

---

# **Dreamcast/ Dev.Box System Architecture**

**Last Update: 99/09/02 18:41  
(Preliminary)**

---

## **REVISION HISTORY**

[1999]

- 9/2
  - Modified terms: KATANA -> Dreamcast, SET5 -> Dev.Box (Revisions in green: Up to next release)
  - Distinction between HOLLY1 and HOLLY2 were unnecessary, and so combined as just HOLLY. (Revisions in green: Up to next release)
- 8/23
  - Corrected description of FNS and OCT settings in section 8.1.1.4.
  - Corrected description of FNS[9:0] in the register descriptions (channel data) in section 8.4.5.
- 3/10
  - Added descriptions of register modification procedures in section 2.7.2.
- 2/4
  - Corrected register addresses in section 8.4.1.1: 6930 -> 6920, 6950 -> 6930, etc. (Revisions in green: Up to next release)

1/27

[1998]

- 11/25
  - Corrected descriptions of revisions added to section 1.4 and 9.
  - Corrected description of parameters in the common data for the AICA register in section 8.4.5.
- 11/20
  - Made additions to and corrected descriptions related to revisions in sections 1.4 and 9.
  - Added description for the STARTRENDER register (0x005F8014) in section 8.4.2.
  - Corrected a portion of the description of G2-related registers in section 8.4.1.4.
  - Made additions to and corrected supplementary descriptions for the mapping table in section 2.1.
- 11/10
  - Corrected description and headings concerning section 4.2, "G2 Interface." (Revisions in green: Up to next release)
  - Corrected description of G2-related registers in section 8.4.1.4. (Revisions in green: Up to next release)
- 11/4
  - Corrected description of Maple-related registers in section 8.4.1.2. (Revisions in green: Up to next release)
- 10/30
  - Corrected description concerning PVR-DMA registers in section 8.4.1.5. (Revisions in green: Up to next release)
  - Corrected description in section 5, "User Interface" -- Described hard triggers again. (Revisions in green: Up to next release)
- 10/15
- 10/13
  - Corrected description in section 3.6.2.3, "VQ Textures," and Figs. 3-1 and 3-2.
- 9/30
  - Corrected description in Section 1.4, "Dev.Box Board."
  - Changed description in section 9, "Bug List."
  - Added description to section 8.4.2 for the STARTRENDER Register (0x005F8014).
- 9/25
- 9/16
  - Corrected description in section 1.1.5, "Expansion Devices."
- 8/31
  - Corrected description in section 3.7.4.4.3, "Obj Control." bit 16-8 -> bit
- 8/28
  - Added description in section 4.2.3, "RTC."
  - Corrected description in section 8.4.5, "AICA Registers," concerning channel data:PCMS, and cutoff frequency (FLV) and common data: DLG.
  - Corrected description of external memory specifications in section 4.2.2.4, "Wave Memory (AICA)."
  - Corrected portion of description in section 8.1.1, "Audio-related..."
  - Corrected description in section 3.4.5.3, "Modifier Volume Processing of Various Polygons."
  - Corrected table in section 3.5.1, "Sync Pulse Generator."
- 8/21
  - Added and corrected a portion of text and figure in section 3.6.2.4, "MIPMAP Texture."
  - Added to and corrected ISP\_FEED\_CFG Register in section 8.4.2(0x005F8098).
  - Changed a portion of 3.4.3.1, "ISP Cache Size."
  - Changed a portion of 3.4.5.2, "Volume Mode."
  - Changed a portion of 3.4.11.2, "Y Scaler."

- 8/18 · Changed Type A diagram in section 3.4.12, "Flicker-free Interlacing"
- 8/7 · Changed "ARC" to "SRC" in section 3.4.7.2.3, "Trilinear Filtering."
- 8/4 · Corrected a portion of ISP\_FEED\_CFG Register in section 8.4.2(0x005F8098).  
· Corrected a portion of display lists in sections 3.7.8 and 3.7.9.2.  
· Corrected a portion of section 3.4.3, Punch Through Polygons."
- 7/31
- 7/28 · Changed a portion of section 3.3, "Register Map (Graphics System)."
- 7/27 · Corrected a portion of Figs. 3-77 and 3-78 in section 3.7, "Display List Details."  
· Corrected a portion of FPU\_PARAM\_CFG Register (0x005F807C) in section 8.4.2.
- 7/14 · Corrected a portion of section 3.1.1.1.  
· Corrected a portion of the HOLY version table in section 1.4, "Dev.Box."
- 7/9 · Corrected all pages.  
· Changed "Opaqu" to "Opaque" in section 3.1.1.9, "Polygon List."
- 7/7 · Made "Expansion Devices" the term used consistently for external expansion devices that are connected to the G2 Bus.  
· Added description to section 3.4.3, "Punch Through Polygons."  
· Corrected references in section 3.7.7, "Region Array Data Configuration."  
· Corrected explanation in section 4.2.5, "Expansion Devices."  
· Corrected SDRAM\_CFG Register (0x005F80A8) in section 8.4.2.
- 7/1 · Deleted the hidden character portion of section 4.1.4, "System Codes."  
· Changed the underlining of the additional HOLLY2 specifications in section 3, "Graphics System," from a broken line to a wavy line.  
· Returned the setting for the portion of section 5.1.6 that was composed of hidden characters back to normal characters.
- 6/30 · Submitted to Software Technology Development Group.
- 6/10 · Submitted to Software Technology Development Group.
- 5/26 · Submitted to Software Technology Development Group.
- 5/1 · Submitted to Software Technology Development Group.
- 4/21 · Submitted to Software Technology Development Group.
- 3/27 · Submitted to Software Technology Development Group.
- 2/24 · Submitted to Software Technology Development Group.
- 2/4 · Submitted to Software Technology Development Group.
- 1/30 · Submitted to Software Technology Development Group.
- 1/23 · Submitted to Software Technology Development Group.
- 12/16 · Submitted to Software Technology Development Group.
- 11/25 · Submitted to Software Technology Development Group.

---

## Table of Contents

Dreamcast/Dev.Box System Architecture .....	1
<u>REVISION HISTORY</u> .....	2
§1 THE SYSTEM.....	9
§1.1 Overview .....	10
§1.2 System Architecture.....	10
§1.3 Block Diagram.....	11
§1.4 Dev.Box Board .....	14
§2 CPU AND PERIPHERAL MEMORY.....	15
§2.1 System Mapping .....	16
§2.1.1 Cache Access .....	18
§2.2 SH4 .....	20
§2.2.1 Overview of the SH4 .....	20
§2.2.2 CPU Bus Interface .....	21
§2.2.3 Initial Settings for the SH4.....	22
§2.3 System Memory .....	34
§2.3.1 System Memory Configuration and Control .....	34
§2.3.2 System Memory Initial Settings.....	34
§2.3.3 Access Procedure .....	36
§2.4 Register Map.....	37
§2.5 Single Access to Each Block .....	41
§2.6 DMA Transfers .....	42
§2.6.1 Overview of DMA Transfers .....	42
§2.6.2 Types of DMA.....	43
§2.6.3 GD-ROM Data Transfers .....	44
§2.6.4 Texture Data Transfers .....	47
§2.6.4.1 Direct Texture Transfers.....	47
§2.6.4.2 YUV Texture Transfer.....	55
§2.6.5 Display List Transfers.....	57
§2.6.5.1 Direct Display list DMA .....	57
§2.6.5.2 TA Input Display List Transfers .....	59
§2.6.5.3 Sort-DMA Transfer of $\alpha$ Polygon Parameters.....	61
§2.6.6 Wave Data Transfers .....	72
§2.6.7 ARM Data Transfers .....	77
§2.6.8 Peripheral Data Transfers .....	78
§2.6.9 Color Palette Transfers .....	80
§2.6.10 External Data Transfer .....	82
§2.7 Interrupts .....	83
§2.7.1 Overview .....	83
§2.7.2 Interrupt Settings and Access Procedures.....	84
§2.7.3 Notes Concerning Interrupts .....	89
§3 The Graphics System .....	90
§3.1 Overview .....	91
§3.1.1 Graphics Architecture.....	91
§3.1.1.1 Basic Polygons.....	91
§3.1.1.2 Coordinate System .....	92
§3.1.1.3 Display List.....	92
§3.1.1.4 Tile Partitioning and Surface Equations.....	92
§3.1.1.5 Block Diagram.....	94
§3.1.1.6 Triangle Setup .....	95

---

---

§3.1.1.7 <i>ISP(Image Synthesis Processor)</i> .....	96
§3.1.1.8 <i>TSP(Texture and Shading Processor)</i> .....	96
§3.1.1.9 <i>Polygon List</i> .....	97
§3.1.2 Drawing Function Overview .....	98
§3.1.3 Display Function Overview.....	99
§3.2 Memory Map .....	99
§3.3 Register Map.....	100
§3.4 Drawing Function Details .....	102
§3.4.1 Background .....	102
§3.4.2 Translucent Polygon Sort.....	103
§3.4.2.1 <i>Auto-sort Mode</i> .....	103
§3.4.2.2 <i>Pre-sort Mode</i> .....	104
§3.4.3 Punch Through Polygons.....	104
§3.4.3.1 <i>ISP Cache Size</i> .....	104
§3.4.3.2 <i>Relationship with Translucent Polygons</i> .....	105
§3.4.4 Processing List Discarding .....	105
§3.4.5 Modifier Volume .....	107
§3.4.5.1 <i>Inclusion and Exclusion Volumes</i> .....	108
§3.4.5.2 <i>Volume Modes</i> .....	108
§3.4.5.3 <i>Modifier Volume Processing for Various Polygons</i> .....	109
§3.4.6 Flow of Texture Mapping and Shading Processing .....	110
§3.4.6.1 <i>Secondary Accumulation Buffer</i> .....	111
§3.4.7 Texture Mapping .....	112
§3.4.7.1 <i>MIPMAP</i> .....	112
§3.4.7.2 <i>Texture Filtering</i> .....	112
§3.4.7.2.1 <i>Point Sampling</i> .....	113
§3.4.7.2.2 <i>Bi-linear Filtering</i> .....	114
§3.4.7.2.3 <i>Tri-linear Filtering</i> .....	116
§3.4.7.2.4 <i>Texture Super-Sampling</i> .....	117
§3.4.7.3 <i>Bump Mapping</i> .....	118
§3.4.7.3.1 <i>Bump Mapping Algorithm</i> .....	119
§3.4.7.3.2 <i>Bump Mapped + Textured Polygons</i> .....	120
§3.4.8 Fog Processing .....	122
§3.4.8.1 <i>Look-up Table Mode</i> .....	122
§3.4.8.2 <i>Per Vertex Mode</i> .....	123
§3.4.9 Clipping .....	124
§3.4.9.1 <i>Tile Clipping</i> .....	124
§3.4.9.2 <i>Pixel Clipping</i> .....	126
§3.4.10 Drawing to a Texture Map.....	126
§3.4.11 X Scaler & Y Scaler.....	127
§3.4.11.1 <i>X Scaler</i> .....	128
§3.4.11.2 <i>Y Scaler</i> .....	129
§3.4.12 Flicker-free Interlacing.....	129
§3.4.12.1 <i>Type A</i> .....	130
§3.4.12.2 <i>Type B</i> .....	131
§3.4.13 Strip Buffers .....	132
§3.4.14 Frame Buffer Drawing Data and Display Data.....	134
§3.5 Display Function Details .....	135
§3.5.1 Sync Pulse Generator .....	135
§3.5.2 Frame Buffer Settings .....	135
§3.6 Texture Definition .....	137
§3.6.1 Texture Pixel Format.....	138
§3.6.1.1 <i>RGB Textures</i> .....	138
§3.6.1.2 <i>YUV Textures</i> .....	138
§3.6.1.3 <i>Bump Map Textures</i> .....	139
§3.6.1.4 <i>Palette Textures</i> .....	140
§3.6.2 Texture Formats .....	141

---

---

§3.6.2.1 Twiddled Format.....	141
§3.6.2.2 Non-Twiddled Format .....	143
§3.6.2.3 VQ Textures .....	144
§3.6.2.4 MIPMAP Texture .....	146
§3.6.3 Color Data Extension .....	149
§3.6.4 Texture Format Combinations.....	149
§3.6.5 Efficient Storage in Texture Memory.....	151
§3.7 Display List Details.....	152
§3.7.1 Polygon List Input.....	155
§3.7.1.1 TA Parameter Input Flow .....	157
§3.7.1.2 TA Register Settings for List Input.....	157
§3.7.1.3 Region Array Data Storage.....	159
§3.7.1.4 Object List Starting Address for Each List .....	161
§3.7.2 Tile Arrangement.....	163
§3.7.3 Tile Accelerator .....	164
§3.7.3.1 Strip Partitioning.....	164
§3.7.3.2 Tile Division .....	165
§3.7.3.3 Tile Clipping.....	166
§3.7.3.4 Object List Generation .....	166
§3.7.3.4.1 List Initialization Processing and List Continuation Processing.....	167
§3.7.3.4.2 Adding an OPB.....	170
§3.7.3.4.3 Processing When a Limit Address Is Exceeded .....	173
§3.7.3.5 ISP/TSP Parameter Generation .....	174
§3.7.4 Explanation of TA Parameters .....	175
§3.7.4.1 Control Parameter .....	175
§3.7.4.2 Global Parameter .....	177
§3.7.4.3 Vertex Parameter.....	178
§3.7.4.4 Parameter Control Word.....	179
§3.7.4.4.1 Para Control .....	179
§3.7.4.4.2 Group Control .....	180
§3.7.4.4.3 Obj Control.....	180
§3.7.5 Parameter Format .....	183
§3.7.5.1 Control Parameter Format .....	183
§3.7.5.2 Global Parameter Format.....	184
§3.7.5.3 Vertex Parameter Format.....	186
§3.7.6 Overview of TA Parameters .....	190
§3.7.6.1 Notes When Using the TA .....	190
§3.7.6.2 Parameter Combinations .....	191
§3.7.6.3 Parameter Input Example .....	191
§3.7.7 Region Array Data Configuration .....	194
§3.7.8 Object List Data Configuration .....	197
§3.7.9 ISP/TSP Parameter Data Configuration .....	199
§3.7.9.1 ISP/TSP Instruction Word .....	201
§3.7.9.2 TSP Instruction Word.....	205
§3.7.9.3 Texture Control Word.....	210
§3.8 Details on Miscellaneous Functions .....	212
§3.8.1 YUV-data Converter .....	212
§4 Peripheral Interface .....	215
§4.1 G1 Bus .....	216
§4.1.1 GD-ROM .....	216
§4.1.1.1 Register Map.....	216
§4.1.1.2 Access Methods.....	217
§4.1.1.3 Initial Settings.....	217
§4.1.1.4 Access Procedure .....	217
§4.1.2 System ROM.....	218
§4.1.2.1 Access Methods.....	218
§4.1.2.2 System Initial Settings.....	218
§4.1.2.3 Access Procedure .....	218

---

---

§4.1.3 FLASH Memory.....	219
<b>§4.1.3.1 System Initial Settings .....</b>	<b>219</b>
<b>§4.1.3.2 Access Procedure .....</b>	<b>219</b>
§4.1.4 System Code.....	220
<b>§4.1.4.1 Initial Setting .....</b>	<b>220</b>
<b>§4.1.4.2 Access Procedure .....</b>	<b>220</b>
§4.2 G2 Interface .....	221
§4.2.1 Interface .....	221
§4.2.2 AICA.....	225
<b>§4.2.2.1 Memory/Register Map.....</b>	<b>226</b>
<b>§4.2.2.2 Initial Settings .....</b>	<b>227</b>
<b>§4.2.2.3 Access Procedure .....</b>	<b>227</b>
<b>§4.2.2.4 Wave Memory .....</b>	<b>229</b>
§4.2.3 RTC(Real Time Clock).....	230
<b>§4.2.3.1 Access Method .....</b>	<b>230</b>
§4.2.4 MODEM.....	231
<b>§4.2.4.1 Address Map .....</b>	<b>231</b>
<b>§4.2.4.2 Access Method .....</b>	<b>231</b>
§4.2.4.2.1 ID .....	232
§4.2.4.2.2 Reset.....	232
§4.2.5 Expansion Devices.....	233
§5 User Interface.....	239
§5.1 Peripherals.....	240
§5.1.1 Overview .....	240
§5.1.2 Register Map.....	242
§5.1.3 Operating Sequence .....	243
§5.1.4 Access Procedure .....	245
§5.1.5 Example of Transmission and Reception Data.....	247
§5.1.6 Notes Regarding Access.....	249
§5.2 Control Pad .....	251
§5.3 Light Phaser Gun .....	251
§5.4 Backup (Option) .....	252
§5.5 Sound Recognition (Option) .....	252
§6 Peripheral Devices .....	253
§6.1 DVE (Digital Video Encoder).....	254
§7 deBugger .....	256
§8 Appendix .....	258
§8.1 Technical Explanations .....	259
§8.1.1 Technical Explanation Concerning Audio .....	259
<b>§8.1.1.1 Loop Control .....</b>	<b>259</b>
<b>§8.1.1.2 ADPCM .....</b>	<b>261</b>
<b>§8.1.1.3 AEG.....</b>	<b>264</b>
<b>§8.1.1.4 PG.....</b>	<b>265</b>
<b>§8.1.1.5 LFO .....</b>	<b>266</b>
<b>§8.1.1.6 Mixer .....</b>	<b>267</b>
<b>§8.1.1.7 FEG .....</b>	<b>269</b>
<b>§8.1.1.8 Audio DSP .....</b>	<b>270</b>
§8.1.2 Reset Sequence.....	275
§8.1.3 Clock .....	280
<b>§8.1.3.1 PLL .....</b>	<b>280</b>
<b>§8.1.3.2 Clock Tree.....</b>	<b>280</b>
§8.1.4 JTAG Interface .....	282
<b>§8.1.4.1 SH4 .....</b>	<b>282</b>
<b>§8.1.4.2 HOLLY .....</b>	<b>282</b>

---

---

<b>§8.1.4.3 AICA .....</b>	<b>282</b>
§8.2 Individual Block Diagrams .....	282
§8.2.1 Detailed Block Diagram of Entire System .....	282
§8.2.2 CPU Subsystem (Including System Memory) .....	282
§8.2.3 HOLLY Subsystem .....	282
§8.2.4 GD-ROM Subsystem .....	282
§8.2.5 AICA Subsystem .....	282
§8.2.6 Digital Video Encoder Subsystem .....	282
§8.2.7 16Mbit SDRAM (16bit) .....	282
§8.2.8 64Mbit SGRAM (32bit) .....	282
§8.2.9 Power Supply .....	282
§8.3 Pin Assignments (with Descriptions of Pins) Pin Assignments for Each Chip .....	282
§8.3.1 CPU .....	282
§8.3.2 HOLLY .....	282
§8.3.3 GD-ROM .....	282
§8.3.4 AICA .....	282
§8.3.5 Digital Video Encoder .....	282
§8.3.6 16Mbit SDRAM (16bit) .....	282
§8.3.7 64Mbit SGRAM (32bit) .....	282
§8.4 List of Registers .....	283
§8.4.1 System Bus Register .....	283
<b>§8.4.1.1 System Registers .....</b>	<b>284</b>
<b>§8.4.1.2 Maple Peripheral Interface .....</b>	<b>296</b>
<b>§8.4.1.3 G1 Interface .....</b>	<b>303</b>
<b>§8.4.1.4 G2 Interface .....</b>	<b>314</b>
<b>§8.4.1.5 PowerVR Interface .....</b>	<b>323</b>
§8.4.2 CORE Registers .....	327
§8.4.3 Tile Accelerator Registers .....	349
§8.4.4 GD-ROM Registers .....	354
§8.4.5 AICA Register .....	360
§8.5 List of Interrupts .....	377
§8.5.1 Interrupt Tree .....	377
§8.5.2 List of Interrupt Sources .....	378
§8.6 List of Input Parameters .....	382
§9 Bug List .....	390

---



## **§ 1 THE SYSTEM**

## § 1.1 Overview

The Dreamcast system, based on the PowerVR Family core, includes a high-performance graphics system, a 64-channel audio system that is capable of various effects, and a 12x (max.) GD-ROM drive. In addition, the Dreamcast system provides excellent cost performance.

In addition, in order to facilitate expansion into the network/internet business, the main unit of the Dreamcast system is designed to accept a plug-in modem card.

This section provides an overview of the Dreamcast system. For further details on specific blocks, please refer to the individual sections that correspond to those blocks later in this manual.

## § 1.2 System Architecture

The basic hardware specifications for the Dreamcast system are listed below.

- CPU: Hitachi SH4 - 200MHz, super-scalar RISC processor, 360MIPS, 1.4GFLOPS
- ASICs: Graphics: VL/NEC HOLLY - 100MHz; audio: Yamaha AICA - 22/25MHz
- Polygon performance: 1 million polygons/sec (100-pixel triangles, opaque - 75%, translucent - 25%)
- Polygon functions: Shadowing, trilinear mip-map, Fog, Z buffering, etc.
- 16MB system memory
- 8MB texture memory (can be expanded up to 16MB)
- 2MB audio memory (can be expanded up to a maximum of 8MB)
- 2MB system ROM
- 128K flash memory (for system code)
- Audio function: 64ch PCM/ADPCM, 44.1kHz
- 4x to 12x CAV-type GD-ROM drive (128K of built-in buffer RAM)
- Four game ports (peripheral ports)
- RTC that permits battery backup
- Supports NTSC/PAL and VGA video output
- 33.6kbps modem card (LINE jack) that operates off of 3.3V

The specifications for Dreamcast system options are listed below.

- Supports light phaser gun, backup storage media, and voice recognition as peripheral devices that connect to a game port.
- Supports externally connected expansion devices.

## § 1.3 Block Diagram

Fig. 1-1 shows a block diagram of the system.

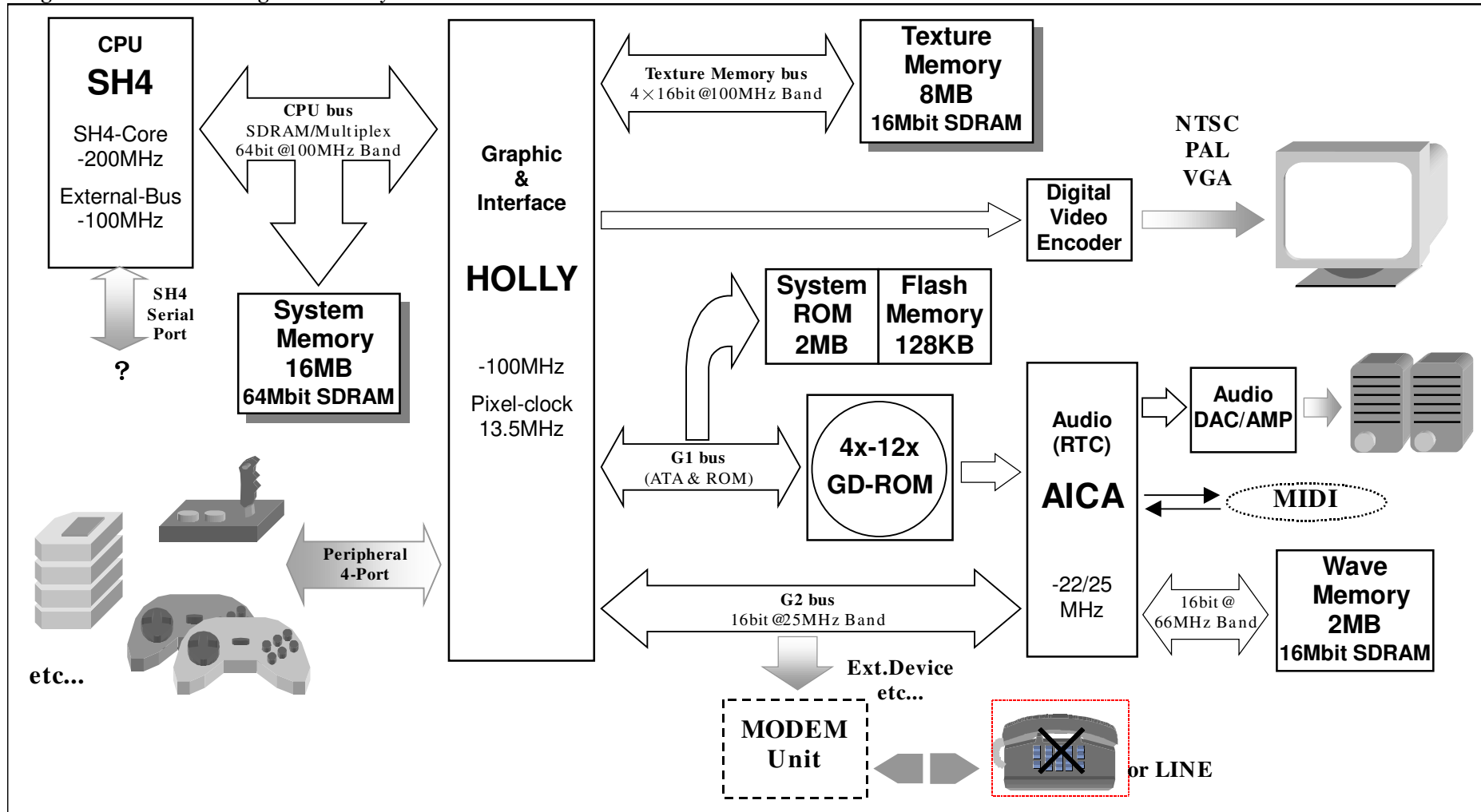


Fig. 1-1 System Block Diagram

Overviews of each block and the primary devices are provided below.

### **CPU**

The main CPU is a Hitachi SH4, which accepts a 33.3MHz clock signal from the system and, by means of an internal PLL, operates at 1.8V/200Mhz internally and at 3.3V/100MHz for the external bus. The SH4 is primarily responsible for processing concerning the game sequence, AI, 3D calculations, and issuing 3D graphics instructions. In addition, the SH4 also provides a general-purpose serial port with a FIFO buffer for use by external I/O devices. The serial port uses start-stop synchronization, and supports a maximum transfer speed of 1.5625Mbps.

### **Peripheral Memory**

In order to make the best use of the performance capabilities of the SH4, SDRAM is used for the main system memory, and is connected directly to the SH4. There are 16MB of main memory, the bus width is 64 bits, and the operating frequency is 100MHz. The (theoretical) maximum burst transfer speed is 800MB/s. In addition, aside from DMA transfers from the graphics and the interface chip, this system memory is used only by the SH4. 7ns chips or the equivalent are used for this memory.

The Dreamcast System also has 2MB of system ROM, where the operating system, boot routine, etc., are stored.

There is also a 128K flash memory that is used to store system information, such as region information, manufacturer code, etc.

### **Graphics System**

A feature of the Dreamcast graphics system is high-performance 3D graphics, and can produce output in a variety of video modes with 8-bit RGB data as the color information. The Dreamcast graphics core uses the DMA transfer capability of the SH4 (the CPU) to retrieve display lists created by the SH4 in system memory; the graphics core then uses these display lists to generate 3D images internally. Because the raster algorithm is used for the drawing method, there is no need for a frame buffer in order to generate 3d graphics; for texture mapping, textures are loaded in from dedicated texture memory. The standard graphics memory in the Dreamcast System is 8MB, but this can be expanded to 16MB for development work.

The Dreamcast System supports video output for typical NTSC/PAL monitors as well as for VGA monitors (such as personal computer displays). In addition to the stereo sound that is output from the audio system, the Dreamcast system also outputs audio on a general-purpose RCA connector and an extended VGA connector.

### **Audio System**

The Dreamcast System can generate stereo output from the 64-channel PCM/ADPCM sound source that is built into the audio chip, and also supports various effects through the sound CPU and DSP that are also built into the chip. This output can also be mixed with sound data that is output from the GD-ROM. The system and the sound CPU and DSP all share a common wave memory, which has a 2MB capacity in the base system. (This capacity can be expanded to 8MB for development work.) The MIDI interface is also supported for development work as the audio peripheral interface.

The GD-ROM drive that is built into the Dreamcast system is used to load sound data as well as data that is used by the game software, etc. The GD-ROM drive supports various CD formats, and rotates according to the CAV system. The data reading speed ranges from 4x to 12x. An ATAPI device is used for the drive.

The stereo sound that is generated by the audio system passes through an audio DAC/AMP, and is then output on the RCA connector and extended VGA connector, along with the video output that is generated by the graphics system.

### **User Interface**

Sega's proprietary serial peripheral interface is used for the user interface devices, such as the control pads. The main unit of the Dreamcast system supports up to four ports. In addition to control pads, these ports also support connection with light phaser gun, backup storage media, etc. the maximum transfer rate through these ports is 2Mbps.

### **Expansion Device**

Debuggers for use in software development can be connected to the expansion connector as expansion devices.

### **Communications System**

The Dreamcast system supports a plug-in modem card. The communications speed of this modem is 33.6Kbps, and the modem includes a modular line jack.

Supplemental descriptions of the buses in relation to the hardware are provided below.

### **CPU bus**

This bus connects the SH4, the CPU, to the 16MB system memory and to "HOLLY," the graphics/interface core. Between the CPU and system memory, this is an SDRAM interface with a 64-bit data width, and between the CPU and HOLLY, this is a 64-bit bus interface on which addresses and data are multiplexed.

As mentioned on the previous page, the bus clock in both cases is 100MHz.

### **Texture memory bus**

Supported by the HOLLY's internal PowerVR core, this is an SDRAM interface bus for texture memory, which is memory that is used for drawing and display functions. The bus clock is 100MHz, and the bus width is 64 bits (16 bits x 4).

### **Wave memory bus**

This is an SDRAM interface bus for audio that is supported by AICA. The bus clock is 67.7MHz (2 x 33.8688MHz, which is supplied from the GD-ROM to AICA). The bus width is 16 bits.

### **G1 bus**

The G1 bus is supported by HOLLY. The GD-ROM, system ROM, flash memory and other asynchronous devices are connected to the G1 bus in parallel. The access method used on the G1 bus differs according to the target device, with accesses to the GD-ROM device being different from accesses to system ROM or flash memory. Access is based on the ATA standard, according to a protocol that supports the ATA standard in part. One interrupt line from the GD-ROM is supported. Regarding data transfers, DMA transfers are possible in the GD-ROM area.

This G1 bus also supports the loading of 8 bits of data (a country code) that are set on the board.

### **G2 bus**

The G2 bus is supported by HOLLY. This bus supports the audio chip AICA, a modem, external expansion devices, and other synchronous devices. The G2 bus is basically a PCI-like bus, with a bus clock of 25MHz and a bus width of 16 bits. The bus supports three interrupt lines, one for each of the supported devices listed above. Aside from the modem, DMA transfer is possible with the AICA and expansion devices.

## § 1.4 Dev.Box Board

This section describes the board settings and electrical aspects of the hardware.

- About the HOLLY revisions (=CLX: chipmaker code name)...  
Each revision of Holly is the result of problems with chips or changes to the specifications.  
The version can be identified by the SH4 (the CPU) by reading the revision register in the system bus block and the CORE block.  
The following table lists the three types of internal registers that are used to identify the chip in each block. The register addresses shown in the table below are the addresses in the P2 guncacheable) area of the SH4. (Refer to section 2.1.)

