/
0x0012C000          /*SrcADDR*/
0x00000100          /*SrcStride*/
0x00000000          /*SrcRectYs SrcRectXs*/
0x00400040          /*BRsizeY BRsize*/
0x00000000          /*DestOffset*/
0x21180000      /*Type:G_Begin Command:Triangle_Fan.Int*/
0x30000000      /*Type:G_Vertex */
0xC3000000          /*float x0 */
0xC3000000          /*float y0 */
0x00000000          /*float s0 */
0x00000000          /*float t0 */
0x30000000      /*Type:G_Vertex */
0xC3000000          /*float x1 */
0xC2800000          /*float y1 */
0x00000000           /*float s1 */
0x3F800000          /*float t1 */
0x30000000      /*Type:G_Vertex */
0xC2800000          /*float x2 */
0xC2800000          /*float y2 */
0x3F800000          /*float s2 */
0x3F800000          /*float t2 */
0x30000000          /*Type:G_Vertex */
0xC2800000           /*float x3 */
0xC3000000           /*float y3 */
0x3F800000           /*float s3 */
0x00000000           /*float t3 */
0x23000000      /*Type:G_End *
```

Loading of 1 Tex Data

Drawing of A

/Repeated

5.4.2.3 Sprite image drawing performance

The following describes the drawing performance of ***5.4.2.1*** and ***5.4.2.2***.

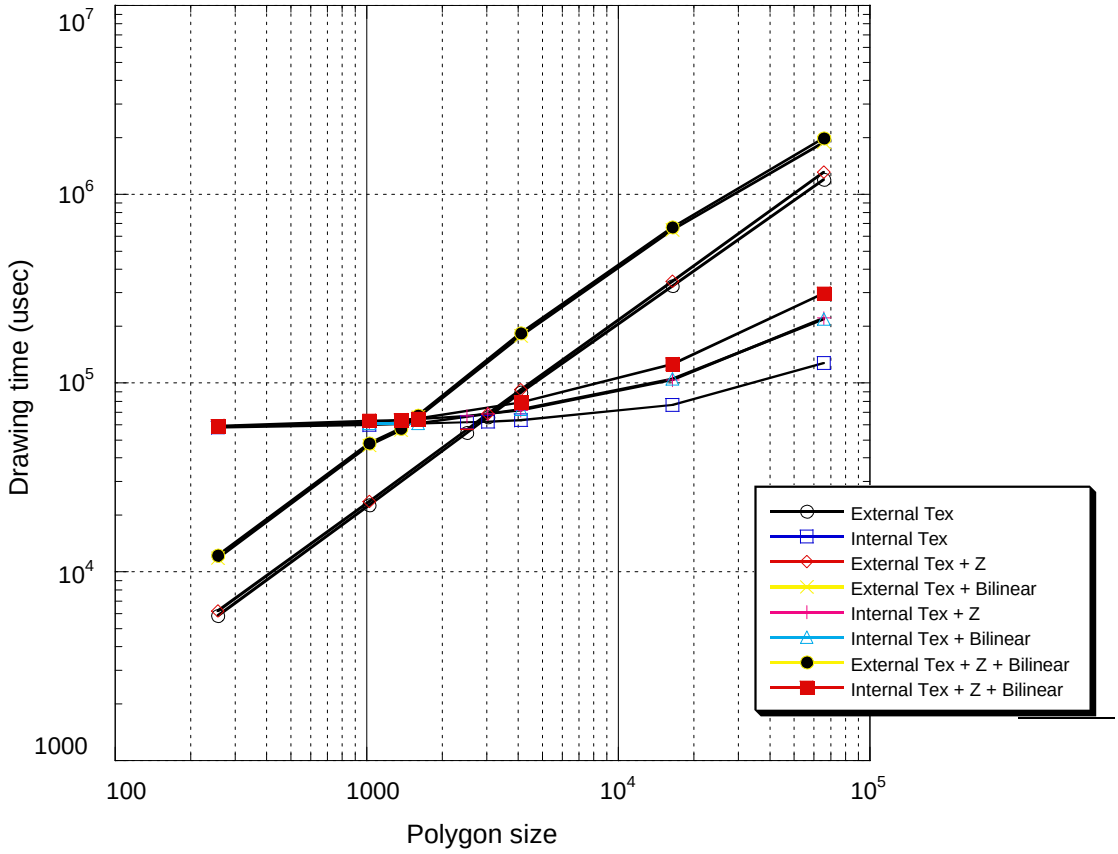
- Time all objects are drawn 100 times.
- Time units are usec.