0xA05F689C bit7-0 (reg. SB_SBREV)	0xA05F7880 bit7-0 (reg. SB_G2ID)	0xA05F8004 bit15-0 (REVISION)	Chip currently in use
0x01	0x12	0x0001	HOLLY1.0 / 1.1 (CLX1 1.0 / 1.1)
0x02	0x12	0x0001	HOLLY1.5/1.6 (CLX1 1.5/1.6)
0x08	0x12	0x0011	HOLLY ES2.2 (CLX2.2)
0x09	0x12	0x0011	HOLLY ES2.3 (CLX2.3)
0x0A	0x12	0x0011	HOLLY ES2.4/2.41 (CLX2.4/2.41)
0x0B	0x12	0x0011	HOLLY ES2.42 (CLX2.42)

There are different versions of Dev.Box for the different HOLLY versions, as described below:

Dev.Box Ver. 5.05	Dev.Box Ver. 5.16	Dev.Box Ver. 5.22	Dev.Box Ver. 5.23	Dev.Box Ver. 5.24
HOLLY ES1.1	HOLLY ES1.6	HOLLY ES2.2	HOLLY ES2.3	HOLLY ES2.4 ~

\* Details on each HOLLY revision are provided in section 9.

- The drive capacity of all Maple-related pins on the HOLLY chip is 6mA (BFU C23).
- The withstand voltage for G1 devices connected to the HOLLY's G1 bus is ~~3.3V for the HOLLY1 and~~ **5V for the HOLLY2.**

## **§ 2 CPU AND PERIPHERAL MEMORY**

This section describes the main processor of the Dreamcast System and the system memory.

## § 2.1 System Mapping

Table 2-1 shows the memory map for physical addresses in the Dreamcast System. Refer to their respective sections for details on individual functions.

Area	Physical Address	Type	Function	Size	Access	Note
0	0x0000000 - 0x001FFFFF	MPX	System/Boot ROM	2MB	1/2/4/32	in G1 i/f
	0 - 0x0021FFFF		Flash Memory	128KB	1/2/4/32	in G1 i/f
	0x0020000 - 0x005F67FF		Unassigned	-	-	Reserved
	0 - 0x005F69FF		System Control Reg.	512B	4	
	0x0040000 - 0x005F6CFF		Maple i/f Control Reg.	256B	4	
	0 - 0x005F70FF		GD-ROM	256B	1/2	in G1 i/f
	0x005F680 - 0x005F74FF		G1 i/f Control Reg.	256B	4	
	0 - 0x005F78FF		G2 i/f Control Reg.	256B	4	
	0x005F6C0 - 0x005F7CFF		PVR i/f Control Reg.	256B	4	
	0 - 0x005F9FFF		TA / PVR Core Reg.	8KB	4/32	
	0x005F700 - 0x006007FF		MODEM	2KB	1	in G2 i/f
	0 - 0x006FFFFFFF		G2 (Reserved)	-	-	in G2 i/f
	0x005F740 - 0x00707FFF		AICA- Sound Cntr. Reg.	32KB	4	in G2 i/f
	0 - 0x0071000B		AICA- RTC Cntr. Reg.	12B	4	in G2 i/f
	0x005F780 - 0x00FFFFFFF		AICA- Wave Memory	2/8MB	4	in G2 i/f
	0 - 0x01FFFFFFF		Ext. Device	16MB	1/2/4/32	in G2 i/f
	0x005F7C0 - 0x03FFFFFFF*		Image Area*	32MB*		
	0					
	0x005F800					
	0					
	0x0060000					
	0					
	0x0060080					
	0					
	0x0070000					
	0					
	0x0071000					
	0					
	0x0080000					
	0					
	0x0100000					
	0					
	0x0200000					
	0					
1	0x0400000 - 0x04FFFFFFF	MPX	Tex.Mem. 64bit Acc.	8/16MB	2/4/32	in TA/PVR
	0 - 0x05FFFFFFF		Tex.Mem. 32bit Acc.	8/16MB	2/4/32	in TA/PVR
	0x0500000 - 0x07FFFFFFF*		Image Area*	32MB*		
	0					
2	0x0600000	-	Unassigned	-	-	
	0					
3	0x0800000 - 0x0BFFFFFFF	SDRAM	System Memory (Image) Image Area*	16MB 16MB 32MB*	1/2/4/32	Work
	0 - 0x0DFFFFFFF					
	0x0D0000 - 0x0FFFFFFF*					
	00					
	0x0E00000					
	0					
	0					



4	0x1000000 - 0x107FFFFF 0 - 0x10FFFFFF 0x1080000 - 0x11FFFFFF 0 - 0x13FFFFFF* 0x1100000 0 0x1200000 0	MPX	TA FIFO Polygon Cnv. TA FIFO YUV Conv. Tex.Mem. 32/64bit Acc. Image Area*	8MB 8MB 16MB 32MB*	32(w) 32(w) 32(w)	in TA block in TA block thru TA
5	0x1400000 - 0x17FFFFFF 0	MPX	Ext. Device	64MB	1/2/4/32	in G2 i/f
6	0x1800000 - 0x1BFFFFFF 0	-	<i>Unassigned</i>	-	-	Reserved
7	0x1C00000 - 0x1FFFFFFF 0	-	(SH4 Internal area)	-	-	

Notes:

- Locations marked with an asterisk in the above table indicate the address image for the first half of the corresponding 64MB area, divided into 32MB sections. (Example: System Control Registers = 0x005F6800 ~ → 0x025F6800 ~)

In addition, the area from 0x02000000 to 0x021FFFFFFF does not contain the System/Boot ROM image, and the area from 0x02200000 to 0x023FFFFFFF does not contain the flash memory image. Both are unused areas.

Image areas other than those indicated by the "\*" mark are not marked.

"Area" refers to the area divisions in the CPU, each of which is a block of 64MB of physical area.

"Access" shows the unit of access, in bytes. All accesses can basically be reads or writes, but those locations where the "(w)" notation appears are write-only accesses.

Table 2-1 Physical Memory Map

### § 2.1.1 Cache Access

Table 2-1 indicates the mapping of physical addresses in the system, which corresponds to an external memory space that is addressed using the 29-bit (A[28:0]) addresses used by the SH4, the CPU. The specification of actual addresses from the SH4 varies according to the SH4 cache access selection, and depends on the contents of the upper three bits (A[31:29]) of the SH4 physical memory space (A[31:0]), as shown in the table below.

A[31:29] ]	Address	Area	Cache
000	0x00000000~0x1FFFFFFF	P0	Cacheable
001	0x20000000~0x3FFFFFFF	P0	Cacheable
010	0x40000000~0x5FFFFFFF	P0	Cacheable
011	0x60000000~0x7FFFFFFF	P0	Cacheable
100	0x80000000~0x9FFFFFFF	P1	Cacheable
101	0xA0000000~0xAFFFFFFF	P2	Non-Cacheable
110	0xC0000000~0xCFFFFFFF	P3	Cacheable
111	0xE0000000~0xFFFFFFFF	P4	Non-Cacheable(SH4 internal area)

Table 2-2 Cache Access

The following table shows the areas for which cache access is possible by the CPU. The addresses shown in parentheses are an image area.

Address	Device/Block	Access
0x00000000~0x001FFFFFFF (0x02000000~0x021FFFFFFF)	System/Boot ROM	R/-
0x00200000~0x0021FFFFFFF (0x02200000~0x0221FFFFFFF)	FLASH Memory	R/-
0x0C000000~0x0CFFFFFFF (0x0E000000~0x0EFFFFFFF)	System Memory	R/W
0x10000000~0x107FFFFFFF (0x12000000~0x127FFFFFFF)	Polygon Converter [Thru TA FIFO]	-/W
0x10800000~0x10FFFFFFF (0x12800000~0x12FFFFFFF)	YUV Converter [Thru TA FIFO]	-/W
0x11000000~0x117FFFFFFF (0x13000000~0x137FFFFFFF)	Texture Memory [Thru TA FIFO]	-/W
0x04000000~0x047FFFFFFF (0x06000000~0x067FFFFFFF)	Texture Memory-64bit Acc. [Thru PVR i/f]	R/W
0x05000000~0x057FFFFFFF (0x07000000~0x077FFFFFFF)	Texture Memory-32bit Acc. [Thru PVR i/f]	R/W
0x01000000~0x01FFFFFFF (0x03000000~0x03FFFFFFF) , 0x14000000~0x17FFFFFFF	G2 External area	Depends on device

Table 2-3 Cache Accessible Area

Cautions concerning cache access are shown below.

- When using a path through a TA FIFO for a cache access, set the write address on a 32-byte boundary. (The data is written in the order that it was output to the FIFO.)
- With the TA FIFO, if a writeback is generated before 32 bytes are collected, the data that is

available at that point is sent to the TA FIFO. Therefore, it is necessary to control writebacks when using a cache via the TA FIFO.

Note that the areas other than the system boot ROM and the flash memory that are shown in the table above can be accessed through the SH4's store queue function. For details, on the SH4 memory space and cache area, and the store queue function, refer to the SH4 manual.

## § 2.2 SH4

The CPU used in the Dreamcast system is the Hitachi SH4; in 3D game programming, this CPU is primarily responsible for processing concerning the game sequence, AI, physical calculations, 3D conversion, etc. This section explains the settings for the SH4 and its peripheral circuits.

### § 2.2.1 Overview of the SH4

Table 2-4 below lists the main features of the SH4.

<b>Core</b>	
Instruction core	32-bit RISC, 16-bit instructions
Pipeline	5 stages
FPU	Single/double precision IEEE754
Clock	Internal: 200MHz; external: 100MHz (1/2, 1/3, 1/4); peripheral clock: 50MHz
Performance	360MIPS (core), 1.4GFLOPS (matrix multiplier)
Super scalar	2
Cache	I\$: 8K; D\$: 16K (direct mapping for both), index/RAM functions
Miscellaneous	Capable of high-speed packet transfer through store queue function (32 bytes, two channels)
<b>Peripheral circuits</b>	
Arithmetic operations	Matrix multiplier, $1/\sqrt{\quad}$
DMAC	4 channels, DDT (on Demand Data Transfer) for channel 0
Memory interface	SDRAM, multiplex interface
MMU	Page sizes: 1KB, 4KB, 64KB, 1MB
UBC	Two break points
SCI	Clock synchronization, start-stop synchronization serial interface
Timer	3 channels
RTC	Real-time clock, alarm, calendar
<b>Process and package</b>	
Power consumption	1.8W
Operating voltage	External: 3.3V DC; internal: 1.8V DC
Process	0.25μm, 4-layer metal, 1.8V/3.3V
Package	256-pin BGA

Table 2-4 Features of the SH4

For details on the SH4, refer to the SH4 manuals (regarding hardware, programming, etc.).

### § 2.2.2 CPU Bus Interface

The buses that connect the peripheral devices to the SH4 (the CPU buses) consist of a 26-bit address bus (the SH4 uses 32-bit internal addressing) and a 64-bit data bus that permits byte access. The CPU buses connect directly to the main system memory (SDRAM) and to the HOLLY chip, which is the graphics/interface core.

The clock speed is 100MHz, with single accesses being 1/2/4 bytes and burst accesses being 32 bytes. The SH4 uses two different bus protocols on the CPU buses, depending on the memory area that has been mapped. One of the following two choices is selected, depending on memory area is to be accessed:

- (1) Direct operation to SDRAM
- (2) MPX operation to HOLLY

A "Direct operation to SDRAM" is an operation that is performed once the SH4 is directly connected to SDRAM, the system memory. The address and data buses form the interface with SDRAM. Area 3 (of the seven physical memory area divisions in the SH4) is used, and memory access is possible in bank active mode. Note that this type of access does not use the upper address bits (A[25:17]).

An "MPX operation to HOLLY" is an operation that multiplexes the address and data on the 64-bit data bus, and, in the Dreamcast System, is performed in all SH4 areas 0, 1, 4, and 5. Areas 0, 1, and 5 use 3 soft waits (+ external waits), and area 4 uses 0 waits (+ external waits). The devices that are assigned to each area are listed below (refer to Table 2-1):

- (1) Area 0 = Accesses to system ROM, GD-ROM, AICA, and other peripheral devices, and the control registers
- (2) Area 1 = Accesses to texture memory
- (3) Area 4 = Write accesses to the HOLLY TA area (FIFO, YUV converter), and to texture memory
- (4) Area 5 = Area for expansion devices on the G2 bus

The graphics/interface core HOLLY is divided into three blocks: the Power VR core (CORE) block, which is the graphics-related block; the Tile Accelerator (TA) block, which is used during data transfers to the CORE; and the System Bus (SB) block, which is the interface block that handles data transfers among all devices, including the graphics-related block. (Details on each of these blocks are provided in subsequent sections.)

Each of these blocks is accessed from the SH4 through the interfaces listed below.

#### <System register interface>

This is the interface between the SH4 and the HOLLY's internal system registers; the root bus (the bus that links all of the interfaces in the SB block) does not pass through this interface.

This interface uses no waits (5 clock operation) and only 4-byte access; the transfer speed is 80MB/s.

#### <Root bus interface>

This interface is used to access the root bus that carries data between peripheral devices. The number of waits and the number accesses both depend on the target device of the access, but basically accesses are made in units of 1/2/4/32 bytes.

Burst access utilizes the wraparound function. This interface has a 32-byte write buffer (large enough for two single writes or one burst write). Only when there are consecutive single writes do consecutive writes occur on the root bus, making high-speed access possible (but attention must be made to possession of the bus). The maximum transfer speed is 356MB/s (during a burst write).

#### <TA FIFO interface>

This interface is primarily used for transferring polygon data and texture data to the TA FIFO.

This interface supports 32-byte writes only. Wraparound operation is not supported, so access must start from address 0x00. (0x08, 0x10, and 0x18 are not permitted.) The number of waits is dependent on the TA FIFO, and the capacity of the FIFO can be checked in the registers. Furthermore, because there is a possibility that DMA will not be performed properly, writes during ch2-DMA operations are prohibited. The maximum transfer speed is 640MB/s (during a burst write).

For details on the SB block (DMA and other data transfers), refer to section 2.5 and beyond; for details on graphics, refer to section 3.

### § 2.2.3 Initial Settings for the SH4

Tables 2-5 and 2-6 show the settings that are used for the SH4 in the Dreamcast System.

Table 2-5 lists the operation modes of the SH4, which are selected by means of the MD[7:0] pins. These pins are used for other functions, and are sampled internally when the reset condition is released. The same applies to the clock mode, which is set again by software after the system boots up.

Item	Setting
External clock	100MHz (cycle time: 10ns)
Internal clock	200MHz (cycle time: 5ns)
Endian	Little Endian (Intel style)
Area 0 interface	MPX (multiplex)

Table 2-5 SH4 Configuration

Mode Pin	Setting	Operation
MD8	0	Use oscillator.
MD7	1 *	SH4 operates in master mode. *
MD6	0 *	MPX operation is used for area 0. *
MD5	1	All buses are Little Endian format.
MD4	0	64-bit bus width operation
MD3	0	64-bit bus width operation
MD2	1	Clock mode
MD1	0	Clock mode
MD0	1	Clock mode

The configuration of the items marked by an asterisk ("\*") in the table may differ, depending on the SH4 process. (The values indicated in the above table are for the 25μ process.)

Table 2-6 SH4 Operation Mode Settings

The initial settings and notes concerning each of the SH4's functions listed below are shown in the following pages. For details on each of the settings, please refer to the SH4 hardware manual.

The register settings and addresses indicated in this section are the values in the SH4's P4 area. (Caching not permitted; refer to section 2.1.1 and the SH4 manual.)

- Low power consumption mode
- Clock oscillation circuit
- Real-time clock (RTC)
- Time unit (TMU)
- Bus state controller (BSC)
- Direct memory access controller (DMAC)
- Interrupt controller (INTC)

Use of the following SH4 functions, even those that are for debugging only, is prohibited.

- Serial communication interface (SCI)
- Smart card interface
- User break controller (UBC) [for debugging only]
- Hitachi user debugging interface (HITACHI-UDI) [for debugging only]

#### <Low power consumption mode>

The following limitations apply to low power consumption mode.

- (1) Use of standby mode is prohibited, because in that mode the clock is not output and the system hangs.
- (2) Sleep and module standby modes can be used. In addition, because the SCI and RTC are not used, no clock signal is supplied.
- (3) Set the RTC control register 2 (RCR2) before setting the standby control register (STBCR).

The register settings are shown below. (Only valid bits are shown.)

**STBCR** (standby control register) 0xFFC00004 (8 bits) ← 0x03 (initial value: 0x00)

bit7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1

- |                     |       |   |
|---------------------|-------|---|
| <i>bit7 - STBY</i>  | ← '0' | Enters sleep mode in response to the SLEEP instruction.                                   |
| <i>bit6 - PHZ</i>   | ← '0' | Peripheral module-related pins (*1) do not go to high impedance in standby mode.          |
| <i>bit5 - PPU</i>   | ← '0' | Peripheral module-related pins (*2) are pulled up when they are inputs or high impedance. |
| <i>bit4 - MSTP4</i> | ← '0' | Activates the DMAC.   |
| <i>bit3 - MSTP3</i> | ← '0' | As desired (SCIF clock supplied/not supplied)   |
| <i>bit2 - MSTP2</i> | ← '0' | As desired (TMU clock supplied/not supplied)  |
| <i>bit1 - MSTP1</i> | ← '1' | Stops clock supplied to the RTC.  |
| <i>bit0 - MSTP0</i> | ← '1' | Stops clock supplied to the SCI.  |

\*1 MD0/SCK,MD1/TXD2,MD2/RXD2,MD7/TXD,MD8/RTS2,CTS2,DACK0/TDACK,DRAK0/BAVL,DACK1/ID0,DRAK1/ID1

\*2 MD0/SCK,MD1/TXD2,MD2/RXD2,MD7/TXD,MD8/RTS2,SCK2/MRESET,RXD,CTS2,DREQ0/DBREQ,DACK0/TDACK,DRAK0/BAVL,DREQ1/TR,DACK1/ID0,DRAK1/ID1,TCLK

#### <Clock oscillation circuit>

The clock oscillation circuit settings also conform with the MD pin settings in Table 2-6.

- (1) Using an oscillator, not a crystal resonator                      \* MD8= 0
  - (2) Using clock operation mode "5"                                      \* MD2= 1, MD1= 0, MD0= 1
- The detailed settings are as follows:

- |                          |               |
|--------------------------|---------------|
| 1/2 divider:             | OFF           |
| PLL1:                    | ON            |
| PLL2:                    | ON            |
| EXTAL clock input:       | 33MHz         |
| CPU clock:               | 200MHz (× 6)  |
| Bus clock:               | 100MHz (× 3)  |
| Peripheral module clock: | 50MHz (× 3/2) |

- (3) CKIO is clock output
- (4) Watchdog timer mode is prohibited since resets are applied to SH4 only, and not to other chips.

The register settings are shown below. (Only valid bits are shown.)

**FRQCR** (frequency control register): 0xFFC00000 /initial value: 0x0E0A

**With the MD pin settings shown in Table 2-6, it is not necessary to set this register. (The initial settings are adequate.)**

bit15-12	11	10	9	8-6	5-3	2-0
0000	1	1	1	000	001	010

[Bits 15:12 are reserved. (Specify "0x0".)]

*bit11* - CKOEN ← '1' CKIO clock input  
*bit10* - PLL1EN ← '1' Use PLL1  
*bit9* - PLL2EN ← '1' Use PLL2  
*bit8:6* - IFC[2:0] ← '000' CPU clock × 1  
*bit5:3* - BFC[2:0] ← '001' Bus clock × 1/2  
*bit2:0* - PFC[2:0] ← '010' Peripheral clock × 1/4

**WTCNT** (watchdog timer counter): 0xFFC00008 /initial value: 0x0000

bit15-8	7	6	5	4	3	2	1	0
0101 1010	*	*	*	*	*	*	*	*

[Bits 15:8 are reserved. (Specify "0x5A".)]

*bit7:0* Don't care

**WTCSR** (watchdog timer control/status): 0xFFC0000C ← 0xA500/Initial value: 0x0000

bit15-8	7	6	5	4	3	2-0
1010 0101	*	0	*	*	*	***

[Bits 15:8 are reserved. (Specify "0xA5".)]

*bit7* - TME ← Don't care Timer enable  
*bit6* - WT/IT ← '0' Used in interval timer mode  
*bit5* - RSTS ← Don't care Reset type that was generated (Ignored in interval timer mode.)  
  
*bit4* - WOVF ← Don't care Overflow flag (Not set in interval timer mode.)  
*bit3* - IOVF ← Don't care Overflow flag (Used in interval timer mode.)  
*bit2:0* - CKS[2:0] ← Don't care WTCNT clock select (The clock from divider 2 is 200MHz.)

#### <Real-time clock (RTC)>

Use of the RTC is prohibited, and it is necessary to make the setting that stops it. Note that the TCLK pin is an input, and the RTC control register 2 (RCR2) must be set before the standby control register (STBCR) is set.

Including those registers for which access is prohibited, the RTC-related register settings are as listed below.

<b>R6 4CNT</b>	(64Hz counter)	: 0xFFC80000	
			<b>Access prohibited</b>
<b>RSECCNT</b>	(seconds counter)	: 0xFFC80004	<b>Access prohibited</b>
<b>RMINCNT</b>	(minutes counter)	: 0xFFC80008	<b>Access prohibited</b>
<b>RHRCNT</b>	(hours counter)	: 0xFFC8000C	<b>Access prohibited</b>
<b>RWKCNT</b>	(day of the week counter)	: 0xFFC80010	<b>Access prohibited</b>
<b>RDAYCNT</b>	(day counter)	: 0xFFC80014	<b>Access prohibited</b>
<b>RMONCNT</b>	(month counter)	: 0xFFC80018	<b>Access prohibited</b>
<b>RYRCNT</b>	(year counter)	: 0xFFC8001C	<b>Access prohibited</b>
<b>RSECAR</b>	(seconds alarm): 0xFFC80020		<b>Access prohibited</b>
<b>RMINAR</b>	(minutes alarm): 0xFFC80024		<b>Access prohibited</b>
<b>RHRAR</b>	(hours alarm)	: 0xFFC80028	<b>Access prohibited</b>
<b>RWKAR</b>	(day of the week alarm)	: 0xFFC8002C	<b>Access</b>



***RDAYAR*** **prohibited** (day alarm) : 0xFFC80030

***RMONAR*** (month alarm) **Access prohibited** : 0xFFC80034 **Access prohibited**

***RCR1*** (RTC control register 1): 0xFFC80038 (8 bits) ← 0x00 **-- Setting is not required**

bit7	6	5	4	3	2	1	0
*	-	-	0	0	-	-	*

*bit7* - *CF* ← Don't care Carry flag  
*bit4* - *CIE* ← '0' Do not generate carry interrupt  
*bit3* - *AIE* ← '0' Do not generate alarm interrupt  
*bito* - *AF* ← Don't care Alarm flag

***RCR2*** (RTC control register 2): 0xFFC8003C (8 bits) ← 0x00 /Initial value: 0x0001001

bit7	6-4	3	2	1	0
*	000	0	0	0	0

*bit7* - *PEF* ← Don't care Periodic interrupt flag  
*bit6:4* - *PES[2:0]* ← '000' Periodic interrupt generation off  
*bit3* - *RTCEN* ← '0' RTC crystal oscillator stopped  
*bit2* - *ADJ* ← '0' Normal clock operation  
*bit1* - *RESET* ← '0' Normal clock operation  
*bito* - *START* ← '0' Alarm flag

#### <Timer unit (TMU)>

Because the TCLK pin is a pull-up input, the TMU cannot be used with an external clock or the input capture function. Furthermore, the TMU cannot be used with the built-in RTC output clock. The peripheral module clock is 50MHz.

The register settings are as shown below.

***TOCR*** (Timer output control register): 0xFFD80000 ← 0x00 **-- Setting is not required**

bit7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	*

*bito* - *TCOE* ← '0' TCLK pin input

***TSTR*** (Timer start register): 0xFFD80004 /Initial value 0x00

bit7	6	5	4	3	2	1	0
-	-	-	-	-	*	*	*

*bit2* - *STR2* ← Don't care Timer counter 2 on/off  
*bit1* - *STR1* ← Don't care Timer counter 2 on/off  
*bito* - *STRO* ← Don't care Timer counter 2 on/off

***TCORo*** (Timer constant register 0): 0xFFD80008 (32 bits) **-- Set as desired**

***TCNTo*** (Timer counter register 0): 0xFFD8000C (32 bits) **-- Set as desired**

***TCRo*** (Timer control register 0): 0xFFD80010 /Initial value 0x0000

bit15-9	8	7	6	5	4-3	2-0
0000 000	*	-	-	*	00	***

*bit8* - *UNF* ←Don't care Underflow flag  
*bit5* - *UNIE* ←Don't care Underflow interrupt  
*bit4:3* - *CKEG[1:0]* ← '00' Rising edge  
*bit2:0* - *TPSC[2:0]* ←Don't care Timer prescaler (101, 110, and 101 are **prohibited**)

**TCOR1** (Timer constant register 1): 0xFFD80014 (32 bits) -- **Set as desired**

**TCNT1** (Timer counter register 1): 0xFFD80018 (32 bits) -- **Set as desired**

**TCR1** (Timer control register 1): 0xFFD80010C /Initial value 0x0000

bit15-9	8	7	6	5	4-3	2-0
0000 000	*	-	-	*	00	***

*bit8* - *UNF* ←Don't care Underflow flag  
*bit5* - *UNIE* ←Don't care Underflow interrupt  
*bit4:3* - *CKEG[1:0]* ← '00' Rising edge  
*bit2:0* - *TPSC[2:0]* ←Don't care Timer prescaler (101, 110, and 101 are prohibited)

**TCOR2** (Timer constant register 2): 0xFFD80020 (32 bits) -- **Set as desired**

**TCNT2** (Timer counter register 2): 0xFFD80024 (32 bits) -- **Set as desired**

**TCR2** (Timer control register 2): 0xFFD80028 /Initial value 0x0000

bit15-10	9	8	7-6	5	4-3	2-0
0000 00	*	*	00	*	00	***

*bit9* - *ICPF* ←Don't care Input capture interrupt flag  
*bit8* - *UNF* ←Don't care Underflow flag  
*bit7:6* - *ICPE[1:0]* ← '00' Use of input capture prohibited  
*bit5* - *UNIE* ←Don't care Underflow interrupt  
*bit4:3* - *CKEG[1:0]* ← '00' Rising edge  
*bit2:0* - *TPSC[2:0]* ←Don't care Timer prescaler (101, 110, and 101 are prohibited)

**TCPR2** (input capture) 0xFFD8002C **Access prohibited**

### <Bus state controller (BSC)>

The BSC is a register for SH4 external bus-related settings. For details on the main settings, refer to the settings for MD[3:7] in the MD pin settings shown in Table 2-6. The settings for SDRAM, the system memory, are described in section 2.3.2.

The register settings are shown below.

**BCR1** (Bus state control 1): 0xFF800000 ← 0xA3020008 /Initial value: 0xA0000000

bit31	30	29	28-26	25	24	23-22	21	20	19	18	17	16	15	14	13-11	10-8	7-5	4-2	1	0
1	0	1	-	1	1	-	0	0	0	0	0	-	0	0	000	000	000	010	-	0

<i>bit31</i>	- <i>ENDIAN</i>	← '1'	Little Endian
<i>bit30</i>	- <i>MASTER</i>	← '0'	Master
<i>bit29</i>	- <i>AOMPX</i>	← '1'	Area 0 is MPX
<i>bit25</i>	- <i>IPUP</i>	← '1'	Do not pull-up the controller pins (*3)
<i>bit24</i>	- <i>OPUP</i>	← '1'	Do not pull-up the controller pins (*4)
<i>bit21</i>	- <i>A1MBC</i>	← '0'	Area 1 normal
<i>bit20</i>	- <i>A4MBC</i>	← '0'	Area 4 normal
<i>bit19</i>	- <i>BREQEN</i>	← '0'	External request invalid
<i>bit18</i>	- <i>PSHR</i>	← '0'	Master mode
<i>bit17</i>	- <i>MEMMPX</i>	← '1'	Area 1 to 6 MPX
<i>bit15</i>	- <i>HIZMEM</i>	← '0'	High impedance during standby (*5)
<i>bit14</i>	- <i>HIZCNT</i>	← '0'	High impedance during standby or when bus is granted
<i>bit13:11</i>	- <i>A0BST[2:0]</i>	← '000'	Area 0 normal memory
<i>bit10:8</i>	- <i>A5BST[2:0]</i>	← '000'	Area 5 normal memory
<i>bit7:5</i>	- <i>A6BST[2:0]</i>	← '000'	Area 6 normal memory
<i>bit4:2</i>	- <i>DRAMTP[2:0]</i>	← '010'	Area 2 normal memory, area 3 SDRAM
<i>bito</i>	- <i>A56PCM</i>	← '0'	Area 5 6 normal memory

\*3 NMI,IRL[3:0],BREQ,MD6,RDY

\*4 A[25:0],BS,CSn,RD,WEn,RD/WR,RAS,RAS2,CE2A,CE2B,RD2,RD/WR2

\*5 A[25:0],BS,CSn,RD/WR,CE2A,CE2B,RD/WR2

\*6 RAS,RAS2,WEn,RD,RD2

**BCR2** (Bus state control 2): 0xFF800004 ← 0x0000 /Initial value 0x3FFC

bit15-14	13-12	11-10	9-8	7-6	5-4	3-2	1	0
00	00	00	00	00	00	00	-	0

<i>bit15:14</i>	- <i>A0SZ[1:0]</i>	← '00'	Area 0 is 64 bits
<i>bit13:12</i>	- <i>A6SZ[1:0]</i>	← '00'	Area 6 is 64 bits
<i>bit11:10</i>	- <i>A5SZ[1:0]</i>	← '00'	Area 5 is 64 bits
<i>bit9:8</i>	- <i>A4SZ[1:0]</i>	← '00'	Area 4 is 64 bits
<i>bit7:6</i>	- <i>A3SZ[1:0]</i>	← '00'	Area 3 is 64 bits
<i>bit5:4</i>	- <i>A2SZ[1:0]</i>	← '00'	Area 2 is 64 bits
<i>bit3:2</i>	- <i>A1SZ[1:0]</i>	← '00'	Area 1 is 64 bits
<i>bito</i>	- <i>PORTEN</i>	← '0'	Ports D47 to D32 are unused

**WCR1** (Wait control 1): 0xFF800008 ← 0x01110111

/Initial value: 0x77777777

bit31	30-28	27	26-24	23	22-20	19	18-16	15	14-12	11	10-8	7	6-4	3	2-0
-	000	-	001	-	001	-	001	-	000	-	001	-	001	-	001

*bit30:28* - DMAIW[2:0] ← '000' SDRAM is RAS down mode  
*bit26:24* - A6IW[2:0] ← '001' Area 6 - 1 idle cycle between cycles  
*bit22:20* - A5IW[2:0] ← '001' Area 5 - 1 idle cycle between cycles  
*bit18:16* - A4IW[2:0] ← '001' Area 4 - 1 idle cycle between cycles  
*bit14:12* - A3IW[2:0] ← '000' SDRAM is RAS down mode  
*bit10:8* - A2IW[2:0] ← '001' Area 2 - 1 idle cycle between cycles  
*bit6:4* - A1IW[2:0] ← '001' Area 1 - 1 idle cycle between cycles  
*bit2:0* - A0IW[2:0] ← '001' Area 0 - 1 idle cycle between cycles

**WCR2** (Wait control 2): 0xFF80000C ← 0x018060D8

/Initial value 0xFFFEFFFF

bit31-29	28-26	25-23	22-20	19-17	16	15-13	12	11-9	8-6	5-3	2-0
000	000	011	000	000	-	011	-	000	011	011	000

*bit31:29* - A6W[2:0] ← '000' Area 6 Read: 1 data, 1 wait; others, 0 waits  
*bit28:26* - A6B[2:0] ← '000' Area 6 burst pitch = 0  
*bit25:23* - A5W[2:0] ← '011' Area 5 1 data, 3 waits; others, 0 waits  
*bit22:20* - A5B[2:0] ← '000' Area 5 burst pitch = 0  
*bit19:17* - A4W[2:0] ← '000' Area 4 Read: 1 data, 1 wait; others, 0 waits  
*bit15:13* - A3W[2:0] ← '011' SDRAM CAS latency = 3  
*bit11:9* - A2W[2:0] ← '000' Area 2 Read: 1 data, 1 wait; others, 0 waits  
*bit8:6* - A1W[2:0] ← '011' Area 1 1 data, 3 waits; others, 0 waits  
*bit5:3* - A0W[2:0] ← '011' Area 0 1 data, 3 waits; others, 0 waits  
*bit2:0* - A0B[2:0] ← '000' Area 0 burst pitch = 0

**WCR3** (Wait control 3): 0xFF800010 ← 0x07777777 -- **Setting is not required**

bit31-27	26	25-24	23	22	21-20	19	18	17-16	15	14	13-12	11	10	9-8	7	6	5-4	3	2	1-0
-	1	11	-	1	11	-	1	11	-	1	11	-	1	11	-	1	11	-	1	11

*bit26* - A6S0 ← '1' Area 6 Write strobe setup = 1  
*bit25:24* - A6H[1:0] ← '11' Area 6 Data hold = 3  
*bit22* - A5S0 ← '1' Area 5 Write strobe setup = 1  
*bit21:20* - A5H[1:0] ← '11' Area 5 Data hold = 3  
*bit18* - A4S0 ← '1' Area 4 Write strobe setup = 1  
*bit17:16* - A4H[1:0] ← '11' Area 4 Data hold = 3  
*bit14* - A3S0 ← '1' Area 3 Write strobe setup = 1  
*bit13:12* - A3H[1:0] ← '11' Area 3 Data hold = 3  
*bit10* - A2S0 ← '1' Area 2 Write strobe setup = 1  
*bit9:8* - A2H[1:0] ← '11' Area 2 Data hold = 3  
*bit6* - A1S0 ← '1' Area 1 Write strobe setup = 1  
*bit5:4* - A1H[1:0] ← '11' Area 1 Data hold = 3  
*bit2* - A0S0 ← '1' Area 0 Write strobe setup = 1  
*bit1:0* - A0H[1:0] ← '11' Area 0 Data hold = 3

**PCR** (PCMCIA control): 0xFF800018-- **Setting is not required**

Other BSC-related register settings are described in section 2.3.2.

<Direct memory access controller (DMAC)>

The DMAC-related settings are described below.

- (1) DMAC uses DDT mode.
- (2) Because DMA channel 0 is used by the hardware, use by the software is prohibited. (The DMA end interrupt cannot be used.)
- (3) DMA operations are performed on channel 2 as a set with ch2-DMA of the HOLLY chip. (The SH4's DMAC channel 2 register must be controlled by software at the same time.) Channel 2 DMA depends on the following settings.
  - Transfer data length: Only 32-byte block transfer is permitted.
  - Address mode: Only single address mode is permitted.
  - Transfer initiation request: Only external requests (external address space -> external device) are permitted.
  - Bus mode: Only burst mode is permitted.
  - DMA end interrupt: Generation of both SH4:DMAC and HOLLY:ch2-DMA is permitted.  
→ Select one or the other. (If two are enabled, it will just result in interrupts being generated twice.)
- (4) Channels 1 and 3 can be used in the following manner:
  - The allowed transfer data length (8/16/32 bits, 32 bytes) depends on the transfer area. The 64-bit transfer data length specification is permitted only for system memory.
  - Address mode: Only dual address mode is permitted.
  - Transfer initiation request: SCIF interrupt and auto request are permitted.
  - Bus mode: Only cycle steal mode is permitted.
  - DMA end interrupt: Can be used.

The register settings are shown below.

<b>SARo</b> (DMA source address 0): 0xFFFA00000	<b>-- Access prohibited</b>
<b>DARo</b> (DMA destination address 0): 0xFFFA00004	<b>-- Access prohibited</b>
<b>DMATCRo</b> (DMA transfer count 0): 0xFFFA00008	<b>-- Access prohibited</b>
<b>CHCRo</b> (DMA channel control 0): 0xFFFA0000C	<b>-- Access prohibited</b>
<b>SAR1</b> (DMA source address 1): 0xFFFA00010	-- Set as desired
<b>DAR1</b> (DMA destination address 1): 0xFFFA00014	-- Set as desired
<b>DMATCR1</b> (DMA transfer count 10): 0xFFFA00018	-- Set as desired

**CHCR1** CHCR1 (DMA channel control 1): 0xFFA0001C ← 0x00005440 / Initial value: 0x00000000

bit31-29	28	27-25	24	23-20	19	18	17	16	15-14	13-12	11-8	7	6-4	3	2	1	0
000	0	000	0	-	0	0	0	0	**	**	****	0	***	-	*	*	*

<i>bit31:29</i>	- <i>SSA[2:0]</i>	← '000'	No PCMCIA (PCMCIA source address space attributes)
<i>bit28</i>	- <i>STC</i>	← '0'	No PCMCIA (PCMCIA source address wait)
<i>bit27:25</i>	- <i>DSA[2:0]</i>	← '000	No PCMCIA (PCMCIA destination address space attributes)
<i>bit24</i>	- <i>DTC</i>	← '0'	No PCMCIA (PCMCIA destination address wait)
<i>bit19</i>	- <i>DS</i>	← '0'	DREQ low level detection
<i>bit18</i>	- <i>RL</i>	← '0'	DDT mode (DRAK active high)
<i>bit17</i>	- <i>AM</i>	← '0'	DACK output for reads
<i>bit16</i>	- <i>AL</i>	← '0'	DDT mode (DACK active high)
<i>bit15:14</i>	- <i>DM[1:0]</i>	← Don't care	Destination address mode
<i>bit13:12</i>	- <i>SM[1:0]</i>	← Don't care	Source address mode
<i>bit11:8</i>	- <i>RS[3:0]</i>	← Don't care	Resource select - Only 0100, 0101, 0110, 1010, and 1011 can be set
<i>bit7</i>	- <i>TM</i>	← '0'	Cycle steal mode
<i>bit6:4</i>	- <i>TS[2:0]</i>	← Don't care	Transfer size (64-bit transfer data length specification is permitted only for system memory)
<i>bit2</i>	- <i>IE</i>	← Don't care	Interrupt enable
<i>bit1</i>	- <i>TE</i>	← Don't care	Transfer end
<i>bit0</i>	- <i>DE</i>	← Don't care	DMAC enable

**SAR2** (DMA source address 2): 0xFFA00020

This is a system memory (SDRAM) address setting. The address must be a 32-byte boundary address. (Specify "0" for bits 4 through 0.)

**DAR2** (DMA destination address 2): 0xFFA00024    **- -Access prohibited**

**DMATCR2** (DMA transfer count 2): 0xFFA00028

This sets the transfer length, in 32-byte units.

\* It is necessary to set the same transfer amount as the "transfer count" on the HOLLY side. Although the DMAC in the SH4 is set in 32-byte units, the value is set in 1-byte units in the HOLLY.

**CHCR2** (DMA channel control 2): 0xFFA0002C ← 0x000052C0 / Initial value: 0x00000000

bit31-29	28	27-25	24	23-20	19	18	17	16	15-14	13-12	11-8	7	6-4	3	2	1	0
000	0	000	0	-	0	-	0	-	01	**	0010	1	100	-	*	*	*

<i>bit31:29</i>	- <i>SSA[2:0]</i>	← '000'	No PCMCIA (PCMCIA source address space attributes)
<i>bit28</i>	- <i>STC</i>	← '0'	No PCMCIA (PCMCIA source address wait)
<i>bit27:25</i>	- <i>DSA[2:0]</i>	← '000	No PCMCIA (PCMCIA destination address space attributes)
<i>bit24</i>	- <i>DTC</i>	← '0'	No PCMCIA (PCMCIA destination address wait)
<i>bit19</i>	- <i>DS</i>	← '0'	DREQ low level detection
<i>bit17</i>	- <i>AM</i>	← '0'	DACK output for reads
<i>bit15:14</i>	- <i>DM[1:0]</i>	← '01'	DDT mode (destination address increment)
<i>bit13:12</i>	- <i>SM[1:0]</i>	← Don't care	Source address mode
<i>bit11:8</i>	- <i>RS[3:0]</i>	← '0010'	External request (external address space → external device)
<i>bit7</i>	- <i>TM</i>	← '1'	Burst mode
<i>bit6:4</i>	- <i>TS[2:0]</i>	← '100'	32-byte block transfer
<i>bit2</i>	- <i>IE</i>	← Don't care	Interrupt enable
<i>bit1</i>	- <i>TE</i>	← Don't care	Transfer end

*bito* - *DE* ← Don't care DMAC enable

**SAR3** (DMA source address 3): 0xFFA00030 -- Set as desired

**DAR3** (DMA destination address 3): 0xFFA00034 -- Set as desired

**DMATCR3** (DMA transfer count 3): 0xFFA00038 -- Set as desired

**CHCR3** (DMA channel control 3): 0xFFA0003C ← 0x00005440

bit31-29	28	27-25	24	23-20	19	18	17	16	15-14	13-12	11-8	7	6-4	3	2	1	0
000	0	000	0	-	0	-	0	-	**	**	****	0	***	-	*	*	*

*bit31:29* - *SSA[2:0]* ← '000' No PCMCIA (PCMCIA source address space attributes)  
*bit28* - *STC* ← '0' No PCMCIA (PCMCIA source address wait)  
*bit27:25* - *DSA[2:0]* ← '000' No PCMCIA (PCMCIA destination address space attributes)  
*bit24* - *DTC* ← '0' No PCMCIA (PCMCIA destination address wait)  
*bit19* - *DS* ← '0' DREQ low level detection  
*bit17* - *AM* ← '0' DACK output for reads  
*bit15:14* - *DM[1:0]* ← Don't care Destination address mode  
*bit13:12* - *SM[1:0]* ← Don't care Source address mode  
*bit11:8* - *RS[3:0]* ← Don't care Resource select - Only 0100, 0101, 0110, 1010, and 1011 can be set  
*bit7* - *TM* ← '0' Cycle steal mode  
*bit6:4* - *TS[2:0]* ← Don't care Transfer size (64-bit transfer data length specification is permitted only for system memory)  
*bit2* - *IE* ← Don't care Interrupt enable  
*bit1* - *TE* ← Don't care Transfer end  
*bito* - *DE* ← Don't care DMAC enable

**DMAOR** (DMA operation): 0xFFA00040 ← 0x00008201 /Initial value: 0x00000000

bit31-16	15	14	13	12	11	10	9-8	7	6	5	4	3	2	1	0
-	1	-	-	-	-	-	10	-	-	-	-	-	*	*	1

*bit15* - *DDT* ← '1' DDT mode  
*bit9:8* - *PR[1:0]* ← '10' ch2 priority  
*bit2* - *AE* ← Don't care Address error flag  
*bit1* - *NMIF* ← Don't care NMI flag  
*bito* - *DME* ← '1' DMAC enable

#### <Serial communication interface with built-in FIFO (SCIF)>

There are no particular initial settings for the SCIF.

#### <I/O ports>

The I/O port settings are undefined.

#### <Interrupt Controller (INTC)>

The INTC settings are as described below.

- (1) NMI interrupts (falling edge detection; the NMI pin is pulled up) are for debugging purposes only, and are not supported in the release version (MP).
- (2) Only the IRL1 and 2 interrupts are used (pull up IRL0 and 3), and these interrupts are used as level encoding interrupts. The interrupt levels are "2" (IRL3:0 = 1101), "4" (IRL3:0 = 1011), and "6" (IRL3:0 = 1001).
- (3) The following interrupts are not generated:
 

TMU2/TICPI2:	Input capture interrupt
RTC/ATI:	Alarm interrupt
RTC/PRI:	Cycle interrupt
RTC/CUI:	Carry interrupt
SCI/ERI:	Reception error interrupt
SCI/RXI:	Reception data full interrupt
SCI/TXI:	Transmission data empty interrupt
SCI/TEI:	Transmission end interrupt
REF/RCMI:	Compare match interrupt
DMAC/DMTEO:	DMAC-cho transfer end interrupt
- (4) The Hitachi-UDI interrupt is for debugging only.

The register settings are as shown below.

**IPRA** (interrupt priority level setting register A): 0xFFD00004 /Initial value: 0x00000000

bit15-12	11-8	7-4	3-0
****	****	****	0000

<i>bit15:12</i> - <i>TMU0</i> <i>bit11:8</i> - <i>TMU1</i> <i>bit7:4</i> - <i>TMU2</i> <i>bit3:0</i> - <i>RTC</i>	←Don't care ←Don't care ←Don't care ← '0000'	TMU0 interrupt request (select from among interrupt levels 1101, 1011, and 1001) TMU1 interrupt request (same as above) TMU2 interrupt request (same as above) RTC interrupt request mask (same as above)
--	---	--

**IPRB** (interrupt priority level setting register B): 0xFFD00008 /Initial value: 0x00000000

bit15-12	11-8	7-4	3-0
****	****	0000	-

<i>bit15:12</i> - <i>WDT</i> <i>bit11:8</i> - <i>REF</i> <i>bit7:4</i> - <i>SCI</i>	←Don't care ←Don't care ← '0000'	WDT interrupt request (select from among interrupt levels 1101, 1011, and 1001) REF interrupt request (same as above) SCI interrupt request mask (same as above)
---	--	--

**IPRC** (interrupt priority level setting register C): 0xFFD0000C /Initial value: 0x00000000

bit15-12	11-8	7-4	3-0
-	****	****	****

<i>bit11:8</i> - <i>DMAC</i> <i>bit7:4</i> - <i>SCIF</i>	←Don't care ←Don't care	DMAC interrupt request (select from among interrupt levels 1101, 1011, and 1001) SCIF interrupt request (same as above)
---	----------------------------	--



*bit3:0* - *UDI*      ←Don't care      Hitachi-UDI interrupt request (same as above)

**ICR** (interrupt control register): 0xFFD00000      /Initial value: 0x00000000

bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*_	-	-	-	-	-	*	0	0	-	-	-	-	-	-	-

*bit15* - *NMIL*      ←Don't care      NMI input level  
*bit9* - *NMIB*      ←Don't care      NMI block mode  
*bit8* - *NMIE*      ← '0'      NMI is detected at falling edge  
*bit7* - *IRLM*      ← '0'      IRL interrupts are level encoded interrupts

## § 2.3 System Memory

### § 2.3.1 System Memory Configuration and Control

System memory, which is the main memory in the Dreamcast System, is connected directly to the SH4 (the CPU), and is used by the SH4 to store program code and as work memory. The memory size that will be supported depends on the cost of memory. The specifications for the system memory are shown in Table 2-7. The base configuration is 2 x 64Mbit SDRAMs, which provides a 16MB storage capacity.

Memory size	16MB
Technology	2 × 64Mbit SDRAMs (2 banks × 1024K words × 32 bits)
Total bus width	64 bit
Burst sequence	Sequential
1-chip bus width	32 bit
Operating frequency	100MHz
Peak BBW	800MB/s

Table 2-7 Base Specifications for System Memory

The SDRAM is controlled directly by the SH4's internal BSC (Bus State Controller). The BSC controls all of the SDRAM control signals, and also handles the refresh and precharge operations.

The SDRAM must be set prior to being accessed immediately after the power is applied. The SH4 generates all configuration cycles through software. The configuration cycles are generated by writing to the SH4 registers.

Only the SH4 can be the master for an SDRAM access; an access from the HOLLY chip to SDRAM can only be performed by using the SH4 DMA cycle. In addition, it is possible for one external device to become the logical bus master through the SH4's DDT interface.

The On Demand Data Transfer Mode (DDT) protocol can be used for channel 0. An external device can program the SH4's DMAC channel 0 through DDT. This approach can be used by HOLLY to efficiently access the main memory SDRAM.

### § 2.3.2 System Memory Initial Settings

The initial settings for system memory are identified below.

- Use burst length = 4, wrap type = sequential, CAS latency = 3
- RAS down mode.
- Self-refresh mode may not be used.
- Set the SDRAM refresh interval to 15040nsec.

The register settings are shown below.

**MCR** (individual memory control): 0xFF800014 ← 0xC0121214 /**Dev.Box** memory 32M  
 ← 0xC0091224 /**Dev.Box**/MP  
 (mass production version) memory  
 16M

bit31	30	29-27	26-24	23	22	21-19	18	17-16	15-13	12-10	9	8-7	6	5-3	2	1	0
1	*	000	-	0	-	TPC	-	RCD	000	100	1	00	0	AMX	1	0	0

bit31 - RASD ← '1' RAS down mode  
 bit30 - MRSET ← Don't care setting  
 bit29:27 - TRC[2:0] ← '000' RAS precharge = 0 after refresh  
 bit23 - TCAS ← '0' CAS negate = 1  
 bit21:19 - TPC[2:0] ← '010' PRE-RAS = 3 -- **Dev.Box** memory 32M --  
 '001' PRE-RAS = 2 -- **Dev.Box**/MP memory 16M --  
 bit17:16 - RCD[1:0] ← '10' PRE-CAS = 3 -- **Dev.Box** memory 32M --  
 '01' PRE-CAS = 2 -- **Dev.Box**/MP memory 16M --  
 bit15:13 - TRWL[2:0] ← '000' Write precharge = 1  
 bit12:10 - TRAS[2:0] ← '100' After refresh, command interval = 8 + TRC  
 bit9 - BE ← '1' DRAM burst ("1" due to RAS down)  
 bit8:7 - SZ[1:0] ← '00' SDRAM 64bit  
 bit6 - AMXEXT ← '0' Bank address normal  
 bit5:3 - AMX[2:0] ← '010' 64Mbit, 16-bit bus, 2 banks × 4 -- **Dev.Box** memory 32M --  
 '100' 64Mbit, 32-bit bus, 4 banks × 2 -- **Dev.Box**/MP memory 16M --  
 bit2 - RFSH ← '1' Perform refresh  
 bit1 - RMODE ← '0' CAS-before-RAS refresh (self-refresh may not be used)  
 bit0 - EDOMODE ← '0' "0" due to SDRAM

**SDMR** (SDRAM mode): 0xFF940190 ← 0xFF (MCR-MRSET must also be set at the same time)

bit15-10	9-7	6	5-3	2-0
(000000)	011	0	010	(000)

\*This register is specified by a byte write to write address [0xFF940000 + X]. Either specify the "0" for the other bits in the setting X, or the contents of the data in the byte write do not matter.

bit9:7 - LTMODE ← '011' CAS latency = 3  
 bit6 - WT ← '0' Wrap type = Sequential  
 bit5:3 - BL ← '010' Burst Length=4

**RTCSR** (refresh timer control/status): 0xFF80001C ← 0xA510 /Initial value: 0x0000

bit15-8	7	6	5-3	2	1	0
(1010 0101)	*	0	010	*	*	*

bit7 - CMF ← Don't care Compare match flag  
 bit6 - CMIE ← '0' Compare match interrupt disabled  
 bit5:3 - CKS[2:0] ← '010' Clock = CKIO/16 = 160nsec  
 bit2 - OVf ← Don't care Refresh count overflow flag  
 bit1 - OVIE ← Don't care Refresh count overflow interrupt  
 bit0 - LMTS ← Don't care Refresh count overflow limit

**RTCNT** (Refresh timer counter): 0xFF800020 ← 0xA500 /Initial value 0x0000

bit15-8	7	6	5	4	3	2	1	0
(1010 0101)	0	0	0	0	0	0	0	0

**RTCOR** (Refresh time constant): 0xFF800024 ← 0xA55E /Initial value 0x0000

bit15-8	7	6	5	4	3	2	1	0
(1010 0101)	0	1	0	1	1	1	1	0

( bits 15:8                      Specify 0xA5.)  
  bit7:0                        Specify 0x5E. (0x5E = 94...10nsec \* 16 \* 94 = 15040nsec)

**RFCR** (refresh count): 0xFF800028 ← Don't care /Initial value: 0x0000

bit15-9	8	7	6	5	4	3	2	1	0
(1010 010)	*	*	*	*	*	*	*	*	*

(            bit15:9                      Specify 1010010.)

### § 2.3.3 Access Procedure

In order to use the system memory (SDRAM), it is necessary to set the mode first immediately after power on; after setting the BSC-related registers, write the SDRAM mode register. (Refer to SDMR.)

## § 2.4 Register Map

The register map for the System Bus block is shown below. Shaded items are software debugging registers.

Address	Name	R/W	Description
0x005F 6800	SB_C2DSTAT	RW	ch2-DMA destination address
0x005F 6804	SB_C2DLEN	RW	ch2-DMA length
0x005F 6808	SB_C2DST	RW	ch2-DMA start
0x005F 6810	SB_SDSTAW	RW	Sort-DMA start link table address
0x005F 6814	SB_SDBAAW	RW	Sort-DMA link base address
0x005F 6818	SB_SDWLT	RW	Sort-DMA link address bit width
0x005F 681C	SB_SDLAS	RW	Sort-DMA link address shift control
0x005F 6820	SB_SDST	RW	Sort-DMA start
0x005F 6840	SB_DBREQM	RW	DBREQ# signal mask control
0x005F 6844	SB_BAVLWC	RW	BAVL# signal wait count
0x005F 6848	SB_C2DPRYC	RW	DMA (TA/Root Bus) priority count
0x005F 684C	SB_C2DMAXL	RW	ch2-DMA maximum burst length
0x005F 6880	SB_TFREM	R	TA FIFO remaining amount
0x005F 6884	SB_LMMODE0	RW	Via TA texture memory bus select 0
0x005F 6888	SB_LMMODE1	RW	Via TA texture memory bus select 1
0x005F 688C	SB_FFST	R	FIFO status
0x005F 6890	SB_SFRES	W	System reset
0x005F 689C	SB_SBREV	R	System bus revision number
0x005F 68A0	SB_RBSPLT	RW	SH4 Root Bus split enable
0x005F 6900	SB_ISTNRM	RW	Normal interrupt status
0x005F 6904	SB_ISTEXT	R	External interrupt status
0x005F 6908	SB_ISTERR	RW	Error interrupt status
0x005F 6910	SB_IML2NRM	RW	Level 2 normal interrupt mask
0x005F 6914	SB_IML2EXT	RW	Level 2 external interrupt mask
0x005F 6918	SB_IML2ERR	RW	Level 2 error interrupt mask
0x005F 6920	SB_IML4NRM	RW	Level 4 normal interrupt mask
0x005F 6924	SB_IML4EXT	RW	Level 4 external interrupt mask
0x005F 6928	SB_IML4ERR	RW	Level 4 error interrupt mask
0x005F 6930	SB_IML6NRM	RW	Level 6 normal interrupt mask
0x005F 6934	SB_IML6EXT	RW	Level 6 external interrupt mask
0x005F 6938	SB_IML6ERR	RW	Level 6 error interrupt mask
0x005F 6940	SB_PDTNRM	RW	Normal interrupt PVR-DMA startup mask
0x005F 6944	SB_PDTEXT	RW	External interrupt PVR-DMA startup mask
0x005F 6950	SB_G2DTNRM	RW	Normal interrupt G2-DMA startup mask
0x005F 6954	SB_G2DTEXT	RW	External interrupt G2-DMA startup mask

<Note> RW: Read/write; R: Read only; W: Write only

Address	Name	R/W	Description
0x005F 6C04	SB_MDSTAR	RW	Maple-DMA command table address
0x005F 6C10	SB_MDTSEL	RW	Maple-DMA trigger select

0x005F 6C14	SB_MDEN	RW	Maple-DMA enable
0x005F 6C18	SB_MDST	RW	Maple-DMA start
0x005F 6C80	SB_MSYS	RW	Maple system control
0x005F 6C84	SB_MST	R	Maple status
0x005F 6C88	SB_MSHTCL	W	Maple-DMA hard trigger clear
0x005F 6C8C	SB_MDAPRO	W	Maple-DMA address range
0x005F 6CE8	SB_MMSEL	RW	Maple MSB selection
0x005F 6CF4	SB_MTXDAD	R	Maple Txd address counter
0x005F 6CF8	SB_MRXDAD	R	Maple Rxd address counter
0x005F 6CFC	SB_MRXDBD	R	Maple Rxd base address
0x005F 7404	SB_GDSTAR	RW	GD-DMA start address
0x005F 7408	SB_GDLEN	RW	GD-DMA length
0x005F 740C	SB_GDDIR	RW	GD-DMA direction
0x005F 7414	SB_GDEN	RW	GD-DMA enable
0x005F 7418	SB_GDST	RW	GD-DMA start
0x005F 7480	SB_G1RRC	W	System ROM read access timing
0x005F 7484	SB_G1RWC	W	System ROM write access timing
0x005F 7488	SB_G1FRC	W	Flash ROM read access timing
0x005F 748C	SB_G1FWC	W	Flash ROM write access timing
0x005F 7490	SB_G1CRC	W	GD PIO read access timing
0x005F 7494	SB_G1CWC	W	GD PIO write access timing
0x005F 74A0	SB_G1GDRC	W	GD-DMA read access timing
0x005F 74A4	SB_G1GDWC	W	GD-DMA write access timing
0x005F 74B0	SB_G1SYSM	R	System mode
0x005F 74B4	SB_G1CRDYC	W	G1IORDY signal control
0x005F 74B8	SB_GDAPRO	W	GD-DMA address range
0x005F 74F4	SB_GDSTARD	R	GD-DMA address count (on Root Bus)
0x005F 74F8	SB_GDLEND	R	GD-DMA transfer counter
0x005F 7800	SB_ADSTAG	RW	AICA:G2-DMA G2 start address
0x005F 7804	SB_ADSTAR	RW	AICA:G2-DMA system memory start address
0x005F 7808	SB_ADLEN	RW	AICA:G2-DMA length
0x005F 780C	SB_ADDIR	RW	AICA:G2-DMA direction
0x005F 7810	SB_ADTSEL	RW	AICA:G2-DMA trigger select
0x005F 7814	SB_ADEN	RW	AICA:G2-DMA enable

<Note> RW: Read/write; R: Read only; W: Write only

Address	Name	R/W	Description
0x005F 7818	SB_ADST	RW	AICA:G2-DMA start
0x005F 781C	SB_ADSUSP	RW	AICA:G2-DMA suspend
0x005F 7820	SB_E1STAG	RW	Ext1:G2-DMA G2 start address
0x005F 7824	SB_E1STAR	RW	Ext1:G2-DMA system memory start address
0x005F 7828	SB_E1LEN	RW	Ext1:G2-DMA length

0x005F 782C	SB_E1DIR	RW	Ext1:G2-DMA direction
0x005F 7830	SB_E1TSEL	RW	Ext1:G2-DMA trigger select
0x005F 7834	SB_E1EN	RW	Ext1:G2-DMA enable
0x005F 7838	SB_E1ST	RW	Ext1:G2-DMA start
0x005F 783C	SB_E1SUSP	RW	Ext1: G2-DMA suspend
0x005F 7840	SB_E2STAG	RW	Ext2:G2-DMA G2 start address
0x005F 7844	SB_E2STAR	RW	Ext2:G2-DMA system memory start address
0x005F 7848	SB_E2LEN	RW	Ext2:G2-DMA length
0x005F 784C	SB_E2DIR	RW	Ext2:G2-DMA direction
0x005F 7850	SB_E2TSEL	RW	Ext2:G2-DMA trigger select
0x005F 7854	SB_E2EN	RW	Ext2:G2-DMA enable
0x005F 7858	SB_E2ST	RW	Ext2:G2-DMA start
0x005F 785C	SB_E2SUSP	RW	Ext2: G2-DMA suspend
0x005F 7860	SB_DDSTAG	RW	Dev:G2-DMA G2 start address
0x005F 7864	SB_DDSTAR	RW	Dev:G2-DMA system memory start address
0x005F 7868	SB_DDLLEN	RW	Dev:G2-DMA length
0x005F 786C	SB_DDDIR	RW	Dev:G2-DMA direction
0x005F 7870	SB_DDTSEL	RW	Dev:G2-DMA trigger select
0x005F 7874	SB_DDEN	RW	Dev:G2-DMA enable
0x005F 7878	SB_DDST	RW	Dev:G2-DMA start
0x005F 787C	SB_DDUSP	RW	Dev: G2-DMA suspend
0x005F 7880	SB_G2ID	R	G2 bus version
0x005F 7890	SB_G2DSTO	RW	G2/DS timeout
0x005F 7894	SB_G2TRTO	RW	G2/TR timeout
0x005F 7898	SB_G2MDMTO	RW	Modem unit wait timeout
0x005F 789C	SB_G2MDMW	RW	Modem unit wait time
0x005F 78BC	SB_G2APRO	W	G2-DMA address range
0x005F 78C0	SB_ADSTAGD	R	AICA-DMA address counter (on AICA)
0x005F 78C4	SB_ADSTARD	R	AICA-DMA address counter (on root bus)
0x005F 78C8	SB_ADLEND	R	AICA-DMA transfer counter
0x005F 78D0	SB_E1STAGD	R	Ext-DMA1 address counter (on Ext)
0x005F 78D4	SB_E1STARD	R	Ext-DMA1 address counter (on root bus)
0x005F 78D8	SB_E1LEND	R	Ext-DMA1 transfer counter

<Note> RW: Read/write; R: Read only; W: Write only

Address	Name	R/W	Description
0x005F 78E0	SB_E2STAGD	R	Ext-DMA2 address counter (on Ext)
0x005F 78E4	SB_E2STARD	R	Ext-DMA2 address counter (on root bus)
0x005F 78E8	SB_E2LEND	R	Ext-DMA2 transfer counter
0x005F 78F0	SB_DDSTAGD	R	Dev-DMA address counter (on Ext)
0x005F 78F4	SB_DDSTARD	R	Dev-DMA address counter (on root bus)
0x005F 78F8	SB_DDLLEN	R	Dev-DMA transfer counter
0x005F 7C00	SB_PDSTAP	RW	PVR-DMA PVR start address
0x005F 7C04	SB_PDSTAR	RW	PVR-DMA system memory start address
0x005F 7C08	SB_PDLEN	RW	PVR-DMA length
0x005F 7C0C	SB_PDDIR	RW	PVR-DMA direction

---

0x005F 7C10	SB_PDTSEL	RW	PVR-DMA trigger select
0x005F 7C14	SB_PDEN	RW	PVR-DMA enable
0x005F 7C18	SB_PDST	RW	PVR-DMA start
0x005F 7C80	SB_PDAPRO	W	PVR-DMA address range
0x005F 7CF0	SB_PDSTAPD	R	PVR-DMA address counter (on Ext)
0x005F 7CF4	SB_PDSTARD	R	PVR-DMA address counter (on root bus)
0x005F 7CF8	SB_PDLEND	R	PVR-DMA transfer counter

<Note> RW: Read/write; R: Read only; W: Write only

Table 2-9



## **§ 2.5 Single Access to Each Block**

The devices that can be accessed from SH4 are shown in the memory map in section 2.1, "System Mapping." The areas that can be read/written, and the accessible sizes are also the same.