- External Tex is the drawing time (equivalent to ***5.4.2.1***) when referring Tex data on the graphics memory to draw 2D image data.
- Internal Tex handles Tex data as a unit of 64 × 64, stores it in the internal TexRAM and divides polygons (For Tex size of 256 × 256, they are divided into 16, for 128 × 128, they are divided into 4) to draw. This is the time to draw including the time to load Tex from the graphics memory. (equivalent to ***5.4.2.2***)
- +Z is the time to draw when the geometery and rending Z and W (path) attribures are ON.

For Tex size of 256 × 256

Filter	Polygon Size (Pixels)		External Tex	Internal Tex	External Tex + Z	Internal Tex + Z
Point	256*256	65536	1199120	127718	1311783	220794
	128*128	16384	327364	76546	346900	103883
	64*64	4096	89066	63670	92332	71787
	55*55	3025	65901	62472	68593	68782
	50*50	2500	54510	61915	56727	67108
	32*32	1024	22519	59900	23547	62156
	16*16	256	5840	58020	6205	58888
Bilinear	256*256	65536	1886935	218936	1988065	298688
	128*128	16384	648378	105110	671527	126268
	64*64	4096	178150	72675	183427	78790
	40*40	1600	66129	61120	67328	64811
	37*37	1369	56245	60946	57332	63826
	32*32	1024	46660	61061	47877	63228
	16*16	256	11844	58389	12214	59164

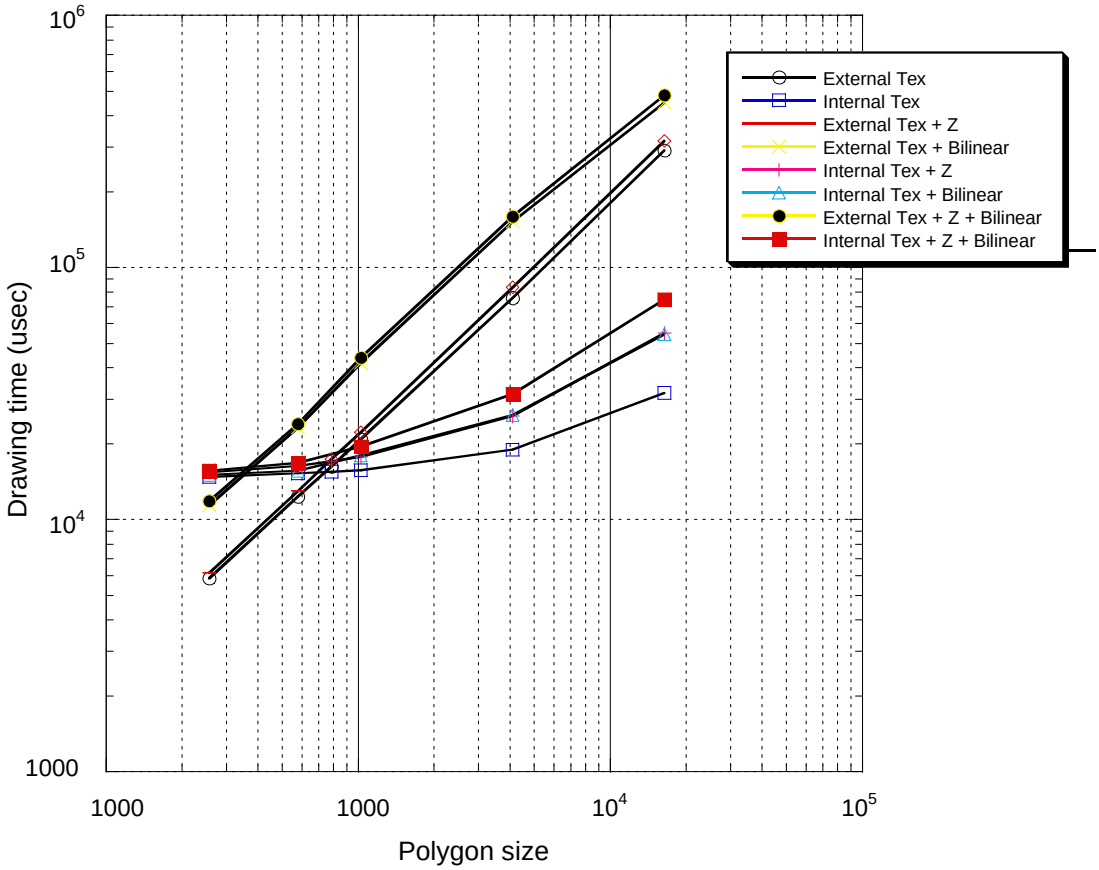
For Tex size of 128 × 128

Filter	Polygon Size (Pixels)		External Tex	Internal Tex	External Tex + Z	Internal Tex + Z
Point	128*128	16384	290857	31772	318667	54850
	64*64	4096	75677	18955	83487	25726
	32*32	1024	20877	15723	22217	17719
	28*28	784	16266	15477	17342	17013
	24*24*	576	12339	15248	12980	16309
	16*16	256	5837	14781	6134	15386
Bilinear	128*128	16384	451236	54532	483523	74585
	64*64	4096	151946	26050	159623	31430
	32*32	1024	41755	17961	43800	19478
	24*24	576	23205	15663	23960	16774
	16*16	256	11394	15068	11835	15610



Graph 1: Tex Size 256 × 256

For Tex sizes of 256 × 256 (loading texture data to the internal TexRAM for 16 times), near the point where the polygon size (the size of the sprite display) exceeds 1000 to 1200 pixels (31 × 31 to 35 × 35), it divides polygons, loads to the internal TexRAM to draw at high speed.