## § 2.6 DMA Transfers

### § 2.6.1 Overview of DMA Transfers

There are two basic types of DMA in this system. There is write-only DMA, which can transfer texture data or display lists (polygon parameters) quickly from system memory to texture memory via the TA Bus of the HOLY internal block bus, and DMA for transfers via the Root Bus.

In addition, there are two types of DMA that use the TA Bus: ch2-DMA and Sort-DMA. ch2-DMA is used to transfer texture data and display lists. Sort-DMA is used to presort display lists in the CPU and then transfer the data in accordance with that list.

There are six types of DMA that use the Root Bus: PVR-DMA, GD-DMA, AICA-DMA, Ext-DMA1, Ext-DMA2, and Maple-DMA. 32 bytes can be transferred in one DMA transfer operation.

The use of each type of DMA is described below.

PVR-DMA is used to overwrite palette RAM, etc., in the CORE from system memory. GD-DMA is used to transfer data from the GD-ROM to system memory or to wave memory (AICA Memory). AICA-DMA (wave DMA) is used to transfer wave data from system memory to wave memory. Ext-DMA1 and 2 are DMA for devices connected to the G2 Bus. (At present, there is no particular use for these types of DMA.) Maple-DMA is used to read commands from system memory, and to write data from a control pad, etc., into system memory.

In addition, even if TA Bus DMA and all six types of Root Bus DMA have been initiated, the CPU can still freely access those areas which it is normally permitted to access. However, because all DMAs steal cycles, it is always necessary to check the DMA end interrupt. A DMA transfer can end either normally or abnormally; the status is reflected in the interrupt register.

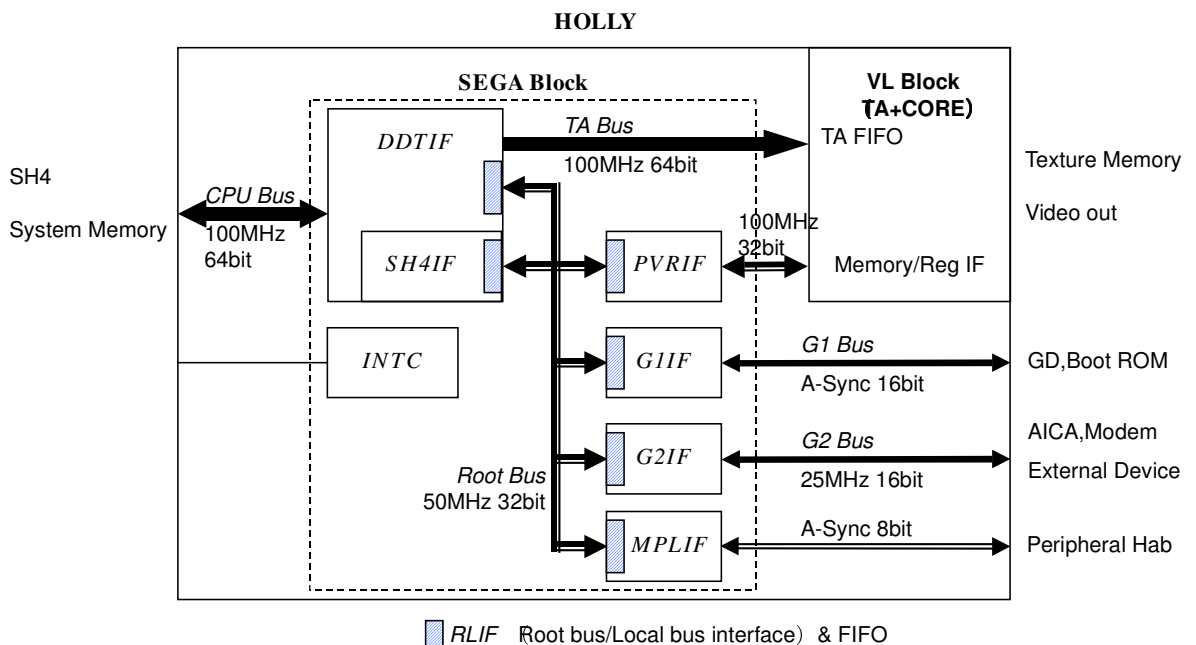


Fig. 2-1

### § 2.6.2 Types of DMA

The DMA types are shown in the table below.

No.	Name of DMA	Use
1	GD Data DMA1	Downloads programs and data from GD-ROM to system memory.
2	GD Data DMA2	Transfers waveform data from GD-ROM to wave memory.
3	Texture DMA	Large, high-speed texture transfer to texture memory. Direct texture transfer from system memory → TA → texture memory.
4	Display list DMA	List transfer of one million polygons from system memory
5	Wave Data DMA	Transfers data from system memory to wave memory.
6	ARM Data DMA	Transfers programs and data from system memory to ARM (sound processor).
7	Peripheral DMA1	Reads the status (pressed or not) of the game pad buttons, etc., into system memory.
8	Peripheral DMA2	Reads the status (pressed or not) of the game pad buttons, etc., into system memory.
9	Color Palette DMA	Block transfers the color palette from system memory.
10	External Area DMA1	Transfers data from an external area, such as a development (debugging) tool, to system memory.
11	External Area DMA2	Transfers data from system memory to an external area, such as a development (debugging) tool.

Table 2-10

\* All of the types of DMA listed in the table above are explained in detail in the sections that follow.

### § 2.6.3 GD-ROM Data Transfers

This section explains the register settings and GD-ROM drive settings that are needed in order to use a DMA transfer to transfer data from the GD-ROM to an area in system memory, texture memory, or wave memory.

(1) SB\_GDAPRO (0x005F74B8) register setting

If the transfer destination is system memory, set System Memory Protection. The upper 16 bits contain the Protection Code "0x8843." 7 bits (bits 14 - 8) out of the lower 16 bits indicate the start address for which transfer is enabled, while another 7 bits (bits 6 - 0) indicate the end address for which transfer is enabled. Each of these groups of 7 bits corresponds to the address bits A26 - A20.

To enable transfer to 0x0C000000 through 0xFFFFFFFF, write 0x8843407F to the register.

To enable transfer to 0xFF000000 through 0xFFFFFFFF, write 0x88437F7F to the register.

To enable transfer to 0xD4000000 through 0xD7FFFFFF, write 0x88435457 to the register.

(2) SB\_G1GDRC (0x005F74A0) register setting

Set the access wait value when reading by a DMA.

Write 0x00001001, which is equivalent to "Multi Word-DMA Mode 2."

(3) SB\_GDSTAR (0x005F7404) setting

Set the transfer start address for the transfer destination (the SH4 address).

(4) SB\_GDLEN (0x005F7408) setting

Specify the number of bytes to be transferred, in units of "0x20".

If an excess results when the amount of data that is to be sent is specified in units of 0x20 bytes, the data that is to be sent is padded with zeroes.

(5) SB\_GDDIR (0x005F740C)

Specify the transfer direction. Write a "1" (GD-ROM → system memory, etc.)

(6) SB\_GDEN (0x005F7414)

Specify "1" for DMA enable.

(7) GD-ROM drive (0x005F7000 - 0x005F70FF) settings

Set the register for the GD-ROM drive.

For details on the settings, refer to the GD-ROM protocol specifications.

(8) SB\_GDST (0x005F7418) settings

DMA starts when the SB\_GDEN register is set to "1" and then a "1" is written to this register (SB\_GDST).

This register also functions as the DMA status register. (0: DMA stopped; 1: DMA in progress)

Example: DMA transfer of 32,768 bytes (16 sectors) from GD-ROM to 0x0C000000 in system memory

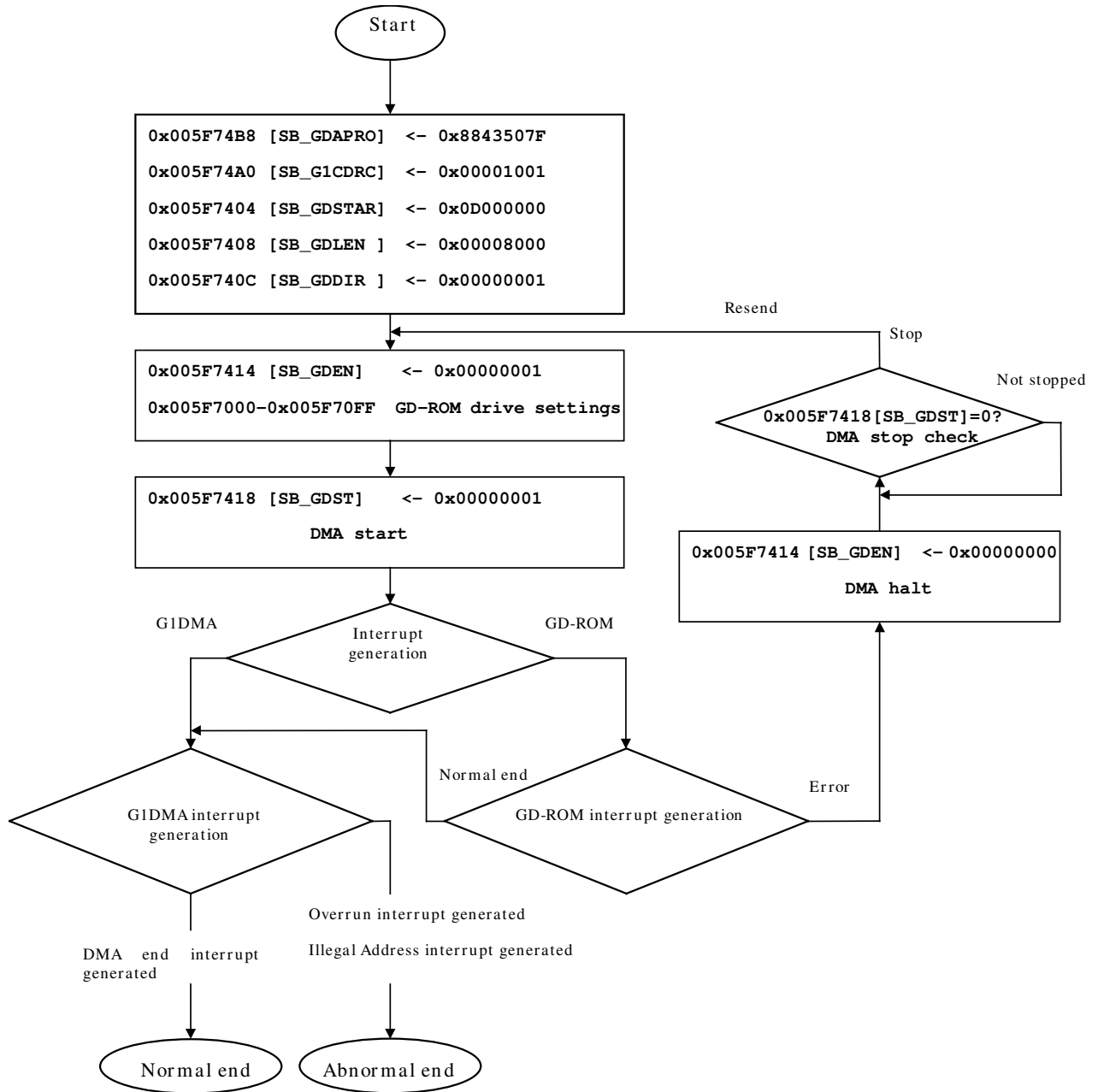


Fig. 2-2

Interrupts from the GD-ROM are allocated to bit 0 of the SB\_ISTEXT (0x005F6904) register. An interrupt is generated if an error of some sort occurred in the GD-ROM drive, or if the transfer ended normally. The status is determined by the Status register in the GD-ROM drive.

Regarding the end of DMA: if DMA ends normally, bit 14 of the SB\_ISTNRM (0x005F6900) register is set to "1" and, if the interrupt mask is not in effect, an interrupt is generated.

In addition, the SB\_GDST register indicates the DMA status; when DMA ends, the value of this register returns to "0." In this case, the value in the SB\_GDEN register remains "1."

Regarding DMA errors: if the DMA address for the transfer destination moves beyond the allowable memory range during a DMA operation, an overrun interrupt is generated and that DMA operation is forcibly terminated. In this case, bit 13 of the SB\_ISTERR (0x005F6908) register is set to "1." Furthermore, if the transfer destination address was incorrectly set outside of the allowable memory range, an illegal address interrupt is generated and bit 12 of the SB\_ISTERR register is set to "1." These interrupts are generated both when the incorrect DMA address is set, and when an attempt is made to initiate DMA with such an incorrect DMA address.

Note that when these errors are generated, the SB\_GDEN register is set to "0."

Cautions during DMA operations: If the SB\_GDAPRO, SB\_G1GDRC, SB\_GDSTAR, SB\_GDLEN, or SB\_GDDIR register is overwritten while a DMA operation is in progress, the new setting has no effect on the current DMA operation. Once the current DMA is terminated and the next DMA is initiated (SB\_GDEN = 1 and SB\_GDST = 1), the values in these five registers are retrieved. A DMA operation that is currently in progress can be forcibly terminated by writing a "0" in the SB\_GDEN register. If an access is in progress when this happens, the value in the SB\_GDST register returns to "0" as soon as the access terminates.

Note that system ROM and flash memory cannot be accessed while a DMA operation is in progress. If a write access is attempted, it is invalid, and if a read access is attempted, the value "0x00" is returned. In this case, bit 14 of the SB\_ISTERR (0x005F6908) register is set to "1." This error has no effect on DMA operations, but it is essential to realize that the access to system ROM or flash memory that was performed is invalid.

## § 2.6.4 Texture Data Transfers

There are two types of texture data transfers: direct texture transfers and YUV texture transfers.

### § 2.6.4.1 Direct Texture Transfers

ch2-DMA\* (explained at the end of this section) is used to conduct DMA transfers of direct textures. The setup procedure is described below.

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set the transfer source address in the SH4-DMAC-SAR2 register.
- (4) Set the size of the transfer (the number of bytes to be transferred/32) in the SH4-DMAC-DMATCR2 register.
- (5) Make the operation settings in the SH4-DMAC-CHCR2 register. When doing so, set the DE bit to "1."
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1." DMA cannot be initiated if the AE, NMIF, and DME bits do not all meet this condition. Furthermore, if the DDT bit is "0," the DMA operation will be performed incorrectly.
- (7) If the address that is set in the SB\_C2DSTAT register is within the range from 0x11000000 to 0x11FFFFE0, set 0x00000000 in the SB\_LMMODE0 register.
- (8) If the address that is set in the SB\_C2DSTAT register is within the range from 0x13000000 to 0x13FFFFE0, set 0x00000000 in the SB\_LMMODE1 register.
- (9) Set the transfer destination address in the SB\_C2DSTAT register.
- (10) Set the transfer size (the number of bytes) in the SB\_C2DLEN register.
- (11) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

The following prohibitions apply during a direct texture DMA transfer:

- Because a direct texture DMA transfer is performed using ch2-DMA, other DMA operations that use ch2-DMA cannot be performed at the same time.
- Never write to any of the registers that are used in a direct texture DMA transfer. The only exception is writing 0x00000000 to the SB\_C2DST register in order to interrupt the DMA transfer.
- The CPU must not perform a burst write to addresses 0x10000000 to 0x13FFFFE0. Doing so could result in the loss of some data in the DMA transfer.

The status of each register when a direct texture DMA transfer ends normally is described below:

- The SH4\_DMAC\_SAR2 register points to the address that follows the location at which the transfer ended.
- The value in the SH4\_DMAC\_DMATCR2 register is 0x00000000.
- The TE bit of the SH4\_DMAC\_CHCR2 register is "1."
- The SB\_C2DSTAT register points to the address that follows the location at which the transfer ended.
- The value in the SB\_C2DLEN register is 0x00000000.
- The value in the SB\_C2DST register is 0x00000000.
- The DMA end interrupt flag (SB\_ISTNRM - bit 19: DTDE2INT) is set to "1."

An example of how to use direct texture DMA transfer is provided below.

Transferring texture data (0x00004000 bytes) from system memory to texture memory

System memory addresses:	0C600000 to 0xC603FFF
TA addresses:	0x11400000 to 0x11403FFF
Texture memory addresses:	x00400000 to 0x00403FFF

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC\_CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set 0xC600000 in the SH4-DMAC\_SAR2 register.
- (4) Set 0x00000200 in the SH4-DMAC\_DMATCR2 register.
- (5) Set 0x000012C1 in the SH4-DMAC\_CHCR2 register.
- (6) Read the SH4-DMAC\_DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1."
- (7) Because the address that is set in the SB\_C2DSTAT register is within the range from 0x11000000 to 0x11FFFFE0, set 0x00000000 in the SB\_LMMODE0 register.
- (8) Because the address that is set in the SB\_C2DSTAT register is not within the range from 0x13000000 to 0x13FFFFE0, do not set the SB\_LMMODE1 register.
- (9) Set 0x11400000 in the SB\_C2DSTAT register.
- (10) Set 0x00004000 in the SB\_C2DLEN register.
- (11) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.



-- (Supplement) About ch2-DMA --

ch2-DMA permits fast data transfer from system memory to texture memory. ch2-DMA cannot be used in the reverse direction, for transfers from texture memory to system memory.

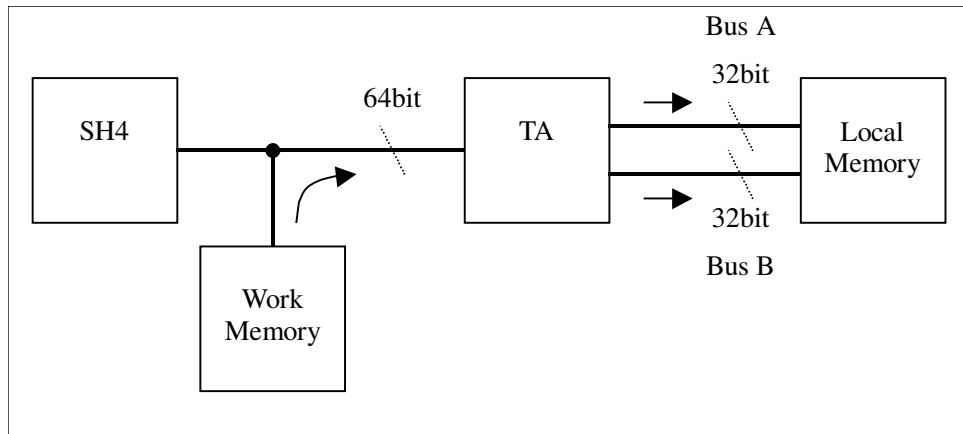


Fig. 2-3 ch2-DMA Transfer Path

There are three possible types of ch2-DMA transfers: transfer to the TA Converter, transfer to the YUV Converter, and direct transfer to texture memory. The transfer to the TA Converter uses the TA function, so this type is used to transfer polygon parameters. The transfer to the YUV Converter is used to transfer YUV texture data. Direct transfer to texture memory is used for direct texture data transfers because it transfers the contents of system memory to texture memory without converting the data. In addition, the bus width for transfers to texture memory can be selected as either 64 bits or 32 bits.

Which of these types of transfers is to be used is determined by the address that is set in the SB\_C2DSTAT register.

The following is a list of the HOLLY registers that are used for ch2-DMA. (For details, refer to section 8.4.1, "System Bus Register.")

<b>SB_C2DSTAT</b>	(0x005F6800)
<b>SB_C2DLEN</b>	(0x005F6804)
<b>SB_C2DST</b>	(0x005F6808)
<b>SB_LMMODE0</b>	(0x005F6884)
<b>SB_LMMODE1</b>	(0x005F6888)

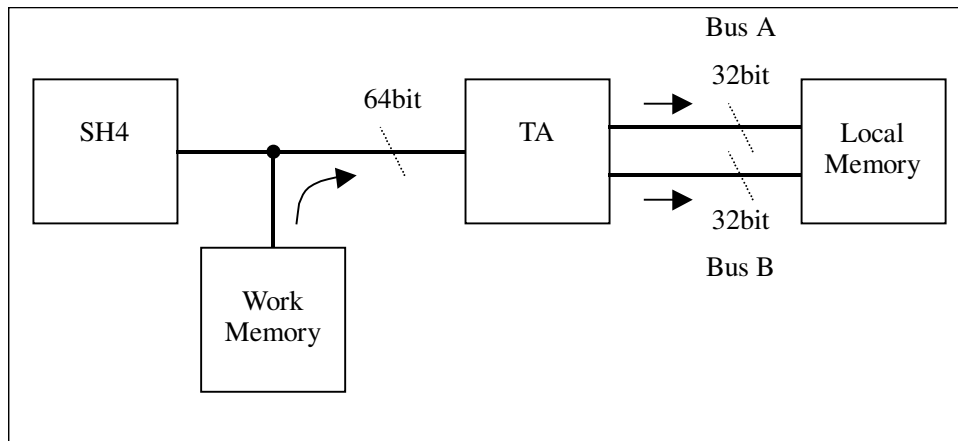


Fig. 2-4 LMMODE0/1 = 0 (Bus A & B)

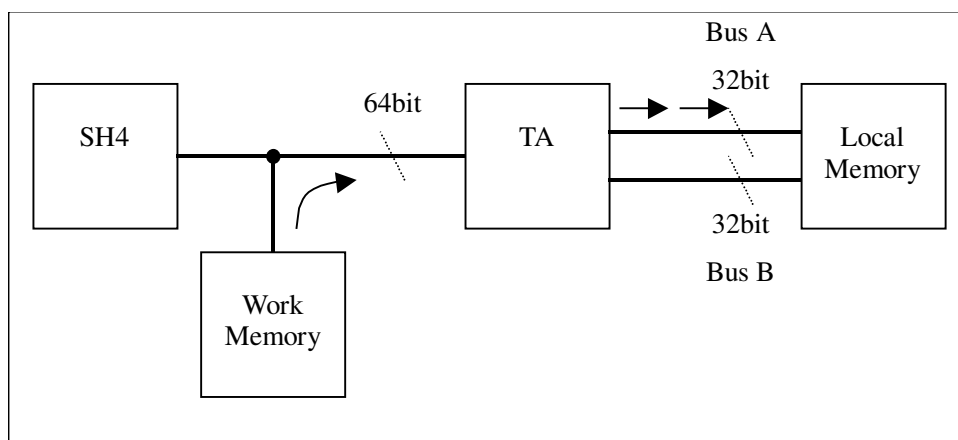


Fig.2-5 LMMODE0/1 = 1 (Bus A)

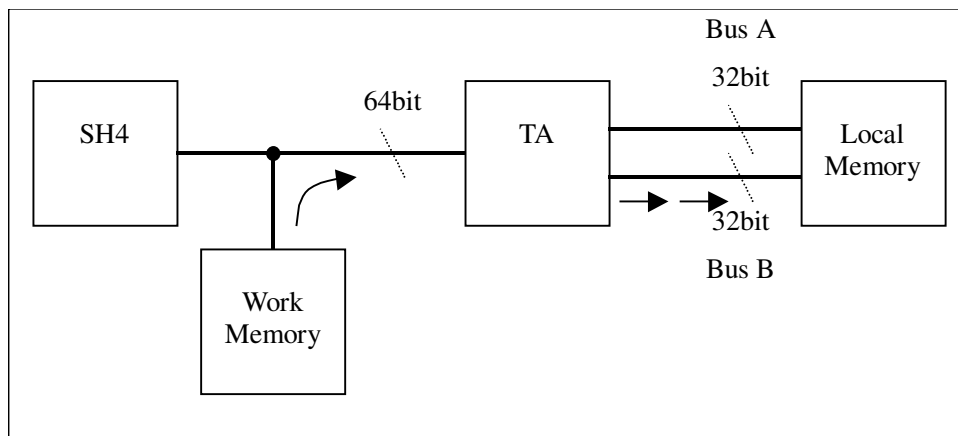


Fig. 2-6 LMMODE0/1 = 0 (Bus B)

When LMMODE0/1 = 0		When LMMODE0/1 = 1		
64bit access		32bit access	32bit access	
Bus A	Bus B	Bus A	Bus B	
0x00000000	0x00000004	}	0x00400000	
0x00000008	0x0000000c		0x00400004	
0x00000010	0x00000014		0x00400008	
0x00000018	0x0000001c		0x0040000c	
0x00000020	0x00000024		0x00400010	
0x00000028	0x0000002c		0x00400014	
.	.		.	
.	.		.	
.	.		.	
0x007ffff8	0x007ffffc		0x003ffffc	
0x00800000	0x00800004		}	0x00c00000
0x00800008	0x0080000c			0x00c00004
0x00800010	0x00800014			0x00c00008
0x00800018	0x0080001c			0x00c0000c
0x00800020	0x00800024	0x00c00010		
0x00800028	0x0080002c	0x00c00014		
.	.	.		
.	.	.		
.	.	.		
0x00ffffff8	0x00ffffffc	0x00bffffc		
		0x00ffffffc		

Fig. 2-7 Texture Memory Address

Similarly, the following section describes the SH4(-DMAC) registers that are used for ch2-DMA. (For details, refer to the item on DMAC in the SH4 manual.)

**SAR2** (ch2-DMA Source Address: P4 addr.-0xFFA00020、area7 addr.-0x1FA00020)

This specifies the ch2-DMA transfer destination address. The address that is set must lie at a 32-byte boundary.

Setting values: 0x0C000000 to 0x0FFFFFFE0 (system memory area)

**DMATCR2** (ch2-DMA Transfer Count: P4 addr.-0xFFA00028、area7 addr.-0x1FA00028)

This specifies the size of the ch2-DMA transfer, in units of 32 bytes.

The transfer size that is set must match the transfer size that is set in the SB\_C2DLEN register. Although the transfer size is specified in bytes in the SB\_C2DLEN register, here the transfer size is specified in units of 32 bytes (number of bytes/32). Values outside of the ranges shown below for the settings must not be set.

Setting values:	0x00000001:	32Bytes
	0x00000002:	64Bytes
	0x00000003:	96Bytes
	~	
	0x0007FFFF:	(16M-32)Bytes
	0x00080000:	16Mbytes

**CHCR2** (ch2-DMA channel Control: P4 addr.-0xFFA0002C、 area7 addr.-0x1FA0002C)  
This is the ch2-DMA control register. (For details, refer to the SH4 manual.)  
Never set a value other than those indicated in the table below.

Set Value	Source Addr. Mode	Transmit Mode	Interrupt Enable	DMA Enable
0x000012C1	Increment	Burst	Disable	Enable
0x000012C0	Increment	Burst	Disable	Disable
0x000012C5	Increment	Burst	Enable	Enable
0x000012C4	Increment	Burst	Enable	Disable
0x00001241	Increment	Cycle steal	Disable	Enable
0x00001240	Increment	Cycle steal	Disable	Disable
0x00001245	Increment	Cycle steal	Enable	Enable
0x00001244	Increment	Cycle steal	Enable	Disable
0x000022C1	Decrement	Burst	Disable	Enable
0x000022C0	Decrement	Burst	Disable	Disable
0x000022C5	Decrement	Burst	Enable	Enable
0x000022C4	Decrement	Burst	Enable	Disable
0x00002241	Decrement	Cycle steal	Disable	Enable
0x00002240	Decrement	Cycle steal	Disable	Disable
0x00002245	Decrement	Cycle steal	Enable	Enable
0x00002244	Decrement	Cycle steal	Enable	Disable
0x000002C1	Fix	Burst	Disable	Enable
0x000002C0	Fix	Burst	Disable	Disable
0x000002C5	Fix	Burst	Enable	Enable
0x000002C4	Fix	Burst	Enable	Disable
0x00000241	Fix	Cycle steal	Disable	Enable
0x00000240	Fix	Cycle steal	Disable	Disable
0x00000245	Fix	Cycle steal	Enable	Enable
0x00000244	Fix	Cycle steal	Enable	Disable
0x00000000	---	---	---	Disable

Table 2-11

\* The IE (Interrupt Enable) bit can also be generated from the HOLLY side; it does not matter which side generates the bit.

**DMAOR** (DMA operation: P4 addr.-0xFFA00040、 area7 addr.-0x1FA00040)  
This register sets the DMA transfer mode. (For details on the contents of this register, refer to the startup procedure or to the SH4 manual.)

The method for confirming the end of ch2-DMA is described below.

- 1) Although an interrupt is used in order to confirm the end of ch2-DMA, in order to generate the interrupt it is necessary to set bit 19 of either the SB\_IML2NRM, SB\_IML4NRM, or SB\_IML6NRM register (for details, refer to the interrupt manual) to "1" and release the ch2-DMA end interrupt mask. The mask needs to be released before initiating ch2-DMA.
- 2) Once ch2-DMA terminates, 0x00000000 is automatically set in the SB\_C2DST register, and at the same time bit 19 of the SB\_ISTNRM register (for details, refer to the interrupt manual) is set to "1." If the mask has been cancelled as described in item 1 above, an interrupt is now generated.
- 3) Once the SH4 receives the interrupt, it determines the source of the interrupt by reading the SB\_ISTNRM, SB\_ISTEXT, and SB\_ISTERR registers. The SH4 is able to confirm that ch2-DMA has ended by checking bit 19 of the SB\_ISTNRM register.
- 4) As soon as it has confirmed that ch2-DMA has ended, the SH4 cancels the interrupt by writing a "1" to bit 19 of the SB\_ISTNRM register.

Note) A separate ch2-DMA end interrupt exists for the SH4-DMAC. when using the interrupt described above, it is necessary to mask the interrupt for the SH4-DMAC by setting the IE bit in the SH4-DMAC-CHCR2 register to "0." Conversely, when using the interrupt for the SH4-DMAC, it is necessary to mask the interrupt described above by setting bit 19 in the SB\_IML2NRM register, the SB\_IML4NRM register, and the SB\_IML6NRM register all to "0."

The following procedure explains how to stop ch2-DMA:

- 1) Request a stop of ch2-DMA by writing 0x00000000 to the SB\_C2DST register.
- 2) Note that after performing step 1, the value in the SB\_C2DST register does not immediately become 0x00000000; instead, the value 0x00000001 is maintained in the register until ch2-DMA stops completely. Therefore, it is necessary to poll the register repeatedly until its value becomes 0x00000000.
- 3) Once the register value becomes 0x00000000, ch2-DMA has stopped. At this point, the contents of the SB\_C2DSTAT, SB\_C2DLEN, SH4-DMAC-SAR2, and SH4-DMAC-DMATCR registers indicate the address from which the next data item was to be transferred and the amount of data remaining to be transferred. If the value of the SB\_C2DLEN and SH4-DMAC-DMATCR registers is 0x00000000, that indicates that the transfer has been completed; in this case, it is not permissible to resume the ch2-DMA transfer.

Supplement 1) After stopping a ch2-DMA transfer, it is possible to change the register values and then begin a new ch2-DMA transfer.

Supplement 2) When a ch2-DMA transfer is stopped, no ch2-DMA end interrupt is generated.

If a ch2-DMA transfer was stopped by the method described above, it can be resumed by writing 0x00000001 to the SB\_C2DST register; this causes the transfer to resume from the position where it was stopped. However, the values in the registers when the transfer is resumed must be identical to the values that were in the registers when the transfer was stopped.

#### § 2.6.4.2 YUV Texture Transfer

ch2-DMA is used to conduct DMA transfers of YUV textures.

In order to conduct a DMA transfer of a YUV texture, two separate procedures are required. First, set the TA registers, and then make the ch2-DMA settings. Each of these procedures is described below.