Graph 2: Tex Size 128×128

For Tex sizes of 128×128 (loading texture data to the internal TexRAM for 4 times), near the point where the polygon size (the size of the sprite display) exceeds 300 to 1000 pixels (17×17 to 31×31), it divides polygons, loads to the internal TexRAM to draw at high speed.

● Reference

The following shows the time (usec) for Bit Texture (to load textures from the graphics memory to the internal TexRAM) and the BLT Copy Alternate time (usec) (copying rectangle area between graphics memories) for reference.

- BltTexture:

  The following one time loads 64 × 64 (=4096) pixels from the graphics memory to the internal TexRAM.

100 times (actual measured value)	3325
Per one time	33.25
For 256 × 256: 33.25 × 16	532
For 128 × 128: 33.25 × 4	133

- BLT Copy Alternate:

  Drawing performance when copied 100 times.

Copy size		Copy time
256*256	65536	97599
128*128	16384	24650
64*64	4096	6379
32*32	1024	1848
16*16	256	654

5.5. Orchid Geometry and Rendering Performance

The following shows ***Orchid*** geometry and rendering performance.

Use these as general performance values for developing applications.

Measuring method:

Orchid operating frequency: 100 MHz

Graphics memory bus width: 64 bits

Transfer mode: Local mode
(Writes and draws the display list from the graphics memory.)

- Orchid geometry performance

  Measuring method:
  Average values when drawing 100 ***Triangle_FAN.int*** 10 times to a 5 × 5 pixel area

Vertex parameter	Performance		
	K Polygon/sec.	K Polygon/frame *1	K Polygon/frame *2
XY	805.15298	26.83843	13.41922
XYZ	354.10765	11.80359	5.90179
XYRGB	206.73972	6.89132	3.44566
XYZRGB	180.50542	6.01685	3.00842
XYST	300.30030	10.01001	5.00501
XYRGBST	137.77900	4.59263	2.29632
XYZSTQ	207.42585	6.91419	3.45710
XYZRGBSTQ	122.10012	4.07000	2.03500

*1: In terms of 30 frames/sec.

*2: In terms of 60 frames/sec.

- Orchid rendering performance

Measuring method:
Average values when drawing 100 Triangle_FAN.int 10 times to a triangle 25 × 20 pixel rarea.

Attribute		Performance	
Alpha blend	Vertex parameter	K Polygon/sec.	Mpixel/sec.
Provided	XY	253.74271	63.43568
	XYZ	120.93361	30.23340
	XYRGB	200.00000	50.00000
	XYZRGB	117.64706	29.41176
	XYST	218.81838	54.70460
	XYRGBST	134.39054	33.59763
	XYZSTQ	116.07661	29.01915
	XYZRGBSTQ	112.79044	28.19761
	XYST ( )	22.39240	5.59810
	XYZSTQ ( )	19.73671	4.93418
	XYRGBST ( )	22.26130	5.56533
	XYZRGBST ( )	19.66994	4.91748
Not provided	XY	168.91892	42.22973
	XYZ	81.28099	20.32025
	XYRGB	154.67904	38.66976
	XYZRGB	80.64516	20.16129
	XYST	155.15904	38.78976
	XYRGBST	133.68984	33.42246
	XYZSTQ	80.59317	20.14829
	XYZRGBSTQ	79.85945	19.96486
	XYST ( )	20.61176	5.15294
	XYZSTQ ( )	18.89966	4.72492
	XYRGBST ( )	20.55837	5.13959
	XYZRGBST ( )	18.86010	4.71502