- Setting the TA registers

- (1) Set the starting address (the relative address from the start of texture memory) where the YUV texture is to be stored in the TA\_YUV\_TEX\_BASE register.
- (2) Make the YUV texture settings in the TA\_YUV\_TEX\_CTRL register.
- (3) Read the TA\_YUV\_TEX\_CTRL register. (Because, from the viewpoint of the CPU, a write to a TA register consists of writing the data to the buffer ahead of the register and then forgetting about it, perform only one read and then wait until the write to the register is completed.)

- Setting up the ch2-DMA transfer

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set the transfer source address in the SH4-DMAC-SAR2 register.
- (4) Set the size of the transfer (the number of bytes to be transferred/32) in the SH4-DMAC-DMATCR2 register.
- (5) Make the operation settings in the SH4-DMAC-CHCR2 register. When doing so, set the DE bit to "1."
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1." DMA cannot be initiated if the AE, NMIF, and DME bits do not all meet this condition. Furthermore, if the DDT bit is "0," the DMA operation will be performed incorrectly.
- (7) Set the address for the TA's YUV texture converter in the SB\_C2DSTAT register.
- (8) Set the transfer size (the number of bytes) in the SB\_C2DLEN register.
- (9) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

The following prohibitions apply during a YUV texture DMA transfer:

- Because a YUV texture DMA transfer is performed using ch2-DMA, other DMA operations that use ch2-DMA cannot be performed at the same time.
- Never write to any of the registers that are used in a YUV texture DMA transfer. The only exception is writing 0x00000000 to the SB\_C2DST register in order to interrupt the DMA transfer.
- The CPU must not perform a burst write to addresses 0x10000000 to 0x13FFFFE0. Doing so could result in the loss of some data in the DMA transfer.

The status of each register when a YUV texture DMA transfer ends normally is described below:

- The SH4\_DMAC\_SAR2 register points to the address that follows the location at which the transfer ended.
- The value in the SH4\_DMAC\_DMATCR2 register is 0x00000000.
- The TE bit of the SH4\_DMAC\_CHCR2 register is "1."
- The SB\_C2DSTAT register retains the value that was set.
- The value in the SB\_C2DLEN register is 0x00000000.
- The value in the SB\_C2DST register is 0x00000000.
- The DMA end interrupt flag (SB\_ISTNRM - bit 19: DTDE2INT) is set to "1."

An example of how to use YUV texture DMA transfer is provided below.

**Transferring YUV420 texture data (8 \* 8 macro blocks: 0x00006000 bytes) from system memory to texture memory, converting the data to YUV422**

System memory addresses (YUV420 texture):	0x0C200000 to 0x0C205FFF
TA address:	0x10800000
Texture memory addresses (YUV422 texture):	0x00600000 -

● Example for setting the TA registers

- (1) Set 0x00600000 in the TA\_YUV\_TEX\_BASE register.
- (2) Set 0x00000707 in the TA\_YUV\_TEX\_CTRL register.
- (3) Read the TA\_YUV\_TEX\_CTRL register.

● Example for setting up the ch2-DMA transfer

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set 0x0C200000 in the SH4-DMAC-SAR2 register.
- (4) Set 0x00000300 in the SH4-DMAC-DMATCR2 register.
- (5) Set 0x000012C1 in the SH4-DMAC-CHCR2 register.
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1."
- (7) Set 0x10800000 in the SB\_C2DSTAT register.
- (8) Set 0x00006000 in the SB\_C2DLEN register.
- (9) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.



## § 2.6.5 Display List Transfers

There are two methods for transferring display lists (polygon parameters): a method that uses ch2-DMA and a method that uses Sort-DMA (cho:DDT).

### § 2.6.5.1 Direct Display list DMA

ch2-DMA is used to conduct DMA transfers of direct display lists. The setup procedure is described below.

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set the transfer source address in the SH4-DMAC-SAR2 register.
- (4) Set the size of the transfer (the number of bytes to be transferred/32) in the SH4-DMAC-DMATCR2 register.
- (5) Make the operation settings in the SH4-DMAC-CHCR2 register. When doing so, set the DE bit to "1."
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1." DMA cannot be initiated if the AE, NMIF, and DME bits do not all meet this condition. Furthermore, if the DDT bit is "0," the DMA operation will be performed incorrectly.
- (7) If the address that is set in the SB\_C2DSTAT register is within the range from 0x11000000 to 0x11FFFFE0, set 0x00000001 in the SB\_LMMODE0 register.
- (8) If the address that is set in the SB\_C2DSTAT register is within the range from 0x13000000 to 0x13FFFFE0, set 0x00000001 in the SB\_LMMODE1 register.
- (9) Set the transfer destination address in the SB\_C2DSTAT register.
- (10) Set the transfer size (the number of bytes) in the SB\_C2DLEN register.
- (11) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

The following prohibitions apply during a direct display list DMA transfer:

- Because a direct display list DMA transfer is performed using ch2-DMA, other DMA operations that use ch2-DMA cannot be performed at the same time.
- Never write to any of the registers that are used in a direct display list DMA transfer. The only exception is writing 0x00000000 to the SB\_C2DST register in order to interrupt the DMA transfer.
- The CPU must not perform a burst write to addresses 0x10000000 to 0x13FFFFE0. Doing so could result in the loss of some data in the DMA transfer.

The status of each register when a direct display list DMA transfer ends normally is described below:

- The SH4\_DMAC\_SAR2 register points to the address that follows the location at which the transfer ended.
- The value in the SH4\_DMAC\_DMATCR2 register is 0x00000000.
- The TE bit of the SH4\_DMAC\_CHCR2 register is "1."
- The SB\_C2DSTAT register points to the address that follows the location at which the transfer ended.
- The value in the SB\_C2DLEN register is 0x00000000.
- The value in the SB\_C2DST register is 0x00000000.
- The DMA end interrupt flag (SB\_ISTNRM - bit 19: DTDE2INT) is set to "1."

An example of how to use direct display list DMA transfer is provided below.

**Transferring a direct display list (0x00002000 bytes) from system memory to texture memory**

System memory addresses: 0x0C400000 to 0x0C401FFF

TA addresses: 0x11600000 to 0x11601FFF

Texture memory addresses: 0x00600000 to 0x00601FFF

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC\_CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set 0x0C400000 in the SH4-DMAC\_SAR2 register.
- (4) Set 0x00000100 in the SH4-DMAC\_DMATCR2 register.
- (5) Set 0x000012C1 in the SH4-DMAC\_CHCR2 register.
- (6) Read the SH4-DMAC\_DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1."
- (7) Because the address that is set in the SB\_C2DSTAT register is within the range from 0x11000000 to 0x11FFFFE0, set 0x00000001 in the SB\_LMMODE0 register.
- (8) Because the address that is set in the SB\_C2DSTAT register is not within the range from 0x13000000 to 0x13FFFFE0, do not set the SB\_LMMODE1 register.
- (9) Set 0x11600000 in the SB\_C2DSTAT register.
- (10) Set 0x00002000 in the SB\_C2DLEN register.
- (11) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

### § 2.6.5.2 TA Input Display List Transfers

In order to perform a DMA transfer for a display list for input to the TA, it is necessary to first set the TA registers and then to set up the ch2-DMA transfer. Each of these setup procedures is described below.

#### ● Setting the TA registers

- (1) Set the starting address (the relative address from the start of texture memory) for where the Object List is to be stored in the TA\_OL\_BASE register.
- (2) Set the starting address (the relative address from the start of texture memory) for where the ISP/TSP Parameters are to be stored in the TA\_ISP\_BASE register.
- (3) Set the limit address (the relative address from the start of texture memory) for where the Object List is to be stored in the TA\_OL\_LIMIT register.
- (4) Set the limit address (the relative address from the start of texture memory) for where the ISP/TSP Parameters are to be stored in the TA\_ISP\_LIMIT register.
- (5) Set the Global Tile Clip value in the TA\_GLOB\_TILE\_CLIP register.
- (6) Set the Object Pointer Block unit size in the TA\_ALLOC\_CTRL register.
- (7) Write 0x80000000 in the TA\_LIST\_INIT register to initialize the TA's internal registers.
- (8) Read the TA\_LIST\_INIT register. (Because, from the viewpoint of the CPU, a write to a TA register consists of writing the data to the buffer ahead of the register and then forgetting about it, perform only one read and then wait until the write to the register is completed.)

#### ● Setting up the ch2-DMA transfer

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set the transfer source address in the SH4-DMAC-SAR2 register.
- (4) Set the size of the transfer (the number of bytes to be transferred/32) in the SH4-DMAC-DMATCR2 register.
- (5) Make the operation settings in the SH4-DMAC-CHCR2 register. When doing so, set the DE bit to "1."
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1." DMA cannot be initiated if the AE, NMIF, and DME bits do not all meet this condition. Furthermore, if the DDT bit is "0," the DMA operation will be performed incorrectly.
- (7) Set the address for the TA's display list input in the SB\_C2DSTAT register.
- (8) Set the transfer size (the number of bytes) in the SB\_C2DLEN register.
- (9) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

The following prohibitions apply during a TA input display list DMA transfer:

- Because a TA input display list DMA transfer is performed using ch2-DMA, other DMA operations that use ch2-DMA cannot be performed at the same time.
- Never write to any of the registers that are used in a TA input display list DMA transfer. The only exception is writing 0x00000000 to the SB\_C2DST register in order to interrupt the DMA transfer.
- The CPU must not perform a burst write to addresses 0x10000000 to 0x13FFFFE0. Doing so could result in the loss of some data in the DMA transfer.

The status of each register when a TA input display list DMA transfer ends normally is described below:

- The SH4\_DMAC\_SAR2 register points to the address that follows the location at which the transfer ended.

- The value in the SH4\_DMAC\_DMATCR2 register is 0x00000000.
- The TE bit of the SH4\_DMAC\_CHCR2 register is "1."
- The SB\_C2DSTAT register retains the value that was set.
- The value in the SB\_C2DLEN register is 0x00000000.
- The value in the SB\_C2DST register is 0x00000000.
- The DMA end interrupt flag (ISTNRM - bit 19: DTDE2INT) is set to "1."

An example of how to use TA input display list DMA transfer is provided below.

**Transferring a display list (0x00008000 bytes) from system memory to texture memory**

System memory addresses:	0x0C400000 to 0x0C407FFF
TA address:	0x10000000
Texture memory addresses (Object List):	0x00100000~
Texture memory addresses (ISP/TSP Parameters):	0x00000000~

● Example for setting the TA registers

- (1) Set 0x00100000 in the TA\_OL\_BASE register.
- (2) Set 0x00000000 in the TA\_ISP\_BASE register.
- (3) Set 0x00200000 in the TA\_OL\_LIMIT register.
- (4) Set 0x00100000 in the TA\_ISP\_LIMIT register.
- (5) Set 0x000E0013 in the TA\_GLOB\_TILE\_CLIP register.
- (6) Set 0x00000202 in the TA\_ALLOC\_CTRL register.
- (7) Write 0x80000000 in the TA\_LIST\_INIT register to initialize the TA's internal registers.
- (8) Read the TA\_LIST\_INIT register.

● Example for setting up the ch2-DMA transfer

- (1) Read the SB\_C2DST register and confirm that the value is 0x00000000.
- (2) Read the SH4-DMAC-CHCR2 register, and confirm that both the TE bit and the DE bit are both set to "0." If either bit is set to "1," set that bit to "0."
- (3) Set 0x0C400000 in the SH4-DMAC-SAR2 register.
- (4) Set 0x00000400 in the SH4-DMAC-DMATCR2 register.
- (5) Set 0x000012C1 in the SH4-DMAC-CHCR2 register.
- (6) Read the SH4-DMAC-DMAOR register and confirm that the DDT bit is set to "1," the AE bit is set to "0," the NMIF bit is set to "0," and the DME bit is set to "1."
- (7) Set 0x10000000 in the SB\_C2DSTAT register.
- (8) Set 0x00008000 in the SB\_C2DLEN register.
- (9) Write 0x00000001 in the SB\_C2DST register to initiate the DMA operation.

### § 2.6.5.3 Sort-DMA Transfer of $\alpha$ Polygon Parameters

DMA cho (DDT) is used to transfer  $\alpha$  polygon parameters by means of a Sort-DMA transfer. In order to perform an  $\alpha$  polygon Sort-DMA transfer, it is necessary to first set the TA registers and then to set up the Sort-DMA transfer. Each of these setup procedures is described below.

#### ● Setting the TA registers

- (1) Set the starting address (the relative address from the start of texture memory) for where the Object List is to be stored in the TA\_OL\_BASE register.
- (2) Set the starting address (the relative address from the start of texture memory) for where the ISP/TSP Parameters are to be stored in the TA\_ISP\_BASE register.
- (3) Set the limit address (the relative address from the start of texture memory) for where the Object List is to be stored in the TA\_OL\_LIMIT register.
- (4) Set the limit address (the relative address from the start of texture memory) for where the ISP/TSP Parameters are to be stored in the TA\_ISP\_LIMIT register.
- (5) Set the Global Tile Clip value in the TA\_GLOB\_TILE\_CLIP register.
- (6) Set the Object Pointer Block unit size in the TA\_ALLOC\_CTRL register.
- (7) Write 0x80000000 in the TA\_LIST\_INIT register to initialize the TA's internal registers.
- (8) Read the TA\_LIST\_INIT register. (Because, from the viewpoint of the CPU, a write to a TA register consists of writing the data to the buffer ahead of the register and then forgetting about it, perform only one read and then wait until the write to the register is completed.)

#### ● Setting up the Sort-DMA transfer

- (1) Read the SB\_SDST register and confirm that the value is 0x00000000.
- (2) Set the start address of the Start Link Address Table in the SB\_SDSTAW register.
- (3) Set the Link Base Address in the SB\_SDBAAW register.
- (4) Set the bit width of the Start Link Address in the SB\_SDWLT register.
- (5) Set the Link Address shift control in the SB\_SDLAS register.
- (6) Write 0x00000001 in the SB\_SDST register to initiate the DMA operation.

The following prohibitions apply during an  $\alpha$  polygon Sort-DMA transfer:

- Never write to any of the registers that are used in an  $\alpha$  polygon Sort-DMA transfer. The only exception is writing 0x00000000 to the SB\_SDST register in order to interrupt the DMA transfer.
- The CPU must not perform a burst write to addresses 0x10000000 to 0x13FFFFE0. Doing so could result in the loss of some data in the DMA transfer.

The status of each register when an  $\alpha$  polygon Sort-DMA transfer ends normally is described below:

- The SB\_SDSTAW register value is incremented.
- The value in the SB\_SDST register is 0x00000000.
- The SB\_SDDIV register the number of times that the Sort-DMA operation read the Start Link Address.
- The DMA end interrupt flag (SB\_ISTNRM - bit 20: DTDESINT) is set to "1."

An example of how to use  $\alpha$  polygon Sort-DMA transfer is provided below.

**Transferring a display list ( $\alpha$  polygon) from system memory to texture memory by means of Sort-DMA**

Start Link Address Table address in system memory:	0x0C600000
Link Base Address in system memory:	0x0C604000
TA address (fixed):	0x10000000
Texture memory addresses (Object List):	0x00100000~
Texture memory addresses (ISP/TSP Parameters):	0x00000000~

● Example for setting the TA registers

- (1) Set 0x00100000 in the TA\_OL\_BASE register.
- (2) Set 0x00000000 in the TA\_ISP\_BASE register.
- (3) Set 0x00200000 in the TA\_OL\_LIMIT register.
- (4) Set 0x00100000 in the TA\_ISP\_LIMIT register.
- (5) Set 0x000E0013 in the TA\_GLOB\_TILE\_CLIP register.
- (6) Set 0x00000202 in the TA\_ALLOC\_CTRL register.
- (7) Write 0x80000000 in the TA\_LIST\_INIT register to initialize the TA's internal registers.
- (8) Read the TA\_LIST\_INIT register.

● Example for setting up the Sort-DMA transfer

- (1) Read the SB\_SDST register and confirm that the value is 0x00000000.
- (2) Set 0x0C600000 in the SB\_SDSTAW register.
- (3) Set 0x0C604000 in the SB\_SDBAAW register.
- (4) Set 0x00000001 in the SB\_SDWLT register.
- (5) Set 0x00000000 in the SB\_SDLAS register.
- (6) Write 0x00000001 in the SB\_SDST register to initiate the DMA operation.

-- (Supplement) About Sort-DMA --

Sort-DMA permits the transfer of random data from system memory to texture memory by adding link information to the polygon parameters.

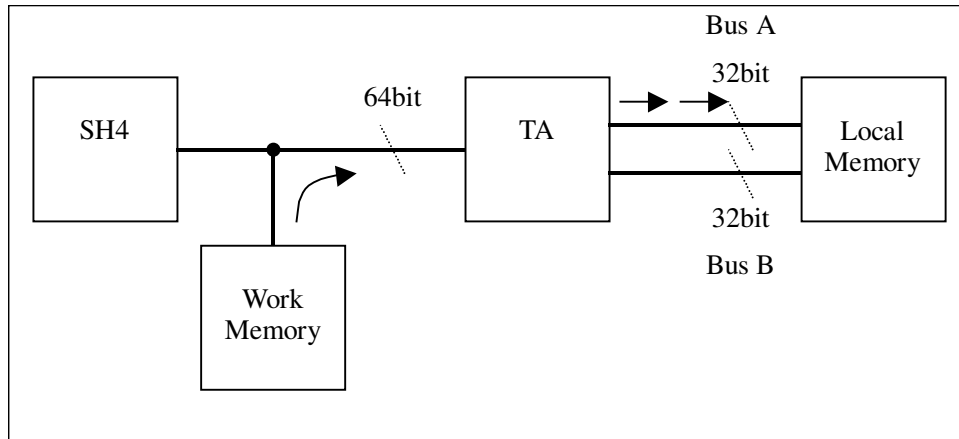


Fig. 2-8 Sort-DMA Transfer Path (Bus A)

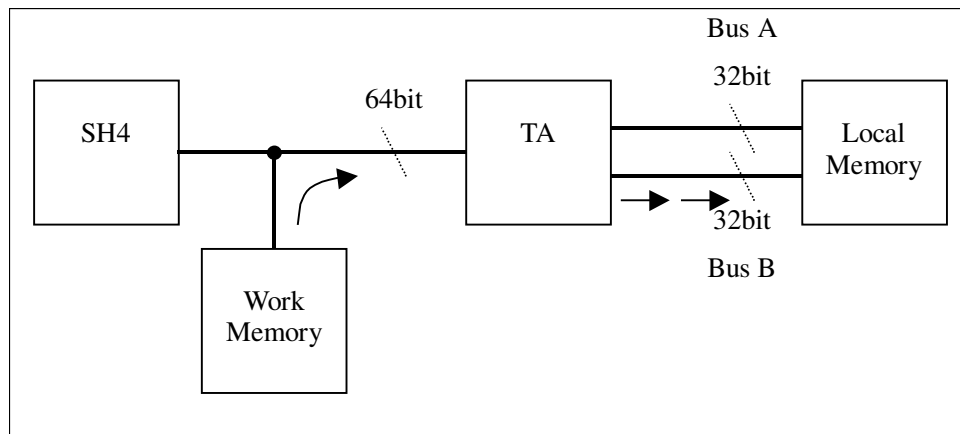


Fig. 2-9 Sort-DMA Transfer Path (Bus B)

Polygon parameters must be sorted in order to draw an  $\alpha$  polygon. Normally, this is accomplished either by using the Renderer's sorting function, or by having the CPU sort the parameters beforehand. When drawing a large number of  $\alpha$  polygons, it is probably more effective to have the CPU perform the sorting rather than using an auto-sorting function. However, sorting a large amount of polygon data can consume a large amount of CPU time. If the CPU generates link information in the  $\alpha$  polygon parameters, and then performs the transfer using the Sort-DMA function, it is possible to reduce the load on both the CPU and on the Renderer.

The HOLLY registers that are used in Sort-DMA operations are listed below. (For details, refer to section 8.4.1, "System Bus Register.")

Note that it is not possible to specify the texture memory address that is the transfer destination for the data in the Sort-DMA registers. For details on specifying the texture memory address, refer to the TA manual.

<b><i>SB_SDSTAW</i></b>	(0x005F6810)
<b><i>SB_SDBAAW</i></b>	(0x005F6814)
<b><i>SB_SDWLT</i></b>	(0x005F6818)
<b><i>SB_SDLAS</i></b>	(0x005F681C)
<b><i>SB_SDST</i></b>	(0x005F6820)
<b><i>SB_SDDIV</i></b>	(0x005F6860)

### — Start Link Address Table

The Start Link Address Table consists of several Start Link Addresses. A Start Link Address is needed required for Sort-DMA for each Sort-DMA polygon parameter list in order to know the starting position of the link.

- The starting position of the Start Link Address Table is specified by the SB\_SDSTAW register.
- Because the address that is produced by adding the value that is in the SB\_SDBAAW register to the Start Link Address is used in Sort-DMA operations as the link destination address, it is the same as writing the offset from SB\_SDBAAW for the Next Link Address.
- The bit width of the Start Link Address is selected through the SB\_SDWLT register as either 16 bits or 32 bits.
- The value that is written for the Start Link Address is selected through the SB\_SDLAS register as either the original address or the address divided by 32.
- If either 0x0001 (when SB\_SDWLT = 0) or 0x00000001 (when SB\_SDWLT = 1) is written in the Start Link Address, the Sort-DMA operation recognizes this as the End Of List code, and begins to read the next Start Link Address.
- If either 0x0002 (when SB\_SDWLT = 0) or 0x00000002 (when SB\_SDWLT = 1) is written in the Start Link Address, the Sort-DMA operation recognizes this as the End Of DMA code, and ends the transfer.

When the Start Link Address consist of 16bit		When the Start Link Address consist of 32bit	
address	data 16bit	address	data 32bit
SDSTAW +0x0000	Start Link Address #0	SDSTAW +0x0000	Start Link Address #0
0x0002	Start Link Address #1	0x0004	Start Link Address #1
0x0004	Start Link Address #2	0x0008	Start Link Address #2
0x0006	Start Link Address #3	0x000c	Start Link Address #3
0x0008	Start Link Address #4	0x0010	Start Link Address #4
0x000a	Start Link Address #5	0x0014	Start Link Address #5
:	:	:	:
0x0002 * n	Start Link Address #n	0x0004 * n	Start Link Address #n

Fig. 2-10 Start Link Address Table Format



**Start Link Address** : Specify the offset for the starting addresses where the polygon parameters that are to be transferred at the beginning of each parameter list are stored.

Set Value (SB\_SDWLT=0,SB\_SDLAS=0)

0x0000 : Offset Address = 0x00000000  
0x0080 : Offset Address = 0x00000080  
0x00A0 : Offset Address = 0x000000A0  
0x00C0 : Offset Address = 0x000000C0  
~  
0xFFC0 : Offset Address = 0x0000FFC0  
0xFFE0 : Offset Address = 0x0000FFE0  
0x0001 : End OF List  
0x0002 : End OF DMA

Set Value (SB\_SDWLT=0,SB\_SDLAS=1)

0x0000 : Offset Address = 0x00000000  
0x0004 : Offset Address = 0x00000080  
0x0005 : Offset Address = 0x000000A0  
0x0006 : Offset Address = 0x000000C0  
~  
0xFFFE : Offset Address = 0x001FFFC0  
0xFFFF : Offset Address = 0x001FFFE0  
0x0001 : End OF List  
0x0002 : End OF DMA

Set Value (SB\_SDWLT=1,SB\_SDLAS=0)

0x00000000 : Offset Address = 0x00000000  
0x00000080 : Offset Address = 0x00000080  
0x000000A0 : Offset Address = 0x000000A0  
0x000000C0 : Offset Address = 0x000000C0  
~  
0x07FFFFC0 : Offset Address = 0x07FFFFC0  
0x07FFFFE0 : Offset Address = 0x07FFFFE0  
0x00000001 : End OF List  
0x00000002 : End OF DMA

Set Value (SB\_SDWLT=1,SB\_SDLAS=1)

0x00000000 : Offset Address = 0x00000000  
0x00000004 : Offset Address = 0x00000080  
0x00000005 : Offset Address = 0x000000A0  
0x00000006 : Offset Address = 0x000000C0  
~  
0x003FFFFE : Offset Address = 0x07FFFFC0  
0x003FFFFF : Offset Address = 0x07FFFFE0  
0x00000001 : End OF List  
0x00000002 : End OF DMA

- Never specify any values other than those shown above.

#### – Polygon Parameter

There are three types of polygon parameters: Control Parameters, Global Parameters, and Vertex Parameters. In Sort-DMA, the link destination address is calculated on the basis of the link information that is contained in the Global Parameters. Therefore, when sorting several polygons, the data must be created by adding Global Parameters to each Vertex Parameter, divided up by polygons. The parameter format for each polygon is illustrated below.

The seventh 32-bit word from the start of the Global Parameters is allocated for the current data size, and the eighth 32-bit word from the start of the Global Parameters is allocated for the

Next Link Address. Write the size (in units of 32 bytes) of the polygon parameters that are currently being transferred for the current data size, and indicate the address where the next polygon parameters that are to be transferred are stored for the Next Link Address.

- Because the address that is produced by adding the value that is in the SB\_SDBAAW register to the Next Link Address is used in Sort-DMA operations as the link destination address, it is the same as writing the offset from SB\_SDBAAW for the Next Link Address.
- The value that is written for the Next Link Address is selected through the SB\_SDLAS register as either the original address or the address divided by 32.
- If 0x00000001 is written in the Next Link Address, the Sort-DMA operation recognizes this as the End Of List code, and begins to read the next Start Link Address from the Start Link Address Table as the new link destination address.
- If 0x00000002 is written in the Next Link Address, the Sort-DMA operation recognizes this as the End Of DMA code, and ends the transfer.

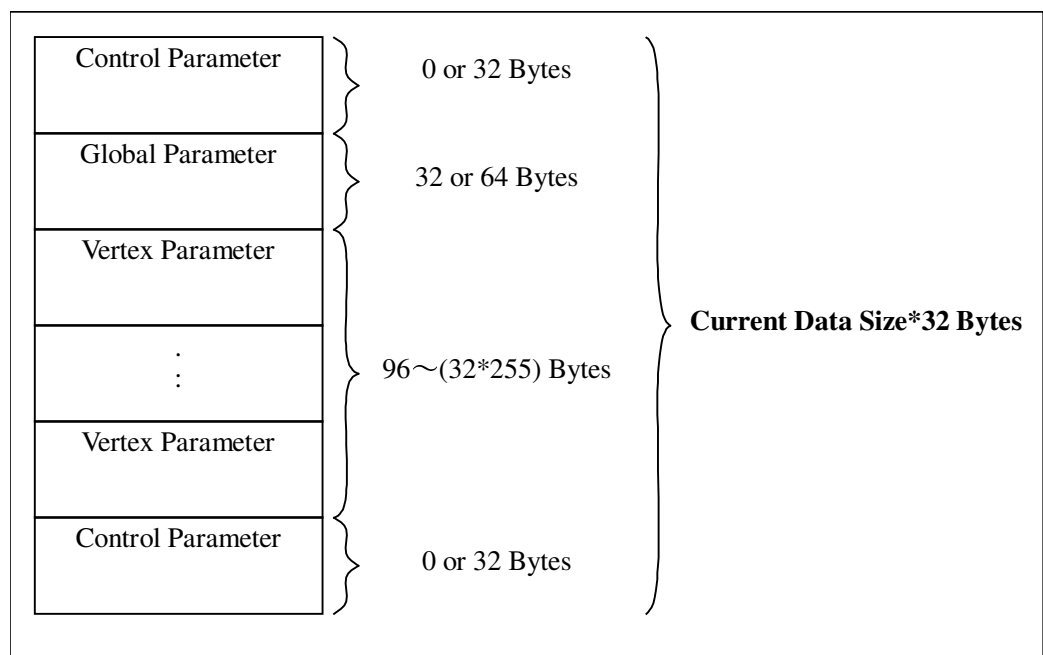


Fig. 2-11 Polygon Parameter Format

When the Global Parameters consist of 32 bytes		When the Global Parameters consist of 64 bytes	
address	data	address	data
0x00	<i>Parameter Control Word</i>	0x00	<i>Parameter Control Word</i>
0x04	<i>ISP/TSP Instruction Word</i>	0x04	<i>ISP/TSP Instruction Word</i>
0x08	<i>TSP Instruction Word</i>	0x08	<i>TSP Instruction Word</i>
0x0c	<i>Texture Control Word</i>	0x0c	<i>Texture Control Word</i>
0x10	-	0x10	-
0x14	-	0x14	-
0x18	<b>Current Data Size</b>	0x18	<b>Current Data Size</b>
0x1c	<b>Next Link Address</b>	0x1c	<b>Next Link Address</b>
		0x20	<i>Face Color Alpha</i>
		0x24	<i>Face Color R</i>
		0x28	<i>Face Color G</i>
		0x2c	<i>Face Color B</i>
		0x30	<i>Face Offset Color Alpha</i>
		0x34	<i>Face Offset Color R</i>
		0x38	<i>Face Offset Color G</i>
		0x3c	<i>Face Offset Color B</i>

Fig. 2-12 Global Parameter Format

**Current Data Size** : Specify the value that is the size of the polygon parameters (control + global + vertex) that are currently being transferred, divided by 32. No values other than those listed below may be specified.

Set Value	0x00000004	: 128Bytes
	0x00000005	: 160Bytes
	0x00000006	: 192Bytes
	~	
	0x000000FF	: 8160Bytes
	0x00000000	: 8192Bytes

**Next Link Address** : Specify the offset value for the starting address where the polygon parameters that are to be transferred next are stored.

Set Value (SB_SDLAS=0)		
	0x00000000	: Offset Address = 0x00000000
	0x00000080	: Offset Address = 0x00000080
	0x000000A0	: Offset Address = 0x000000A0
	0x000000C0	: Offset Address = 0x000000C0
	~	
	0x07FFFFC0	: Offset Address = 0x07FFFFC0
	0x07FFFFE0	: Offset Address = 0x07FFFFE0
	0x00000001	: End Of List
	0x00000002	: End Of DMA

Set Value (SB_SDLAS=1)		
	0x00000000	: Offset Address = 0x 00000000
	0x00000004	: Offset Address = 0x 00000080
	0x00000005	: Offset Address = 0x 000000A0
	0x00000006	: Offset Address = 0x 000000C0
	~	
	0x003FFFFE	: Offset Address = 0x 07FFFFC0
	0x003FFFFF	: Offset Address = 0x 07FFFFE0
	0x00000001	: End Of List
	0x00000002	: End Of DMA

- Never specify any values other than those shown above.

### – Supplement concerning Sort-DMA

The method for confirming the end of Sort-DMA is described below.

- (1) Although an interrupt is used in order to confirm the end of Sort-DMA, in order to generate the interrupt it is necessary to set bit 20 of either the SB\_IML2NRM, SB\_IML4NRM, or SB\_IML6NRM register (for details, refer to the interrupt manual) to "1" and release the Sort-DMA end interrupt mask. The mask needs to be released before initiating Sort-DMA.
- (2) Once Sort-DMA terminates, 0x00000000 is automatically set in the SB\_SDST register, and at the same time bit 20 of the SB\_ISTNRM register (for details, refer to the interrupt manual) is set to "1." If the mask has been cancelled as described in item 1 above, an interrupt is now generated.
- (3) Once the SH4 receives the interrupt, it determines the source of the interrupt by reading the SB\_ISTNRM, SB\_ISTEXT, and SB\_ISTERR registers. The SH4 is able to confirm that Sort-DMA has ended by checking bit 20 of the SB\_ISTNRM register.
- (4) As soon as it has confirmed that Sort-DMA has ended, the SH4 cancels the interrupt by writing a "1" to bit 20 of the SB\_ISTNRM register.

The following procedure explains how to interrupt Sort-DMA:

- (1) Request a stop of Sort-DMA by writing 0x00000000 to the SB\_SDST register.
- (2) Note that after performing step 1, the value in the SB\_SDST register does not immediately become 0x00000000; instead, the value 0x00000001 is maintained in the register until Sort-DMA stops completely. Therefore, it is necessary to poll the register repeatedly until its value becomes 0x00000000.
- (3) Once the register value becomes 0x00000000, Sort-DMA has stopped. At this point, SB\_SDDIV indicates the number of times a Start Link Address was retrieved; this information can be used in order to make a rough estimate of how far the transfer proceeded.

Supplement) When a Sort-DMA transfer is interrupted, no Sort-DMA end interrupt is generated.

The method for generating a Sort-DMA parameter error interrupt is described below.

- (1) In order to generate a Sort-DMA parameter error interrupt it is necessary to set bit 28 of either the SB\_IML2ERR, SB\_IML4ERR, or SB\_IML6ERR register (for details, refer to the interrupt manual) to "1" and release the Sort-DMA parameter error interrupt mask. The mask needs to be released before initiating Sort-DMA.
- (2) If a parameter error is generated, 0x00000000 is automatically set in the SB\_SDST register, and at the same time bit 28 of the SB\_ISTNRM register (for details, refer to the interrupt manual) is set to "1." Sort-DMA is forcibly terminated. If the mask has been cancelled as described in item 1 above, an interrupt is now generated.

Supplement 1) A parameter error occurs when the Global Parameters could not be found in the Sort-DMA transfer data. It is possible either that the format of the source data differs from that shown in the Fig. 2-10 Polygon Parameter format, or an incorrect value was written for a link address.

Supplement 2) When a Sort-DMA transfer is forcibly terminated, no Sort-DMA end interrupt is generated.

– Sort-DMA Operation Flowchart

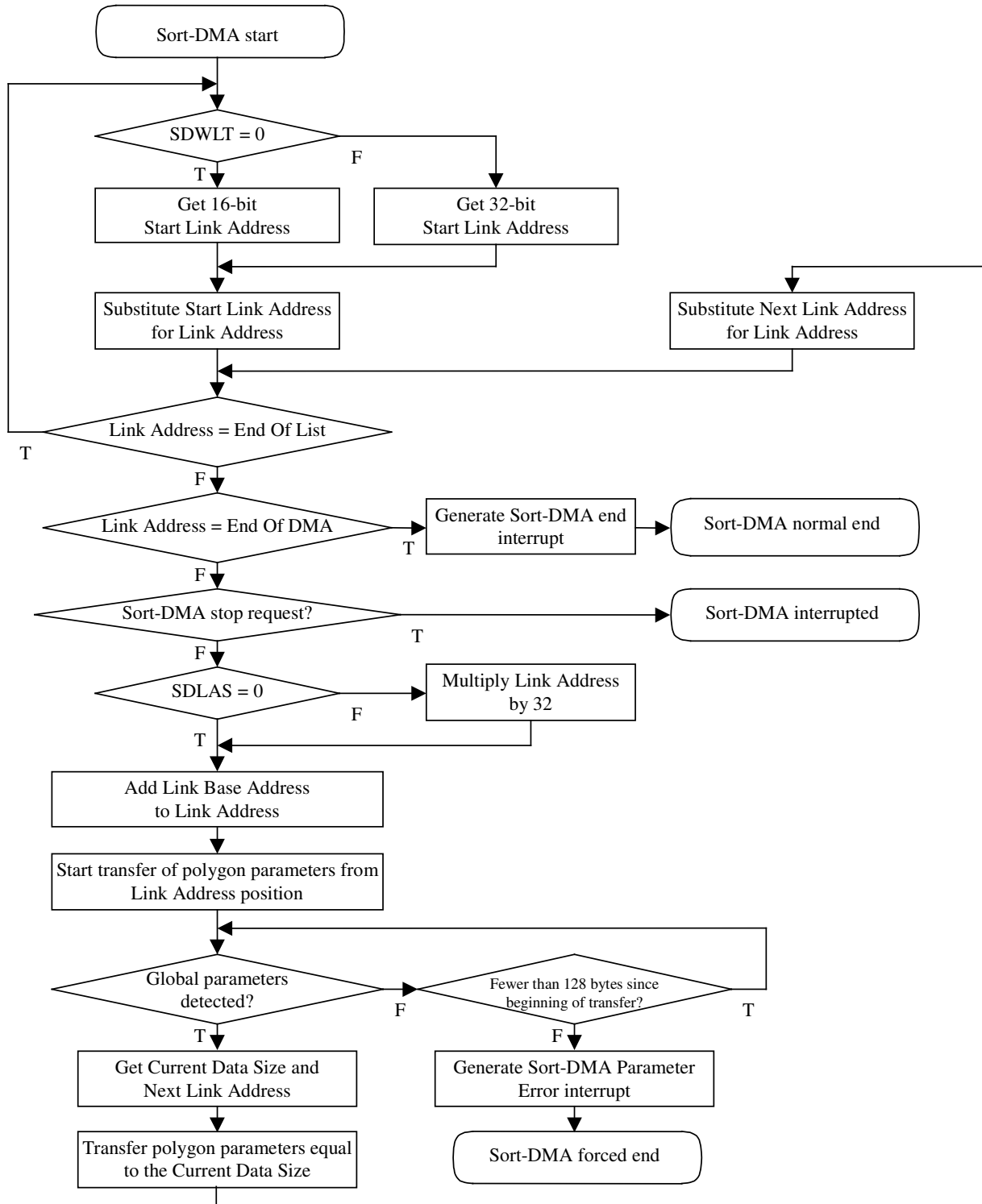


Fig. 2-13 Sort-DMA Operation Flowchart

**– Sort-DMA Transfer Example**

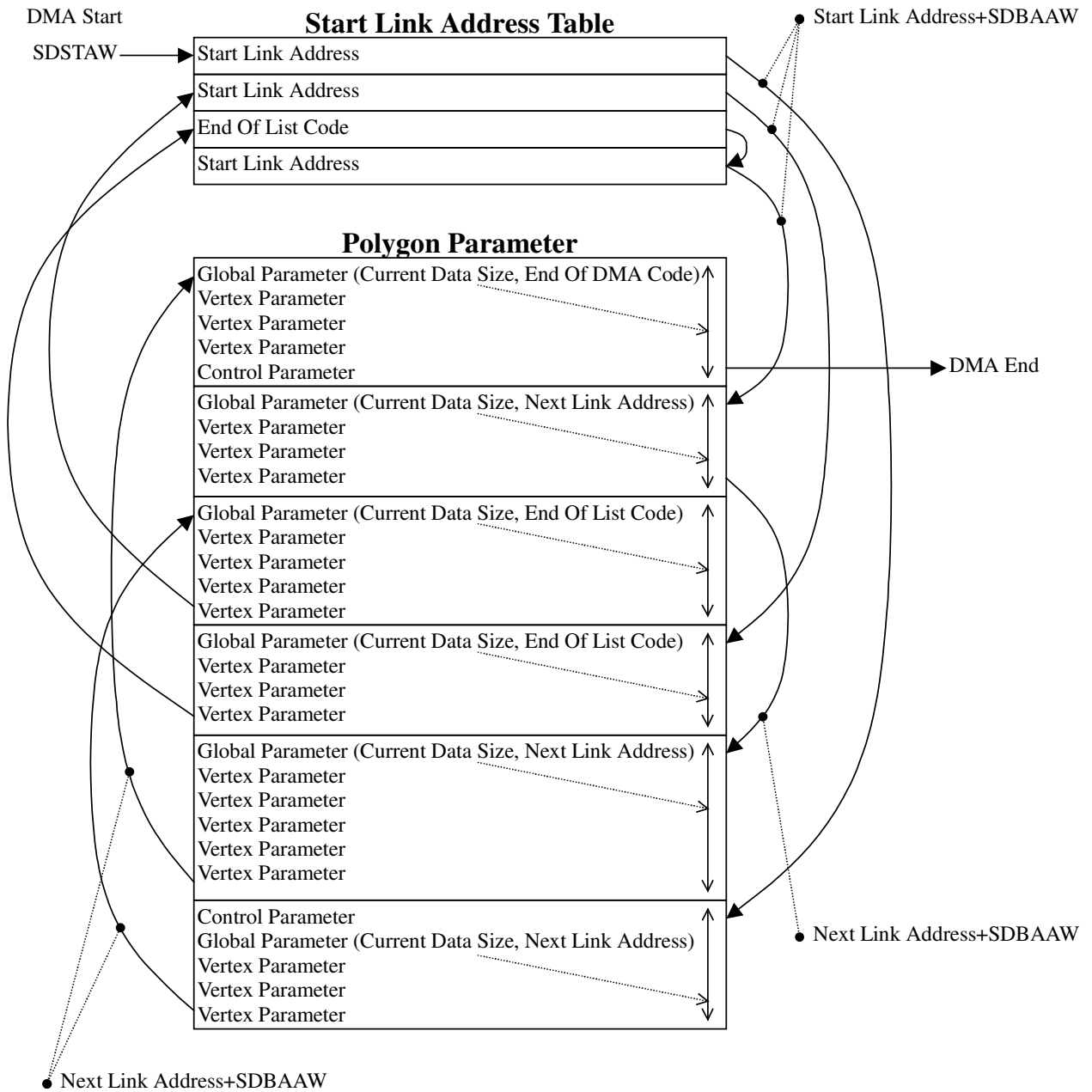


Fig. 2-14 Sort-DMA Transfer Example

## § 2.6.6 Wave Data Transfers

In the Dreamcast System, wave memory is allocated to a 2MB space (0x00800000 to 0x009FFFFFFF) in the G2 Bus area. There are four types of DMA on the G2 Bus: DMA0 (AICA-DMA), DMA1 (External-DMA1), DMA2 (External-DMA2), and DMA3 (Debug-DMA). All of these types of DMA are functionally similar, but can be set and executed independently, except for a few common registers.

The common settings for G2-DMA are shown below, using DMA0 (AICA-DMA) as an example. Regarding DMA1 and 3 for expansion devices, etc., set registers for the AICA setting items for that device.

- (1) Load the AICA address in the SB\_ADSTAG register. (If an incorrect address is set, an illegal address interrupt is generated.) Valid addresses are specified by bits 28 through 5. (The high-order bits 31 to 29 and the low-order bits 4 to 0 are "0".) When setting other addresses in registers, set the highest three bits and the lowest five bits to "0".)
- (2) Load the root bus address in the SB\_ADSTAR register. (If an incorrect address is set, an illegal address interrupt is generated.)
- (3) Set the transfer size (in 32-byte units) in the SB\_ADLEN register.
- (4) Set the transfer direction in the SB\_ADDIR register.  
0: Root → G2  
1: G2 → Root
- (5) Set the initiation trigger in the SB\_ADTSEL register.  
0: CPU trigger  
- Software initiation  
1: HARD trigger  
- AICA (DMA0) when the buffer is empty. In other cases, DMA1 through DMA3 depend on the expansion device.  
2: INT trigger  
- Initiated when any interrupt for which "1" is set in the SB\_SBDTNRM or SB\_G2DTEXT register is received. INT initiation is possible with a variety of sources, requiring procedures that vary according to the interrupt that is being used.

### ■ CPU initiation (ex: DMA0)

- (6) Set "1" in the SB\_ADEN register. (If an incorrect address is set, an illegal address interrupt is generated.)  
0: Disable  
1: Enable
- (7) Set "1" in the SB\_ADST register. If an incorrect address is set, the transfer is not initiated.  
0: STOP  
1: START
- (8) There are two ways to confirm the end of a transfer:  
A. Confirm through the value in the SB\_ADST register.  
0: DMA end  
1: Not end  
B. Confirm the end through the G2DEAINT (AICA-DMA end) interrupt.



■ HARD initiation (ex: DMA0)

- (6) Set "1" in the SB\_ADEN register.  
0: Disable  
1: Enable - After setting "enable," execute the transfer right after the AICA buffer becomes empty.
- (7) End confirmation  
A. Confirm through the value in the SB\_ADEN register.  
0: DMA end  
1: Not end  
B. Confirm the end through the G2DEAINT interrupt.  
Note: Although this usage is the same for DMA1 through DMA3, the initiation source depends on the expansion device.

■ INT initiation (ex.: DMA0)

- (6) Set "1" for the bits corresponding to the interrupt sources in the SB\_G2DTNRM and SB\_G2DTEXT registers.
- (7) Set "1" in the SB\_ADEN register.  
0: Disable  
1: Enable
- (8) End confirmation  
A. Confirm through the value in the SB\_ADEN register. (The value is set to either "1" or "0" by the hardware.)  
0: DMA end  
1: Not end  
- However, due to the time lag in the operation of SB\_ADST, the DMA operation cannot be gauged accurately.  
B. Confirm the end through the G2DEAINT interrupt.

System memory area protection is set by setting 0x4659XXYY in the SB\_G2APRO register.

XX: Starting area where protection is disabled  
YY: Ending area where protection is disabled

System memory is allocated in 0xC0000000 to 0xFFFFFFFF, but 7 bits of XX and YY, respectively, are reflected in bits 26 to 20 of the above area addresses, indicating whether protection is enabled/disabled for the specified area.

To disable protection for the entire area, set 0x4659007F in the SB\_G2APRO register. To enable protection for the entire area, set 0x46597F00 in the same register.

- When a G2-DMA transfer is performed so that it spans a protected area, an overrun error is generated.
- Regarding the timeout setting registers, the values that are set in the SB\_G2DSTO and SB\_G2TRTO registers must satisfy the following relationship:  
$$SB\_G2DSTO \leq SB\_G2TRTO$$

[Supplement] Basically, changes are not possible.

For DMA transfers to wave memory, the type of transfers that are primarily used are data transfers to system memory and data transfers from the GD-ROM on the G1 bus.

The registers that are used for wave data DMA (wave DMA) include registers that are dedicated to wave DMA on the G2 Bus, and registers for interrupts that are shared with other types of DMA. An overview of the registers is provided below.

■ Wave DMA Dedicated Registers

**SB\_ADSTAG** 0x005F7800 : Wave memory start address setting  
Settable area: 0x00800000 to 0x009FFFE0

**SB\_ADSTAR** 0x005F7804 : System memory start address setting  
Settable area (Note: The setting in the System Memory Protection register is also referenced):  
0x0C000000 to 0x0FFFFFFE0

**SB\_ADLEN** 0x005F7808 : Transfer size setting  
Set in 0x20 (32-byte) units.  
The setting of bit 31 enables DMA initiation (SB\_ADEN) when a DMA transfer ends.  
0x00000000: Do not set DMA initiation enable setting to "o."  
0x80000000: Set DMA initiation enable setting to "o."

**SB\_ADDIR** 0x005F780C : Transfer direction setting  
0x00000000 : System memory to wave memory  
0x00000001 : Wave memory to system memory

**SB\_ADTRG** 0x005F7810 : DMA initiation method setting  
0x00000000 : Initiation by CPU  
0x00000002 : Initiation by interrupt

**SB\_ADEN** 0x005F7814 : DMA operation enable  
0x00000000 : DMA operation enabled  
0x00000001 : DMA operation disabled

**SB\_ADST** 0x005F7818 : DMA initiation by CPU  
0x00000000 : ---  
0x00000001 : DMA initiation

**SB\_G2APRO** 0x005F78BC : System memory access restriction setting (shared with other G2 devices)  
0x00000000 : ---  
0x00000001 : DMA initiation

**SB\_IST\*\*\*** 0x005F6900 to 0x005F6908 interrupt status registers

**SB\_IML\*\*\*** 0x005F6910 to 0x005F6938 interrupt mask registers

**SB\_G2DTNRM** 0x005F6950 : Wave DMA interrupt initiation setting 1  
(shared with other G2 devices)

**SB\_G2DTEXT** 0x005F6954 : Wave DMA interrupt initiation setting 2 (shared with other G2 devices)

(Settings 1 and 2 function as a pair.)

Examples of how to use wave DMA are provided below.

(Example 1)

**Transferring wave data (0x00000040 bytes) from wave memory to system memory**

Wave memory address:	0x00800000
System memory address:	0x0C001000

- (1) Set 0x4659007F in the SB\_G2APRO register, completely releasing the system memory protect setting.
- (2) Set 0x00000000 in the SB\_ADEN register, disabling DMA operations.
- (3) Set the wave memory address 0x00800000 in the SB\_ADSTAG register.
- (4) Set the system memory address 0x0C001000 in the SB\_ADSTAR register.
- (5) Set the transfer size (0x00000040: 64 bytes) in the SB\_ADLEN register.
- (6) Set the transfer direction (0x00000001: wave memory to system memory) in the SB\_ADDIR register.
- (7) Set the DMA initiation method (0x00000000: CPU trigger) in the SB\_ADTRG register.
- (8) Set "DMA enabled" (0x00000001) in the SB\_ADEN register.
- (9) Set 0x00000001 in the SB\_ADST register, initiating wave memory DMA.

\*1 When actually using wave DMA, it is necessary to set the System Memory Protection register (SB\_G2APRO).

\*2 Wave memory DMA loads the set value when operation is enabled (a "1" has been written to the SB\_ADEN register). Therefore, when overwriting the registers, always follow the procedure described below.

1. Disable DMA operation. (Wave DMA enable = 0)
2. Update the registers.
3. Enable DMA operation. (Wave DMA enable = 1)

\*3 When system memory access is restricted through the System Memory Protection register (SB\_G2APRO), some address settings may result in a DMA error (Illegal Address Error), causing the DMA transfer to end abnormally.

(Example 2)

**Transferring wave data (0x00000040 bytes) from system memory to wave memory, after having executed example 1**

System memory address:	0x0C001000
Wave memory address:	0x00800040

- (1) Set 0x00000000 in the SB\_ADEN register, disabling DMA operations.
- (2) Set the wave memory address 0x00800040 in the SB\_ADSTAG register.
- (3) Set the transfer direction (0x00000000: system memory to wave memory) in the SB\_ADDIR register.
- (4) Set "DMA enabled" (0x00000001) in the SB\_ADEN register.
- (5) Set 0x00000001 in the SB\_ADST register, initiating wave memory DMA.

\* Because the initial values in the registers are maintained after DMA is completed, the following registers do not change and therefore do not need to be overwritten:

SB\_ADSTAR(system memory address: 0x0C001000)  
SB\_ADLEN(64-byte transfer size: 0x00000040)  
SB\_ADTRG(DMA initiation method - CPU trigger: 0x00000000)

(Example 3)

**Initiating DMA through an interrupt signal from AICA, transferring wave data (0x00000040 bytes) from wave memory to system memory**

Wave memory address:	0x00800000
System memory address:	0x0C001000

- (1) Set 0x00000000 in the SB\_ADEN register, disabling DMA operations.
- (2) Set the wave memory address 0x00800000 in the SB\_ADSTAG register.
- (3) Set the system memory address 0x0C001000 in the SB\_ADSTAR register.
- (4) Set the transfer size (0x00000040: 64 bytes) in the SB\_ADLEN register.
- (5) Set the transfer direction (0x00000001: wave memory to system memory) in the SB\_ADDIR register.
- (6) Set the DMA initiation method (0x00000003: interrupt trigger) in the SB\_ADTRG register.
- (7) Set "DMA enabled" (0x00000001) in the SB\_ADEN register.
- (8) Set initiation by interrupt through the G2DTNRM register and the G2DTEXT register.

\*1 Because this transfer operation is initiated by interrupt, the following register does not need to be set:

SB\_ADST(DMA initiation: 0x00000001)

\*2 In this example, wave DMA is initiated when the AICA interrupt signal is input, but it is also necessary to make settings for AICA that will generate the interrupt.

\*3 When initiating DMA through an interrupt, if the interrupt from AICA is generated immediately after a "1" is written to the SB\_ADEN (DMA operation enable) register, wave DMA might not be initiated. This is because the time at which the settings in the wave DMA registers become effective differs from the time at which the setting in the register that enables initiation by an interrupt becomes effective. In order to prevent this from happening, it is necessary to coordinate the timing of the settings by, for example, reading the SB\_G2ID register (which returns the G2 Bus version information) after setting up wave DMA, and then setting the interrupt-related registers.

### **§ 2.6.7 ARM Data Transfers**

ARM data transfers are DMA transfers programs and data for the ARM, the AICA's internal processor, to wave memory, and are basically similar to G2-DMA DMAo (AICA-DMA) transfers.

## § 2.6.8 Peripheral Data Transfers

The registers that are required for DMA transfers of peripheral data and the procedure for setting up the command file for the controller (Maple-Host) are described in this section.

Because only the minimum requirements in terms of the registers and the procedure for DMA transfers of peripheral data are described below, refer to section 5, "User Interface," if more details are required.

<Registers used for Maple-DMA>

- SB\_MDSTAR** 0x005F6C04: Starting address setting for the command table in system memory  
Settable area: 0x0C000000~0x0FFFFFFE0
- SB\_MDTSEL** 0x005F6C10: Maple-DMA trigger setting  
0x00000000 : Software trigger  
0x00000001 : Hardware trigger
- SB\_MDEN** 0x005F6C14: Enables Maple-DMA  
(Read)  
0x00000000 : Disable  
0x00000001 : Enable  
(Write)  
0x00000000 : Disable  
0x00000001 : Enable
- SB\_MDST** 0x005F6C18: Maple-DMA software start  
(Read)  
0x00000000 : Maple-DMA end  
0x00000001 : Maple-DMA transfer in progress  
(Write)  
0x00000000 : Invalid  
0x00000001 : Maple-DMA start
- SB\_MSYS** 0x005F6C80: Maple system control setting  
For details, refer to section 8.4.1.1, "System Registers."
- SB\_MDAPRO** 0x005F6C8C: Maple-DMA area protection setting  
Settable area: 0x0C000000 to 0x0FFFFFFE0
- SB\_ISTNRM** 0x005F6900: Normal interrupt status  
bit12: Maple-DMA end  
For details on interrupt registers, refer to section 8.4.1.1, "System Registers."

The procedure is described through the use of an example below. (CPU initiation for Maple, 4 port access, interrupts not used)

- (1) Set 0x00001000 in the SB\_ISTNRM register to clear the Maple-DMA end status.
- (2) Set 0x00000000 in the SB\_MDEN register to disable Maple-DMA.
- (3) Read the SB\_MDST register, and confirm that DMA operation is not in progress (0x00000000).
- (4) Set the SB\_MDSYS register. (0xC3500000: timeout 1ms, transfer rate 2Mbps)
- (5) Set the initiation trigger in the SB\_MDSEL register. (0x00000000: Triggered from CPU)
- (6) Set the accessible area in system memory in the SB\_MDAPRO register. (0x6155007F: access range 0x80000000 to 0xFFFFFEE0)
- (7) Set up the following command file in system memory.

(Address)	(Data)	
0xC700000	→	0x00000000 Port 0, 4-byte data transmission (instruction to Maple-Host)
0xC700004	→	0xC800000 Port 0, reception data storage address (instruction to Maple-Host)
0xC700008	→	0x01200000 [Device Request], transfer destination AP: 0x20, transfer source AP: 0x00
0xC70000C	→	0x00010000 Port 1, 4-byte data transmission
0xC700010	→	0xC800100 Port 1, reception data storage address
0xC700014	→	0x01604000 [Device Request], transfer destination AP: 0x60, transfer source AP: 0x40
0xC700018	→	0x00020000 Port 2, 4-byte data transmission
0xC70001C	→	0xC800200 Port 2, reception data storage address
0xC700020	→	0x01A08000 [Device Request], transfer destination AP: 0x80, transfer source AP: 0xA0
0xC700024	→	0x80030000 Port 3, 4-byte data transmission
0xC700028	→	0xC800300 Port 3, reception data storage address
0xC70002C	→	0x01E0C000 [Device Request], transfer destination AP: 0xC0, transfer source AP: 0xE0

- (8) Set the starting address of the command file (0xC700000 in this example) in the SB\_MDSTAR register.
- (9) Set 0x00000001 in the SB\_MDEN register to enable Maple-DMA.
- (10) Write 0x00000001 in the SB\_MDST register to initiate Maple-DMA (software initiation).

After executing steps (1) through (10) above and confirming that bit 12 in the SB\_ISTNRM register is "1" (DMA end), the data that was received can be used to confirm the connection, or that there is no connection, or that an error occurred.

(Specified reception data storage address: 0xC800000)

0xC800000	→ 0x0500201C	[Device Status], transfer destination AP: 00, transfer source AP: 20
0xC800004	→ 0x00000001	112 bytes of fixed data follows
⋮	⋮	
0xC800070	→ 0x00000000	
0xC800000	→ 0xFFFFFFFF	No connection
0xC800000	→ 0xFFFFFFFF	Reception data error

After confirming the device status through the received data described on the previous page, the trigger data can be acquired by using "Get Condition."

Because Maple is initialized through steps (1) through (10) on the previous page, the trigger data can be acquired by changing the command file and by initiating Maple-DMA.

---

- (1) Set 0x00001000 in the SB\_ISTNRM register to clear the Maple-DMA end status.
- (2) Set up the following command file in system memory.

(Address)	(Data)
0x0C700000	→ 0x00000001 Port 0, 8-byte data transmission (instruction to Maple-Host)
0x0C700004	→ 0x0C800000 Port 0, reception data storage address (instruction to Maple-Host)
0x0C700008	→ 0x09200001 [Get Condition], transfer destination AP: 0x20, transfer source AP: 0x00
0x0C70000C	→ 0x00000001 Function type
0x0C700010	→ 0x00010001 Port 2, 8-byte data transmission
0x0C700014	→ 0x0C800100 Port1, reception data storage address
0x0C700018	→ 0x09604001 [Get Condition], transfer destination AP:0x60, transfer source AP:0x40
0x0C70001C	→ 0x00000001 Function Type
0x0C700020	→ 0x00020001 Port 2, 8-byte data transmission
0x0C700024	→ 0x0C800200 Port 2, reception data storage address
0x0C700028	→ 0x09A08001 [Get Condition], transfer destination AP: 0x80, transfer source AP: 0xA0
0x0C70002C	→ 0x00000001 Function type
0x0C700030	→ 0x80030001 Port 3, 8-byte data transmission, command list end
0x0C700034	→ 0x0C800300 Port 3, reception data storage address
0x0C700038	→ 0x09E0C001 [Get Condition], transfer destination AP: 0xC0, transfer source AP: 0xE0
0x0C70003C	→ 0x00000001 Function Type
- (3) Write 0x00000001 in the SB\_MDST register to initiate Maple-DMA (software initiation).

After executing steps (1) through (3) above and confirming that bit 12 in the SB\_ISTNRM register is "1" (DMA end), the data that was received can be used to confirm the connection, or that there is no connection, or that an error occurred.

(Specified reception data storage address: 0x0C800000)

0x0C800000	→ 0x0500201C	[Device Status], transfer destination AP:00, transfer source AP:20
0x0C800004	→ 0x00000001	
0x0C800008	→ 0xFFFF0000	Upper 16 bits: Digital trigger; lower 16 bits:
0x0C800070	→ 0x33008080	Lower 16 bits: Analog 2ch
0x0C800000	→ 0xFFFFFFFF	No connection
0x0C800000	→ 0xFFFFFFF0	Reception data error

Once the trigger data has been acquired through the above sequence, the data can be acquired repeatedly through just the following procedure:

- (1) Set 0x00001000 in the SB\_ISTNRM register to clear the Maple-DMA end status
- (2) Write 0x00000001 in the SB\_MDST register to initiate Maple-DMA (software initiation).
- (3) Check received data.

## § 2.6.9 Color Palette Transfers

When using DMA to transfer data from system memory to palette RAM, the required values must be set in the following registers:

- |     |  |              |          |
|-----|--|--------------|----------|
| (1) | SB_PDSTAP  | (0x005F7C00) | register |
|     | Palette RAM transfer start address (SH4 address) |              |          |
| (2) | SB_PDSTAR  | (0x005F7C04) | register |



- System memory transfer start address (SH4 address)
- |     |   |              |
|-----|---|--------------|
| (3) | SB_PDLEN  | (0x005F7C08) |
|     | Specify the number of transfer bytes in units of 0x20 bytes.  |              |
| (4) | SB_PDDIR  | (0x005F7C0C) |
|     | Specify the transfer direction. Write "0" (system memory to palette RAM).   |              |
| (5) | SB_PDTSEL   | (0x005F7C10) |
|     | Specify the DMA initiation source. This is always "0"   |              |
| (6) | SB_PDEN   | (0x005F7C14) |
|     | Set DMA enable to "1."  |              |
| (7) | SB_PDST   | (0x005F7C18) |
|     | DMA starts when register setup is complete, the SB_PDEN register is "1" and a "1" is written to this register. This register also functions as a DMA status register. (0: DMA is in standby; 1: DMA is in progress) |              |

Regarding the end of DMA: if DMA ends normally, bit 11 of the SB\_ISTNRM (0x005F6900) register is set to "1" and an interrupt is generated.

In addition, the SB\_PDST register indicates the DMA status; when DMA ends, the value of this register returns to "0." In this case, the value in the SB\_PDEN register remains "1."

Regarding DMA errors: if the DMA address in system memory moves beyond the allowable memory range during a DMA operation, an overrun interrupt is generated and that DMA operation is forcibly terminated. In this case, bit 7 of the SB\_ISTERR (0x005F6908) register is set to "1." Furthermore, if the address in system memory or on the PVR side was incorrectly set outside of the allowable memory range, an illegal address interrupt is generated and bit 6 of the SB\_ISTERR (0x005F6908) register is set to "1." These interrupts are generated both when the incorrect DMA address is set, and when an attempt is made to initiate DMA with such an incorrect DMA address.

Cautions during DMA operations: If the SB\_PDSTAP, SB\_PDSTAR, SB\_PDLEN, SB\_PDDIR, or SB\_PDTSEL register is overwritten while a DMA operation is in progress, the new setting has no effect on the current DMA operation. Once the current DMA is terminated and the next DMA is initiated (SB\_PDEN = 1 and SB\_PDST = 1), the values in these five registers are retrieved. A DMA operation that is currently in progress can be forcibly terminated by writing a "0" in the SB\_PDEN register. If an access is in progress when this happens, the value in the SB-PDST register returns to "0" as soon as the access terminates.

#### **§ 2.6.10 External Data Transfer**

This type of DMA transfer is for expansion devices connected to the G2 bus. The details are similar to those of other G2-DMA transfers.

## § 2.7 Interrupts

### § 2.7.1 Overview

The following are the main interrupt sources for the SH4:

- NMI interrupts
- JTAG interrupts
- SH4 external interrupts

Of these, the NMI and JTAG interrupts are controlled by the debugging adapter, which is an external expansion device that manipulates the system reset signal, NMIs, etc., and is used as a software development tool. Other external interrupts that are sent to the SH4 are all controlled by HOLLY, the graphics/interface core.

Interrupt processing within HOLLY is described below.

The graphics/interface core HOLLY includes an interrupt controller that collects interrupts that originate within and outside of the chip. HOLLY accepts interrupts from an internal and external devices, outputs interrupts to the SH4, and generates DMA start signals for the PVR block and devices on the G2 Bus (G2 devices). Of the SH4's interrupt input IRL[3:0], IRL1 and 2 are used for the interrupts that HOLLY outputs to the SH4. The interrupt outputs have four priority levels (including "no interrupt"), and can be associated with any desired interrupt source by making the appropriate register settings. In addition, any desired interrupt source (other than error interrupts, described later) can be associated with the PVR block and G2 device DMA start signals by making the appropriate register settings.

Interrupt sources are divided into the following three types:

- Normal interrupts: 22\*~~(in the HOLLY2 specifications; 21 in the HOLLY1 specifications)~~
- External interrupts: 4
- Error interrupts: 32

Each interrupt signal is processed according to the source type. (Refer to section 8.5.2 for a list of sources and descriptions.)

Normal interrupt and error interrupt input can be confirmed and cleared through the *SB\_ISTNRM* and *SB\_ISTERR* registers, which indicate the status of interrupts of their respective types, by checking the bits assigned to each particular interrupt. External interrupts are interrupt signals from external devices (GD-ROM, AICA, modem, or expansion device), and each latched signal can be checked in *SB\_ISTEXT*, which is the register that indicates the status of external interrupts, by checking the bits assigned to each interrupt. Note that external interrupts cannot be cancelled through this register; external interrupts must be cancelled directly through the corresponding external device.

Interrupt masks are normally set through mask control registers (*SB\_IML2NRM*, *SB\_IML2EXT*, *SB\_IML2ERR*, *SB\_IML4NRM*, *SB\_IML4EXT*, *SB\_IML4ERR*, *SB\_IML6NRM*, *SB\_IML6EXT*, and *SB\_IML6ERR*) for each level of each type of interrupt source: normal, external, or error. If a bit assigned to a source in these registers is set to "1" and an interrupt is received from the corresponding interrupt source, the corresponding interrupt is generated for the SH4. If a bit in these registers is set to "0," output of that interrupt to the SH4 is disabled. These registers have priority over the *SB\_ISTNRM*, *SB\_ISTERR*, and *SB\_ISTEXT* registers; in addition, masking occurs regardless of the timing by which the interrupt was generated. This means that, for example, if a bit for an interrupt that is currently being generated is masked, and then the mask is released while the interrupt is still being generated, the same interrupt might be generated again.

Masked signals are assigned a priority at the level encoding stage, and are output as interrupt signals to the SH4. The order of priority is level 6 > level 4 > level 2. If there are no applicable sources, interrupt processing is not generated.

The DMA start signal is masked by the SB\_PDTNRM and SB\_PDTEXT registers on the PVR side, and by the SB\_G2DTNRM and SB\_G2DTEXT registers on the G2 side. Except for the fact that there are no error interrupt registers and that there is only one level, these registers mask their interrupts in the same manner as interrupts to the SH4 are masked.

For details on interrupt-related registers, refer to section 8.4.1.1.

## § 2.7.2 Interrupt Settings and Access Procedures

Specific procedures are necessary when modifying register-related interrupts to Holly. If these procedures are not followed, jumps to interrupt routines may occur with no value set in the INTEVT register, or interrupts that were presumed to have been canceled may be received again by mistake.

### •**Procedure 1 (normal case)**

Use the following steps (1) to (4) to modify Holly register-related interrupts.

- (1) For CPU processing, mask the objective interrupt using one of the following methods:
  - (1a) Execute SR.IMASK to set a priority higher than the objective interrupt, or
  - (1b) Set SR.BL to 1.
- (2) Modify the Holly register-related interrupts as needed (if multiple modifications are needed, make them all at once here).
- (3) Read the modified Holly register twice.
- (4) Remove the mask applied in step (1).

### •**Procedure 2 (when canceling interrupts from external devices)**

When canceling the interrupts from external devices that are controlled by Holly, special steps are required. There are four types of interrupts, from CD-ROM, AICA, modem and G2 expansion devices, with the following related registers: ISTEXT, IML2EXT, IML4EXT and IML6EXT. Steps (1) to (4) are the same as Procedure 1 above.

- (1) For CPU processing, mask the objective interrupt using one of the following two methods:
  - (1.a) Execute SR.IMASK to set a priority above that of the objective interrupt, or
  - (1.b) Set SR.BL to 1.
- (2) Access the interrupt control register of the external device (CD-ROM, AICA, modem or G2 expansion device) and cancel the interrupt (if multiple interrupts are to be canceled, cancel them all at once here).
- (2.5) Read the ISTEXT register (external interrupt status) and confirm that the interrupt was canceled (if multiple interrupts have been canceled in (2), confirm each interrupt).
- (3) Read the ISTEXT register twice.
- (4) Remove the mask applied in step (1).

### •**Supplementary Issues**

- (a) The previous procedures are required when masked interrupts are not occurring. In other cases, errors should not occur if the above procedures are not followed.

<Required>	When canceling the status of an unmasked (valid) interrupt.
<Not Required>	When canceling the status of a masked (invalid) interrupt.
<Required>	When an interrupt is masked (to disable the interrupt).
<Not Required>	When canceling an interrupt mask (to re-enable the interrupt).
- (b) Normally (when not set intentionally), SR.BL=1 during an interrupt processing routine, so when modifying a register within the interrupt routine, steps (1) and (4) are not required. However, even within the interrupt routine, when SR.BL=0 and multiple interrupts are enabled, the procedures must be followed. Outside of the interrupt routine, steps (1) through (4) must be followed (of course, if either (1a) or (1b) is satisfied, there is no need for steps (1) and (4)).

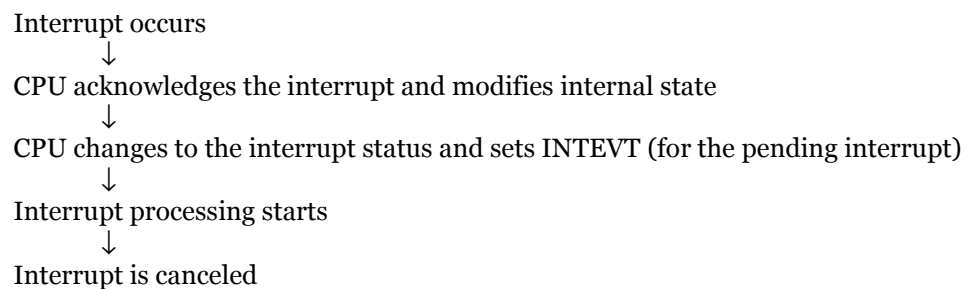
- (c) Step (3) consists of dummy reads to be executed between steps (2) and (4). It ensures sufficient processing time for step (2). When continuously executing step (2), which externally accesses the CPU, and step (4), which internally accesses the CPU, step (4) may be executed in any order. Also, several clocks are required if the Holly status resulting from step (2) affects the internal CPU state.
- (d) Failing to set the INTEVT register results in an INTEVT register value of 0.
- (e) The dummy reads in step (3) must be executed until the interrupt is available to execute step (4) or the RTE command. However, two reads are not required for every single change: they are required only to allow the interrupts to be received after a modification has been made. Other types of accesses may be mixed without problem.

### •**Related Issues**

- (a) The Holly interrupt-related registers are as follows:

ISTNRM	(0xA05F6900)	normal interrupt status
ISTEXT	(0xA05F6904)	external interrupt status
ISTERR	(0xA05F6908)	error interrupt status
IML2NRM	(0xA05F6910)	Level2 normal interrupt mask control
IML2EXT	(0xA05F6914)	Level2 external interrupt mask control
IML2ERR	(0xA05F6918)	Level2 error interrupt mask control
IML4NRM	(0xA05F6920)	Level4 normal interrupt mask control
IML4EXT	(0xA05F6924)	Level4 external interrupt mask control
IML4ERR	(0xA05F6928)	Level4 error interrupt mask control
IML6NRM	(0xA05F6930)	Level6 normal interrupt mask control
IML6EXT	(0xA05F6934)	Level6 external interrupt mask control
IML6ERR	(0xA05F6938)	Level6 error interrupt mask control

- (b) This procedure is required for internal CPU interrupt processing, as described in the hardware manual (section 19.2.3). In that case, only one dummy read is required in step (3).
- (c) Error causes are as follows (for reference).
  - 1) Timing between "interrupt acknowledge" and "set INTEVT" processes: normal processing is as follows:



However, if timing is such that the interrupt is canceled before INTEVT has been set, there is no interrupt to refer to, so INTEVT is set incorrectly, as follows:

```
Interrupt occurs
↓
CPU acknowledges the interrupt and modifies internal state
↓
Interrupt is canceled
↓
CPU changes to the interrupt status and sets INTEVT (for the pending interrupt)
<but there is no pending interrupt at this point!>
↓
Interrupt processing starts
```

The procedures described previously prevent the interrupt being canceled between the two processes (interrupt acknowledgement and INTEVT setting).

2) Also, when the following interrupt routine finishes:

```
MOV.LRo,@R1 ; write to cancel interrupt
RTE
NOP
```

While the interrupt routine is being processed, the interrupt processing could be re-entered because the interrupt has not been canceled, or, even though it may have been canceled within Holly, the cancel has not been issued to the CPU. Therefore time must be allotted for the CPU to enter the interrupt cancel status.

```
MOV.LRo,@R1 ; write to cancel interrupt
MOV.L@R1, Ro ; dummy read 1
MOV.L@R1, Ro ; dummy read 2
RTE
NOP
```

The methods for using and accessing each register are described below.

#### **SB\_ISTNRM (0x005F 6900) normal interrupt status**

This register is used to confirm and cancel normal interrupts. When a normal interrupt is generated internally by Holly, the corresponding bit in this register is set to "1." In addition, any of these interrupts can be cancelled (set to "0") by writing a "1" to the corresponding bit. Note that the two highest bits indicate the OR'ed result of all of the bits in SB\_ISTEXT and SB\_ISTERR, respectively, and writes to these two bits are ignored.

Example 1:

```
read 0x005F 6900 -> 0x0001 0080 G2DE1INT and TAEOINT are being
generated.
```

Example 2:

```
read 0x005F 6900 -> 0x8000 0000 An error interrupt is being generated.
read 0x005F 6908 -> 0x0000 0100 MIAINT is being generated.
write 0x005F 6908 <- 0xFFFF FFFF Cancels all error interrupts.
read 0x005F 6908 -> 0x0000 0000 MIAINT is now cancelled.
read 0x005F 6900 -> 0x0000 0000 The error interrupt image in this register is
now cancelled.
```

Example 3:

```
read 0x005F 6900 -> 0x0000 0030 PCVOINT and PCHIINT are being generated.
write 0x005F 6900 <- 0x0000 0020 Cancels PCHIINT.
read 0x005F 6900 -> 0x0000 0010 Only PCHIINT is cancelled.
```

#### **SB\_ISTEXT (0x005F 6904) external interrupt status**

This register is used to confirm external interrupts. This register is read-only. When an external interrupt is generated by a GD-ROM, AICA, modem, or expansion device, the corresponding bit in this register is set to "1." Note that these interrupts can be cancelled only by canceling the interrupt output directly at the generating source; they cannot be cancelled through this register.

Example:

```
read 0x005F 6904 -> 0x0000 0005 G2MDMINT and G1GDINT are being
generated.
```

#### **SB\_ISTERR (0x005F 6908) error interrupt status**

This register is used to confirm and cancel error interrupts. When an error interrupt is generated, the corresponding bit in this register is set to "1." In addition, any of these interrupts can be cancelled (set to "0") by writing a "1" to the corresponding bit.

Example:

```
read 0x005F 6908 -> G2IAAINT and G1IAINT are being generated.
0x00009000 Cancels G2IAAINT.
write 0x005F 6908 <- Only G2IAAINT is cancelled.
0x00008000 In the meantime, TAINPINT has been
generated as a new interrupt.
```

#### **SB\_IML2NRM (0x005F 6910) Level-2 normal interrupt mask control**

#### **SB\_IML4NRM (0x005F 6920) Level-4 normal interrupt mask control**

#### **SB\_IML6NRM (0x005F 6930) Level-6 normal interrupt mask control**

These registers enable/disable (mask) normal interrupts for the SH4. When a bit is set to "1," the corresponding interrupt is generated for the SH4. This register is a read/write register. Priority is assigned to each interrupt according to their level, with level 6 being the highest priority. These registers mask interrupts without regard to the timing of the signal from the source that is generating the interrupt.

Example 1:

```
Read 0x005F 6910 -> 0x0000 No normal interrupts are set for level 2.
0000 Sets DTDE2INT as a level 2 interrupt.
```

Example 2:

```
write 0x005F 6920 <- 0x0000 Sets PCVOINT and PCVIINT as level 4
0018 interrupts.
Sets PCVOINT as a level 6 interrupt.
write 0x005F 6930 <- 0x0000 Generates a level 6 interrupt.
0010 Masks PCVOINT. Generates a level 4
CR_vout_n has been generated interrupt.
write 0x005F 6930 <- 0x0000 Cancels the PCVOINT interrupt. The level 4
interrupt is cancelled.
```

#### **SB\_IML2EXT (0x005F 6914) Level-2 external interrupt mask control**

#### **SB\_IML4EXT (0x005F 6924) Level-4 external interrupt mask control**

#### **SB\_IML6EXT (0x005F 6934) Level-6 external interrupt mask control**

These registers enable/disable (mask) external interrupts for the SH4. For details on how to use these interrupts, refer to the explanation for *SB\_IML2NRM*.

**SB\_IML2ERR (0x005F 6918)      Level-2 error interrupt mask control**

**SB\_IML4ERR (0x005F 6928)      Level-4 error interrupt mask control**

**SB\_IML6ERR (0x005F 6938)      Level-6 error interrupt mask control**

These registers enable/disable (mask) error interrupts for the SH4. for details on how to use these interrupts, refer to the explanation for *SB\_IML2NRM*.

**SB\_PDTNRM (0x005F 6940)      PVR-DMA trigger select from normal interrupt**

**SB\_PDTEXT (0x005F 6944)      PVR-DMA trigger select from external interrupt**

These interrupts are set when using interrupts as triggers for initiating DMA to the PVR. By setting a bit to "1," the corresponding interrupt signal can be used as a trigger for initiating DMA. These registers are read/write registers. Note that the DMA settings must have been made on the PVR side in order to actually initiate DMA.

Example:

```
write 0x005F 6940 <- 0x0000    Sets MVOINT as a PVR-DMA trigger.
```

**SB\_G2DRNRM (0x005F 6950)      G2-DMA trigger select from normal interrupt**

**SB\_G2DREX (0x005F 6954)      G2-DMA trigger select from external interrupt**

These interrupts are set when using interrupts as triggers for initiating DMA to a G2 device. By setting a bit to "1," the corresponding interrupt signal can be used as a trigger for initiating DMA. These registers are read/write registers. Note that the DMA settings must have been made on the G2 device side in order to actually initiate DMA.

Example:

```
read  0x005F 6954 -> 0x0000    Sets G1GDINT as a G2-DMA trigger.
0001
write 0x005F 6954 <- 0x0000    Changes the G2-DMA trigger to G2AICINT.
G2AICINT is now the G2-DMA trigger.
```



### § 2.7.3 Notes Concerning Interrupts

The following SH4 values require special attention when using interrupts.

- BL bit (bit 28 of the SH4's SR register)
  - 0: Interrupts enabled
  - 1: Interrupts disabled

This bit is set to "1" when the SH4 accepts an interrupt. When a large number of interrupts are generated or interrupt processing is completed, the interrupt processing routine must set this bit back to "0."

- IMASK bit (bits 7 through 4 of the SH4's SR register)
  - Acceptance level setting
  - Interrupt levels of the level that is set or lower are masked.

[7654]	Accepted levels
000x	NMI/6/4/2
001x	NMI/6/4
010x	NMI/6
011x	NMI
100x	NMI
:	:
111x	NMI

- SH4 VBR register
  - Executes a JMP to the address VBR + 0600h when an interrupt is generated. (PC <= VBR + 0x0600)
- INTEVT (address 0xFF000028) (32-bit access r/(w))
  - bit11-0
  - Stores a value that corresponds to the level of the interrupt that was accepted when an interrupt is generated.
  - 0x01C0: NMI
  - 0x0320: level6
  - 0x0360: level4
  - 0x03A0: level2

Others

- ICR (address 0xFFD0 0000) (16-bit access r/w)
  - IRLM(bit7)
  - Set to "0." (initial value)

## **§ 3 The Graphics System**

The Dreamcast graphics system consists of the graphics/interface chip HOLLY, which adopts the Power VR architecture, and its peripheral texture memory. The explanation below focuses primarily on HOLLY.

~~\*There are two versions of the HOLLY chip for Dev.Box, HOLLY1, and HOLLY2, which has additional functions added onto HOLLY1. In this section, a dotted line will be used to indicate descriptions that apply to HOLLY2.~~

## § 3.1 Overview

### § 3.1.1 Graphics Architecture

#### § 3.1.1.1 Basic Polygons

HOLLY supports three basic polygon shapes:

- Single Triangle polygons
- Single Quad polygons
- Stripped Triangle polygons

The Z, U, and V coordinate values of the fourth vertex of a Quad polygon and the Shading Color values are derived automatically from polygon surface equations that are calculated internally by the hardware. In addition, strip triangle polygons are supported for infinite strips. The sequencing and linking of each of the polygon vertices are illustrated below.

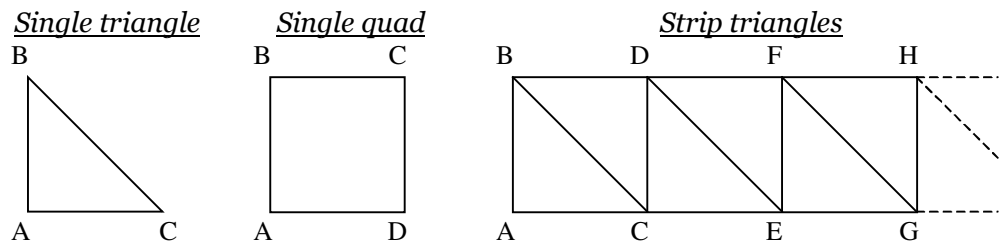


Fig. 3-1

In addition, HOLLY supports six polygon types:

- Non-Textured Flat Shaded
- Non-Textured Gouraud Shaded
- Textured Flat Shaded
- Textured Gouraud Shaded
- Textured Flat Shaded with Offset Color
- Textured Gouraud Shaded with Offset Color

Shading Color includes two data elements, "Base Color" and "Offset Color." The equation that determines the Shading Color on the basis of this data is specified by the control bit (Texture/Shading Instruction: refer to section 3.7.9.2) in the polygon parameters. Basically, the Base Color specifies the shading value for each vertex, and the Offset Color specifies the specular value for each vertex.

### § 3.1.1.2 Coordinate System

The coordinates that are specified for HOLLY are specified in terms of the screen coordinate system. An example of coordinate calculation is shown below.

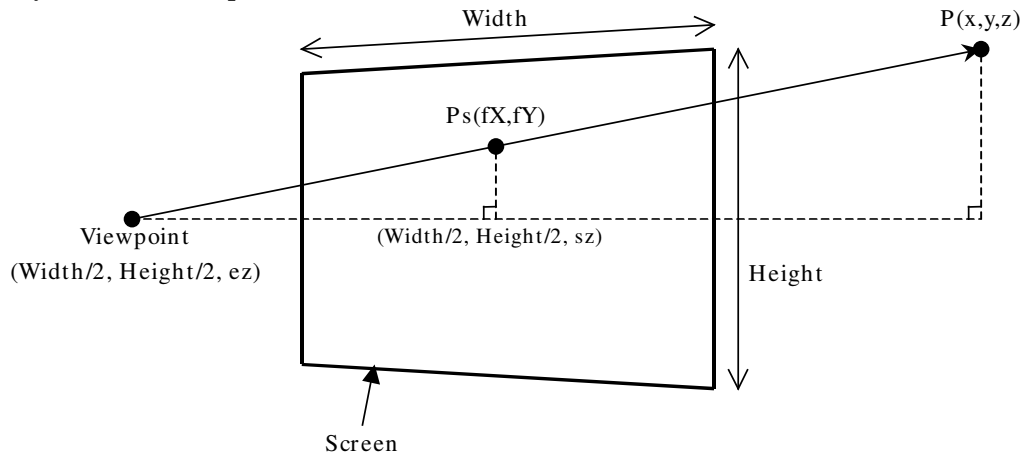


Fig. 3-2

The coordinate values (fX, fY, fInvW) that are passed to the hardware in order to specify the coordinates of point P (x, y, z) in the above diagram are calculated as follows:

$$\begin{aligned} fInvW &= (ez - sz) / (ez - z) \\ fX &= x \times fInvW \\ fY &= y \times fInvW \end{aligned}$$

However, it is not necessary to multiply the UV coordinate value of the texture by fInvW.

### § 3.1.1.3 Display List

There are two HOLLY graphics blocks, one called the "Tile Accelerator (TA)," which assists in generating display lists, and one called the "CORE," which handles drawing functions. Polygon lists for drawing include the "TA parameters," which are input from the CPU to the TA, and the "CORE display list," which the CORE uses when drawing the graphics. The TA block converts the input TA parameters into the CORE display list, which is then automatically stored in the specified area in texture memory. The CORE block uses the CORE display list and texture data in texture memory to draw the polygons, and then stores the screen data in the frame buffer in texture memory.

Normally, the TA parameters are the polygon list that is prepared by the application. Strictly speaking, however, the application must also prepare a portion of the CORE display list. (Refer to section 3.7.)

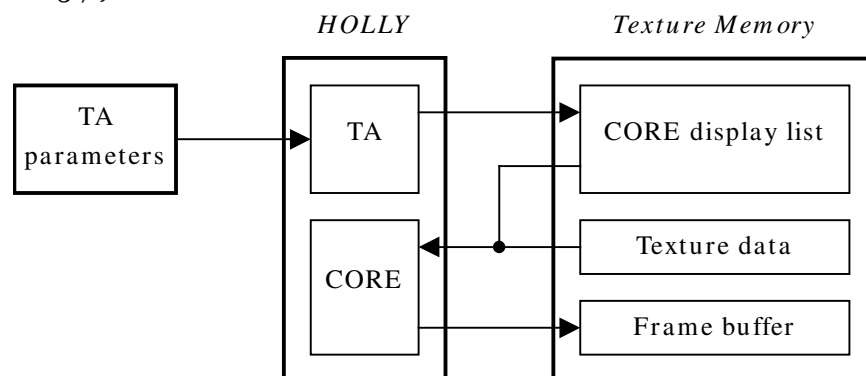


Fig. 3-3

### § 3.1.1.4 Tile Partitioning and Surface Equations

HOLLY feature two graphics architectures: Tile partitioning and polygon surface equations.

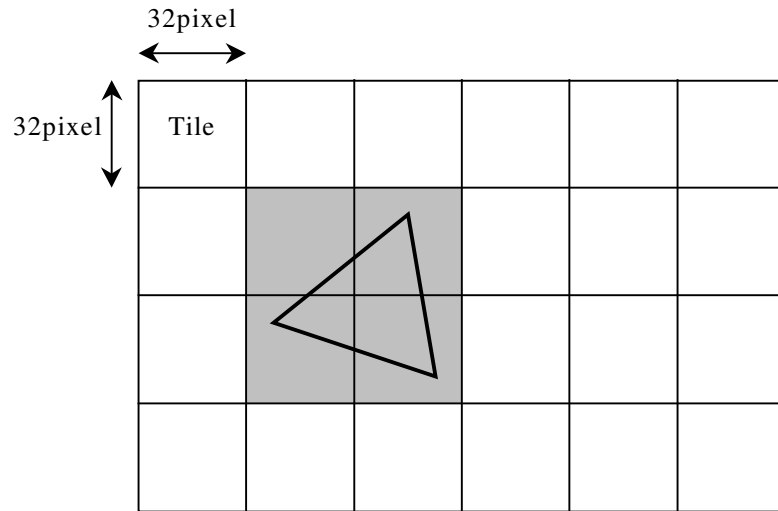


Fig. 3-4 Tile Partitioning

With Tile partitioning, a graphics screen of up to 2048 pixels x 2048 pixels is divided into Tiles that are 32 pixels by 32 pixels. The graphics processing is then performed on these individual Tiles. When drawing a given polygon, that polygon is registered in a list, called the "Object List," which indicates in which Tiles that polygon exists. When polygons are drawn, only those polygons that are registered in the lists that correspond to each Tile are drawn. Because this processing is all performed by the hardware known as the "Tile Accelerator (TA)," applications do not need to be aware of the Tile partitions; they only need to send the vertex data for the triangle or Quad polygon to the Tile Accelerator. The hardware then solves the surface equation  $Ax + By + C$  using the coordinates for three vertices of the registered polygon and draws the pixels.

These two architectures offer a variety of benefits, and permit drawing through Tiles even without enough space for an entire screen in the Z buffer or the frame buffer. In addition, texturing and shading processing is only performed on those pixels within the Tile that are visible. On an actual screen, there are many pixels that are hidden by objects that are closer to the foreground, and processing speed is markedly improved by not performing texturing and shading processing on such pixels.

### § 3.1.1.5 Block Diagram

A block diagram of the CORE block, which handles graphics processing, is shown below.

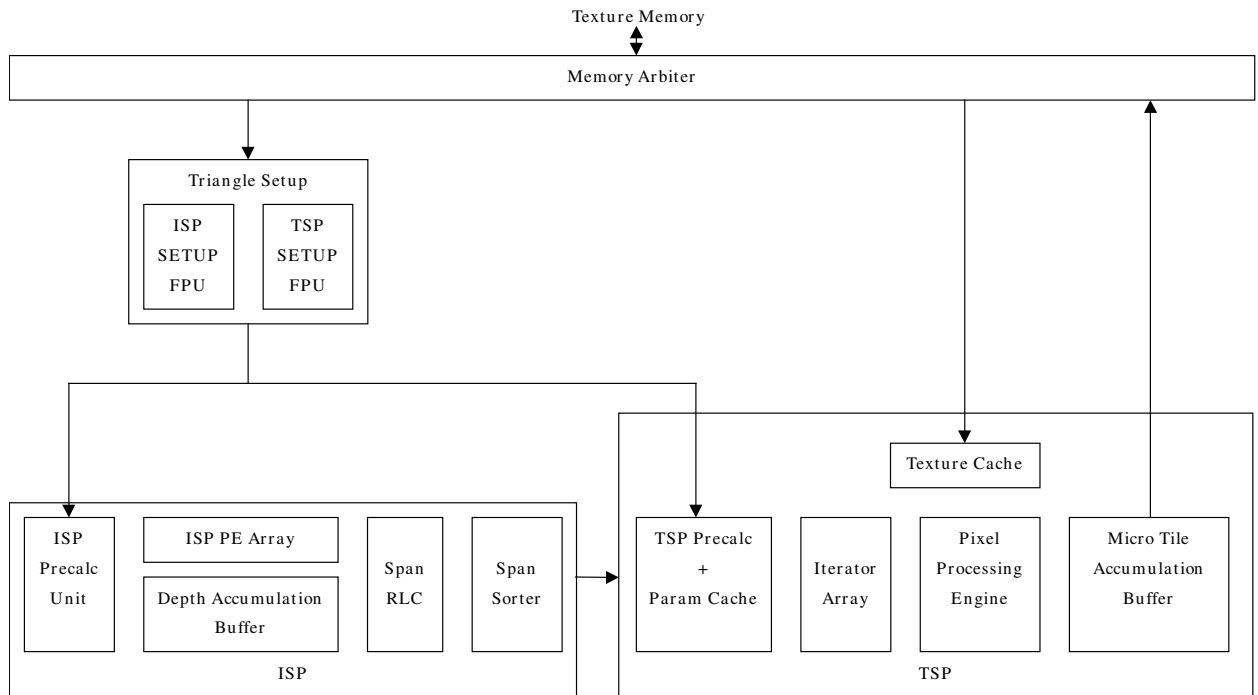


Fig. 3-5 Block Diagram

### § 3.1.1.6 Triangle Setup

The Triangle Setup block consists of the ISP SETUP FPU and the TSP SETUP FPU. this block calculates the polygon surface equations and the texture and shading parameters.

The ISP SETUP FPU calculates the parameters A, B, and C for the surface equation  $Ax + By + C$  from the coordinates of three vertices based on the following adjoint matrix.

$$(A,B,C) \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} = (z_0, z_1, z_2)$$

Solving this adjoint matrix yields the values of A, B, and C needed in order to describe the plane that passes through the three vertices that were provided. The result is:

$$(A, B, C) = (z_0, z_1, z_2) \frac{1}{\Delta} Adj$$

which yields:

$$Adj = \begin{pmatrix} y_1 - y_2 & x_2 - x_1 & x_1 y_2 - x_2 y_1 \\ y_2 - y_0 & x_0 - x_2 & x_2 y_0 - x_0 y_2 \\ y_0 - y_1 & x_1 - x_0 & x_0 y_1 - x_1 y_0 \end{pmatrix}$$

$$\Delta = x_0(y_1 - y_2) + x_1(y_2 - y_0) + x_2(y_0 - y_1)$$

The resulting  $\Delta$  value can be used to perform culling processing for very small polygons.

Note: The x, y, and z values shown in the above equations are all screen coordinates, and are equivalent to (fX, fY, fInvW) shown in section 3.1.1.2.

The ISP SETUP FPU requires 14 clock cycles to calculate the parameters.

The TSP SETUP FPU calculates the surface equations  $Px + Qy + R$  for shading and texture, respectively. The number of parameters that are actually calculated depends on whether the calculations are being made in texture mode or shading mode.

In addition, the parameters that are produced by the TSP SETUP FPU are stored in a cache in the TSP block; the TSP SETUP FPU calculates the parameters only when a miss is generated in the cache.

The TSP SETUP FPU requires 48 to 70 clock cycles to calculate the parameters.

### § 3.1.1.7 ISP(Image Synthesis Processor)

The ISP performs on-chip depth sorting for triangles without requiring an external Z buffer. The ISP works on  $32 \times 32$  Tiles, and performs its processing in a number of clock cycles equivalent to the number of lines in one triangle. All 1024 screen pixels located in a Tile are processed in parallel.

The processed pixels are sent to the Span RLC, which executes Run Length Encoding on 32 pixels in parallel in each clock cycle and sends the result to the Span Sorter. This approach maximizes the data transfer speed between the ISP and the TSP, and lessens the demand for buffering between these two modules.

An overview of the Span RLC is shown below.

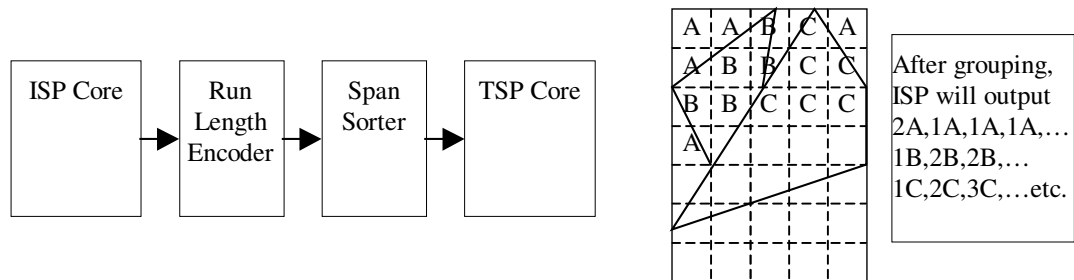


Fig. 3-6 Overview of the Span RLC

The Span Sorter regroups the run length encoded spans from the ISP in the triangle sequence. Therefore, data for the triangles is supplied to the TSP at one time.

Span sorting offers the following benefits:

- Minimizes caching requirements for the TSP parameters.
- Provides the benefits of the Tile-based method (in terms of speed, cost, minimum bandwidth, and no Z buffer).
- Provides additional benefits beyond conventional methods. (Consistency with the Z buffer texture map)

### § 3.1.1.8 TSP(Texture and Shading Processor)

The TSP performs texture and shading processing, and draws in the Tile accumulation buffer. Once all Tiles have been drawn, the TSP writes the contents of the accumulation buffer to texture memory. The TSP has a local cache for parameters that have been calculated, which is used to minimize the recalculation of parameters by taking advantage of consistencies between visible polygons.

There is a  $64 \times 64$ -bit texture cache for normal texels or the VQ texture code book, and a  $64 \times 64$ -bit texture cache for VQ texture indices, for a total of  $128 \times 64$  bits.

The TSP performs perspective compensation for all texture and shading elements, U, V, Alpha, R, G, B, and Fog.



### § 3.1.1.9 Polygon List

HOLLY utilizes the following **five** lists:

- (1) Opaque: Opaque polygon list
- (2) **Punch Through:** **Punch Through polygon list**
- (3) Opaque Modifier Volume: Opaque polygon and Punch Through Polygon Modifier
- (4) Translucent: Translucent polygon list
- (5) Translucent Modifier Volume: Translucent Polygon Modifier Volume list

The Opaque list is for a non-textured polygon with no alpha blending, or for a textured polygon with no alpha blending in which all of the texels are opaque (with an alpha value of 1.0 only). **The Punch Through is for a textured polygon with no alpha blending in which all of the texels are transparent or opaque (with an alpha value of 0.0 or 1.0 only).** The Translucent List is for textured and non-textured polygons with alpha blending, or for a textured polygon with no alpha blending in which the texels are translucent (with an alpha value ranging from 0.0 to 1.0). In addition, Modifier Volume lists are for polygons that are used to distinguish different areas in order to give an object a three-dimensional feel through shadows, etc. There are two types of Modifier Volumes, one for **Opaque and Punch Through polygons** and one for Translucent polygons. (Refer to Section 3.4.3.)

These lists are drawn in order, starting from (1), for each Tile. When drawing Opaque polygons, the ISP processes the number of Opaque polygons that exist in the Tile in question, and then the TSP performs texturing and shading processing on those pixels that are visible. **When drawing Punch Through polygons, the ISP sorts the polygons that exist in the Tile in question, starting from the front, and then the TSP performs texturing and shading processing on those pixels that are visible. This processing by the ISP and the TSP continues until all of the pixels in the Tile have been drawn.** Furthermore, when drawing translucent polygons, the ISP draws the product of the number of translucent polygons that exist in the Tile in question multiplied by the number of overlapping polygons (when in Auto Sort mode), and then the TSP performs texturing and shading processing on all pixels in the translucent polygons. Therefore, it is important to be aware that drawing translucent polygons can require much more processing time than drawing Opaque polygons.

It is also necessary to note that this also applies to Opaque Modifier Volumes and Translucent Modifier Volumes.

### § 3.1.2 Drawing Function Overview

HOLLY has many drawing functions; some typical functions are listed below.

- On-chip deletion of hidden surfaces with 32-bit precision (Z buffer not needed)
- Reduction of memory size for display image data (strip buffer mode)
- Support for infinite strip Triangle polygons
- Generation of display lists for drawing individual Tiles through hardware (Tile Accelerator)
- Punch Through polygon drawing processing
- Texture rings with perspective compensation
- True color Gouraud shading with perspective compensation
- Translucent display with perspective compensation
- Support for full D3D source and destination blending
- Translucent polygon auto sort through hardware
- Shadow and satellite generation (Modifier Volume)
- Texture and Shading Color switching in special areas (Modifier Volume)
- Fog (indices, lines, vertices)
- Clipping of Tile units and pixel units
- Rendering to a texture map
- Dithering
- Full-screen scaling and filtering
- Flicker-free interlacing
- Support for bi-linear and tri-linear filtering
- 4x texture super sampling
- Texture sizes ranging from  $8 \times 8$  to  $1024 \times 1024$
- Mip-map textures
- Support for rectangular textures
- Texture UV flipping and clamping
- Approximately 1/8 texture compression using vector quantization (VQ textures)
- Support for 4BPP and 8BPP palette textures (1024-color palette RAM on chip)
- Support for YUV422 textures (includes YUV420  $\rightarrow$  YUV422 data converter)
- Support for Bump Mapping

### § 3.1.3 Display Function Overview

The video display modes that are supported by this system are listed below.

Display mode	Resolution (pixels)	Interlace mode
NTSC_320 × 240NI	320 × 240	Non-interlaced
NTSC_320 × 240I	320 × 240	Single interlaced
NTSC_640 × 240NI	640 × 240	Non-interlaced
NTSC_640 × 240I	640 × 240	Single interlaced
NTSC_640 × 480	640 × 480	Double interlaced
PAL_320 × 240NI	320 × 240	Non-interlaced
PAL_320 × 240I	320 × 240	Single interlaced
PAL_640 × 240NI	640 × 240	Non-interlaced
PAL_640 × 240I	640 × 240	Single interlaced
PAL_640 × 480	640 × 480	Double interlaced
VGA	640 × 480	Non-interlaced

Note:

Single interlaced: The same image is displayed in odd and even fields (480 display lines).

Double interlaced: Separate images are displayed in odd and even fields.

Table 3-1 Display Mode List

## § 3.2 Memory Map

### § 3.3 Register Map

The HOLLY register map is listed below.

Address	Name	R/W	Description
0x005F 8000	ID	R	Device ID
0x005F 8004	REVISION	R	Revision number
0x005F 8008	SOFTRESET	RW	CORE & TA software reset
0x005F 8014	STARTRENDER	RW	Drawing start
0x005F 8018	TEST_SELECT	RW	Test (writing this register is prohibited)
0x005F 8020	PARAM_BASE	RW	Base address for ISP parameters
0x005F 802C	REGION_BASE	RW	Base address for Region Array
0x005F 8030	SPAN_SORT_CFG	RW	Span Sorter control
0x005F 8040	VO_BORDER_COL	RW	Border area color
0x005F 8044	FB_R_CTRL	RW	Frame buffer read control
0x005F 8048	FB_W_CTRL	RW	Frame buffer write control
0x005F 804C	FB_W_LINESTRIDE	RW	Frame buffer line stride
0x005F 8050	FB_R_SOF1	RW	Read start address for field - 1/strip - 1
0x005F 8054	FB_R_SOF2	RW	Read start address for field - 2/strip - 2
0x005F 805C	FB_R_SIZE	RW	Frame buffer XY size
0x005F 8060	FB_W_SOF1	RW	Write start address for field - 1/strip - 1
0x005F 8064	FB_W_SOF2	RW	Write start address for field - 2/strip - 2
0x005F 8068	FB_X_CLIP	RW	Pixel clip X coordinate
0x005F 806C	FB_Y_CLIP	RW	Pixel clip Y coordinate
0x005F 8074	FPU_SHAD_SCALE	RW	Intensity Volume mode
0x005F 8078	FPU_CULL_VAL	RW	Comparison value for culling
0x005F 807C	FPU_PARAM_CFG	RW	Parameter read control
0x005F 8080	HALF_OFFSET	RW	Pixel sampling control
0x005F 8084	FPU_PERP_VAL	RW	Comparison value for perpendicular polygons
0x005F 8088	ISP_BACKGND_D	RW	Background surface depth
0x005F 808C	ISP_BACKGND_T	RW	Background surface tag
0x005F 8098	ISP_FEED_CFG	RW	Translucent polygon sort mode
0x005F 80A0	SDRAM_REFRESH	RW	Texture memory refresh counter
0x005F 80A4	SDRAM_ARB_CFG	RW	Texture memory arbiter control
0x005F 80A8	SDRAM_CFG	RW	Texture memory control
0x005F 80B0	FOG_COL_RAM	RW	Color for Look Up table Fog
0x005F 80B4	FOG_COL_VERT	RW	Color for vertex Fog
0x005F 80B8	FOG_DENSITY	RW	Fog scale value
0x005F 80BC	FOG_CLAMP_MAX	RW	Color clamping maximum value
0x005F 80C0	FOG_CLAMP_MIN	RW	Color clamping minimum value

Note: RW: read/write; R: read only; W: write only

Address	Name	R/W	Description
0x005F 80C4	SPG_TRIGGER_POS	RW	External trigger signal HV counter value
0x005F 80C8	SPG_HBLANK_INT	RW	H-blank interrupt control

0x005F 80CC	SPG_VBLANK_INT	RW	V-blank interrupt control
0x005F 80D0	SPG_CONTROL	RW	Sync pulse generator control
0x005F 80D4	SPG_HBLANK	RW	H-blank control
0x005F 80D8	SPG_LOAD	RW	HV counter load value
0x005F 80DC	SPG_VBLANK	RW	V-blank control
0x005F 80E0	SPG_WIDTH	RW	Sync width control
0x005F 80E4	TEXT_CONTROL	RW	Texturing control
0x005F 80E8	VO_CONTROL	RW	Video output control
0x005F 80Ec	VO_STARTX	RW	Video output start X position
0x005F 80Fo	VO_STARTY	RW	Video output start Y position
0x005F 80F4	SCALER_CTL	RW	X & Y scaler control
0x005F 8108	PAL_RAM_CTRL	RW	Palette RAM control
0x005F 810C	SPG_STATUS	R	Sync pulse generator status
0x005F 8110	FB_BURSTCTRL	RW	Frame buffer burst control
0x005F 8114	FB_C_SOF	R	Current frame buffer start address
0x005F 8118	Y_COEFF	RW	Y scaling coefficient
0x005F 811C	PT_ALPHA_REF	RW	Alpha value for Punch Through polygon comparison
0x005F 8124	TA_OL_BASE	RW	Object list write start address
0x005F 8128	TA_ISP_BASE	RW	ISP/TSP Parameter write start address
0x005F 812C	TA_OL_LIMIT	RW	Start address of next Object Pointer Block
0x005F 8130	TA_ISP_LIMIT	RW	Current ISP/TSP Parameter write address
0x005F 8134	TA_NEXT_OPB	R	Global Tile clip control
0x005F 8138	TA_ITP_CURRENT	R	Current ISP/TSP Parameter write address
0x005F 813C	TA_GLOB_TILE_CLIP	RW	Global Tile clip control
0x005F 8140	TA_ALLOC_CTRL	RW	Object list control
0x005F 8144	TA_LIST_INIT	RW	TA initialization
0x005F 8148	TA_YUV_TEX_BASE	RW	YUV422 texture write start address
0x005F 814C	TA_YUV_TEX_CTRL	RW	YUV converter control
0x005F 8150	TA_YUV_TEX_CNT	R	YUV converter macro block counter value
0x005F 8160	TA_LIST_CONT	RW	TA continuation processing
0x005F 8164	TA_NEXT_OPB_INIT	RW	Additional OPB starting address
0x005F 8200- 0x005F 83FC	FOG_TABLE	RW	Look-up table Fog data
0x005F 8600- 0x005F 8F5C	TA_OL_POINTERS	R	TA object List Pointer data
0x005F 9000- 0x005F 9FFC	PALETTE_RAM	RW	Palette RAM

Note: RW: read/write; R: read only; W: write only

Table 3-2 Register Map

## § 3.4 Drawing Function Details

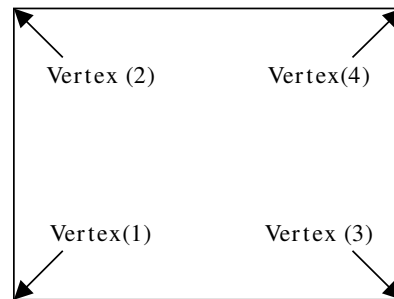
### § 3.4.1 Background

In areas where nothing is drawn by the CORE display list, the background is drawn according to separately specified ISP/TSP Parameters. The background ISP/TSP Parameters are normally stored directly in texture memory without passing through the TA, and the address is specified in the ISP\_BACKGND\_T register. In addition, the depth value is specified in the ISP\_BACKGND\_D register.

*ISP/TSP Parameter*

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex (1) :Vertex X
Vertex (1) :Vertex Y
Vertex (1) :Vertex Z
Vertex (1) :Texture U
Vertex (1) :Texture V
Vertex (1) :Base Color
Vertex (1) :Offset Color
Vertex (2) :Vertex X
Vertex (2) :Vertex Y
Vertex (2) :Vertex Z
Vertex (2) :Texture U
Vertex (2) :Texture V
Vertex (2) :Base Color
Vertex (2) :Offset Color
Vertex (3) :Vertex X
Vertex (3) :Vertex Y
Vertex (3) :Vertex Z
Vertex (3) :Texture U
Vertex (3) :Texture V
Vertex (3) :Base Color
Vertex (3) :Offset Color

*Positions of vertices on the screen*



**<Notes>**

- The Texture Control Word is required even when textures are not being used. However, because it is not actually used, the value does not matter.
- The parameters for vertex (4) are not needed, because they are calculated automatically on the basis of the parameters for the other three vertices.
- The vertex Z values are not the same as the Z value for the background.
- If textures are not used, the texture U and V data is not needed.
- If offset colors are not used, the offset color data is not needed.

Fig. 3-7

Normally, the CORE display list is stored in two parts, one for writing texture memory from the TA, and one for reading texture memory from the CORE. (double buffer processing) Similarly, the background ISP/TSP Parameters also are stored beforehand in two buffer areas in texture memory, with the most efficient approach being to specify through the ISP\_BACKGND\_T register the background ISP/TSP Parameters that are stored in the CORE display list that is used for drawing.

### § 3.4.2 Translucent Polygon Sort

There are two polygon sort modes for drawing translucent polygons: "Auto-sort mode" and "Pre-sort mode." ~~The sort mode specification method differs according to the HOLLY version. In HOLLY1, either sort mode can be specified for individual screens according to the setting in the ISP\_FEED\_CFG register.~~ In Sort mode, the specification differs according to the Region Array data type. For Region Array data type 1 (when bit 21 in the FPU\_PARAM\_CFG register is "0"), the "pre-sort mode" is specified for individual screens in the ISP\_FEED\_CFG register. For Region Array data type 2 (when bit 21 in the FPU\_PARAM\_CFG register is "1"), "pre sort" is specified for individual Tiles in the Region Array data.

#### § 3.4.2.1 Auto-sort Mode

In auto-sort mode, the hardware automatically sorts polygons as individual pixels, and draws the pixels starting from the farthest Z value, regardless of the order in which the polygons were input to the TA (registered in the display list). Therefore,  $\alpha$  blending is performed properly even in a case where two translucent polygons intersect. However, because the polygons are sorted as individual pixels, sort processing must be performed for [the number of registered polygons]  $\times$  [the number of overlapping pixels], with the result that a large amount of processing time is required when a large number of translucent polygons overlap.

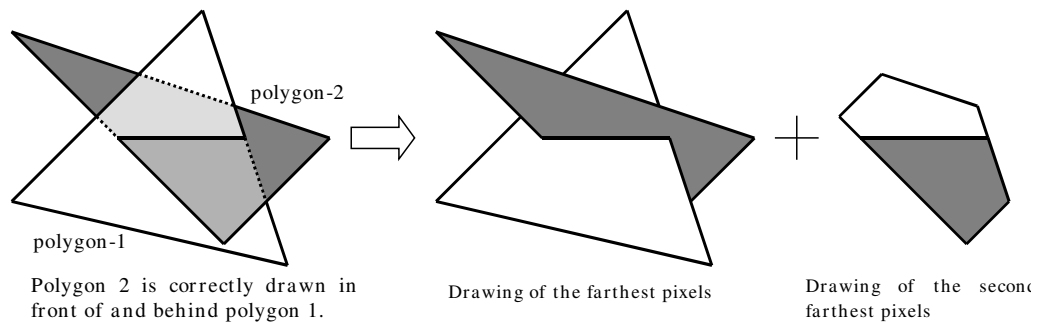


Fig. 3-8

Sprites (textured polygons that use transparent texels) must be drawn with translucent polygons, even if no  $\alpha$  blending is performed. Auto-sort mode is not recommended for use with Sprites in 2D software that uses a lot of Sprites because 3D sorting is not required, and because polygons may overlap much more frequently than might be initially expected.

Furthermore, in auto-sort mode "Depth Compare Mode," specified in the ISP/TSP Instruction Word, is disabled; Z values are always compared on the basis of "greater or equal." When two pixels have the same Z value, the polygon that was input to the TA first is drawn the farthest away.

### § 3.4.2.2 Pre-sort Mode

In pre-sort mode, polygons are drawn in the order in which they were input to the TA, as with a normal Z buffer system.

Because processing is only performed for the number of polygons registered, this mode requires less processing time than auto sort mode. However, because it is essential to sort the polygons before inputting the polygon data to the TA, this mode does increase the CPU's workload. Furthermore, alpha blending is not performed correctly when two polygons intersect.

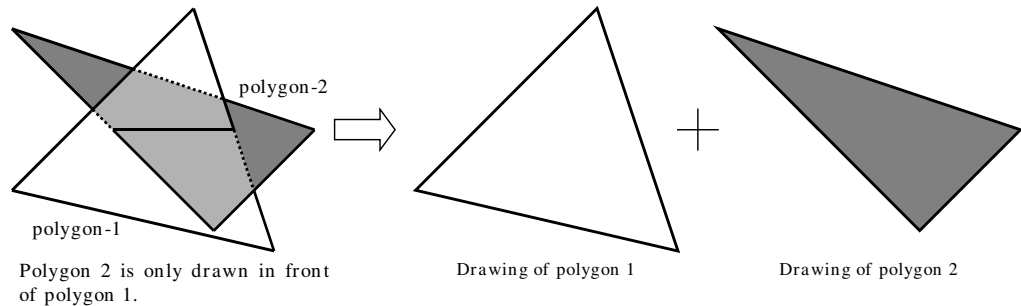


Fig. 3-9

In addition, Translucent Modifier Volumes cannot be used in this mode.

### § 3.4.3 Punch Through Polygons

In drawing Punch Through polygons ~~with the Hollye~~, the hardware automatically sorts the polygon at the pixel level, and draws the pixels in order according to their Z value, starting from the front, regardless of the order in which they were input to the TA (registered in the display list). When drawing, the hardware reads the texture data and draws only the pixels for which (texel alpha value)  $\geq$  (PT\_ALPHA\_REF register value), and processing continues until all pixels within the Tile have been drawn. Normally, "0xFF (=1.0)" should be specified for the PT\_ALPHA\_REF register value. Pixels are drawn with an alpha value of 1.0. (Translucent processing is not performed.)

Depth Compare Mode specified in the ISP/TSP Instruction Word is invalid, and Z values are always compared on a "Greater or Equal" basis. When the Z values of two pixels are identical, the one belonging to the polygon that was input to the TA first is drawn behind the other.

#### § 3.4.3.1 ISP Cache Size

Drawing processing in the Punch Through polygon ISP is performed in units of polygon groups with a number of vertices (ISP cache size) specified by "Punch Through chunk size" in the ISP\_FEED\_CFG register.

- (1) The Punch Through polygon data for the number of vertices specified in the register is stored in the ISP cache.
- (2) While automatically sorting the polygons in the ISP cache, the hardware begins drawing the pixels, starting from the front.
- (3) The processing in step 2 is repeated until all polygons in the ISP cache have been processed.
- (4) If there are more polygons registered in the Tile than the number of vertices specified, steps 1 through 3 are repeated until the registered polygons are all processed.

Normally, 0x040 to 0x080 (0x040 is recommended) is specified for the ISP cache size for Punch Through polygon processing. However, when many of a polygon's transparent texels (alpha value = 0.0) are overlapping, specifying a large ISP cache size may result in a worsened drawing processing efficiency; if this happens, adjust the ISP cache size to a more suitable level.

However, the ISP cache size for Punch Through polygons must be no larger than the ISP cache size for Translucent polygons. ([Punch Through chunk size]  $\leq$  [Cache size for translucency])



### § 3.4.3.2 Relationship with Translucent Polygons

Punch Through polygons are drawn in the same manner if they are registered as Translucent polygons, but normally drawing a polygon as a Punch Through polygon requires much less time than drawing the same polygon as a Translucent polygon. However, if bi-linear filtering is performed in a Punch Through polygon, some opaque texels might not be drawn, depending on the texel sampling position. This is because, in Punch Through polygons, only those pixels with an alpha value (after texture filtering) that is equal to or greater than the value in the PT\_ALPHA\_REF register (normally 10) are drawn. (Refer to section 3.4.7.2.2.)

When a Translucent polygon that is completely identical to a Punch Through polygon has been registered, those pixels with an alpha value of ten are drawn only through the Punch Through polygon; when the Translucent polygon is drawn, those pixels are judged to have already been drawn and are not drawn again. This feature can be used to improve the problem of the disappearance of opaque pixels when using bilinear filtering with Punch Through polygons, without extending the translucent polygon processing time very much.

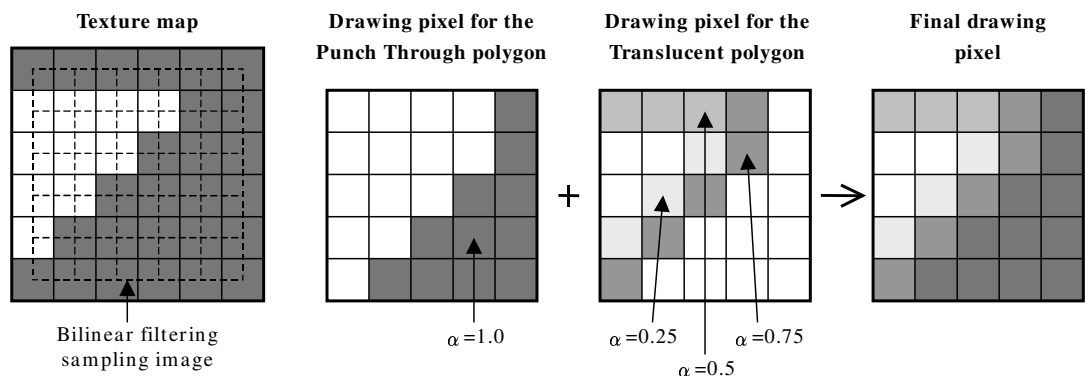


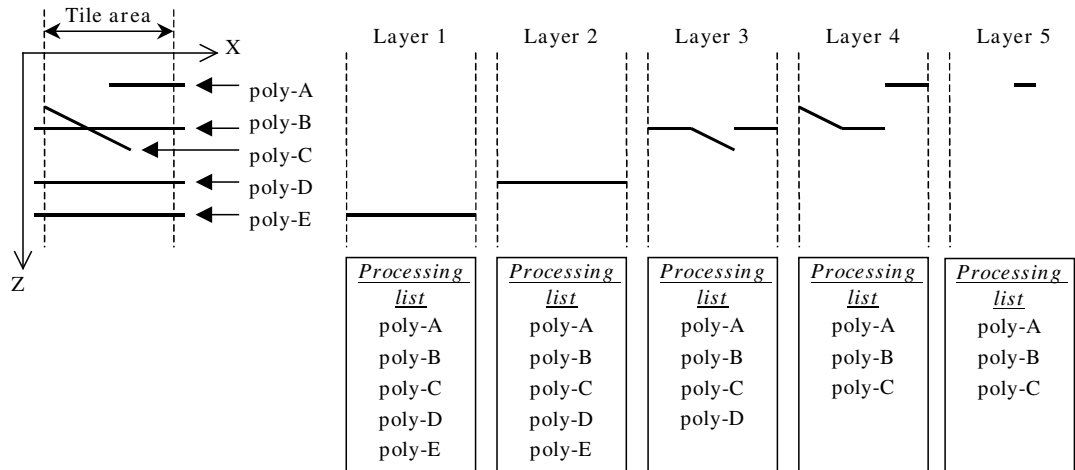
Fig. 3-10

### § 3.4.4 Processing List Discarding

Because the ~~HOLLY~~ hardware automatically draws Punch Through polygons and Translucent polygons (in Auto sort mode) as individual pixels while sorting the polygons at the same time, it is not necessary for the CPU to sort the polygons before inputting them to the TA. However, due to the sort processing, the hardware has to process each registered polygon a number of times. In effect, the hardware is drawing a number of polygons equal to [number of registered polygons] x [number of overlaps].

In order to reduce the amount of such processing in Auto Sort mode, the ~~HOLLY~~ hardware is capable of performing processing called "discarding," in which polygons that have been completely drawn are removed ("discarded") from the processing list. This processing is specified through "Discard Mode" in the ISP\_FEED\_CFG register.

**Example of discarding when processing Translucent polygons**



discarding

In Layer 2 processing, poly-E is deemed to have no drawing pixels remaining, so it is removed from the processing list in Layer 3 processing. In the same fashion, poly-D is removed from the processing list in Layer 4 processing. Because poly-C still has drawing pixels in Layer 4 processing, it is not removed in Layer 5 processing.

Fig. 3-11

In Punch Through polygon drawing processing, the Z value results of previously drawn Opaque polygons are referenced. Furthermore, in Translucent polygon drawing processing, the Z value results of Opaque polygons and Punch Through polygons are referenced. For example, a Punch Through polygon and a Translucent polygon that are fully hidden behind an Opaque polygon are both discarded in layer 1 processing, so they are only processed once.

Therefore, when creating model data in which many Punch Through polygons and Translucent polygons overlap, the drawing time can be reduced by inserting Opaque polygons between them.

### § 3.4.5 Modifier Volume

A Modifier Volume is polygon data that is used to define an area for adding shadows and otherwise generate a 3D fell for normal polygons; the Modifier Volume is not actually drawn on the screen. There are two areas on the screen as a whole that are defined ("area 0" and "area 1"), and the texture and Shading Color for each area can be changed through the Modifier Volume. This function can therefore be used to create a variety of effects, such as shadows, spotlights, or window masking.

**Example: Opaque polygon and opaque modifier volume**

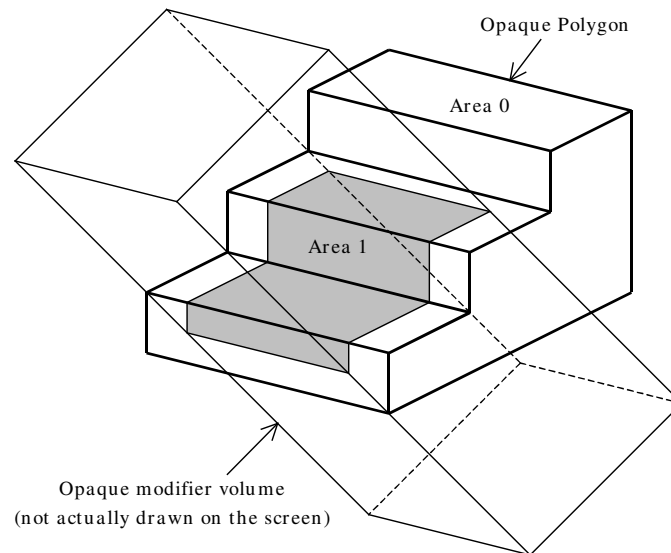


Fig. 3-12

There are two types of Modifier Volumes: "opaque Modifier Volumes," which are effective only for **Opaque polygons and Punch Through polygons**; and "translucent Modifier Volumes," which are effective only for translucent polygons. Although there is no limit on the number of either type of Volume models that may be registered in lists, the maximum number of areas that can be defined is two.

**Example: Opaque Modifier Volumes and Translucent Modifier Volumes**

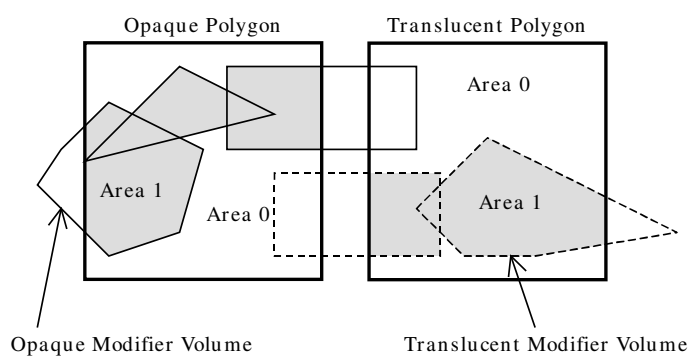


Fig. 3-13

Volume models can be either a protruding shape or a recessed shape, as long as it is a closed shape; otherwise, the three-dimensional area definition will not be performed correctly. However, if planar area definition is sufficient, the Volume model does not need to be a three-dimensional object and does not need to be closed. In this case, the area of the Volume polygon that is deemed to be in front of the normal polygon is designated as "area 1." In addition, it is necessary to make a distinction between and specify the final polygon that forms a Volume model as opposed to the other polygons. (Refer to section 3.7.4.4.3.) If this specification is not made, area definition will not be performed properly.

#### § 3.4.5.1 Inclusion and Exclusion Volumes

There are two types of Modifier Volumes: inclusion volumes and exclusion volumes. An inclusion volume makes the polygon surface that is inside the volume "area 1," while an exclusion volume makes the polygon surface that is inside the volume "area 0." Prior to area determination by volume, all polygon surfaces are "area 0." Therefore, an exclusion volume is used to make "area 0" inside of an "area 1" that was created by an inclusion volume.

CORE has one flag bit per pixel in order to maintain the area status of each individual pixel. When processing multiple volumes, area definition by a volume is performed for one model at a time, and the final area is determined by performing Boolean operations on each result versus the cumulative result of the Boolean operations performed for that pixel up to that point. The flag value defined by a volume is a "1" if that pixel is inside the volume, and a "0" if that pixel is outside the volume. If the final flag value is a "0," that pixel is in area 0; if the final flag value is a "1," that pixel is in area 1. Note that the initial flag value is "0."

The Boolean operations that are performed on the flag bits for inclusion volumes and exclusion volumes are as follows:

For an inclusion volume:

$$(\text{New flag value}) = (\text{current flag value}) | (\text{result indicated by the volume})$$

For an exclusion volume:

$$(\text{New flag value}) = (\text{current flag value}) \& (\text{result indicated by the volume})$$

The inclusion volume and exclusion volume specifications are made in the volume instruction in the ISP/TSP Instruction Word. (Refer to section 3.7.9.1.)

#### § 3.4.5.2 Volume Modes

There are two modes for processing that is performed on the area defined as area 1: "parameter selection volume mode" and "intensity volume mode." Processing performed on area 0 is the same as processing that is performed on a normal object. These modes are specified through the FPU\_SHAD\_SCALE register, and can be selected for an entire screen only.

In addition, it is possible to specify for each object whether processing is to be performed on area 1 or not. This specification is made in the Parameter Control Word for the display list that is input to the TA. (Refer to section 3.7.4.4.3.)

### Parameter Selection Volume Mode

In this mode, there are two sets of ISP/TSP Parameters for one object, and the parameters that are used switch for each area that is defined. When using this mode, it is necessary to input to the TA the object data which enables the Modifier Volume with the specification "with Two Volume format." Parameter 0 that is input to the TA is used for area 0, and parameter 1 is used for area 1.

In this mode, everything that can be specified in the ISP/TSP Parameters, including textures and UV coordinates, can be changed between the two areas. This mode makes possible effects such as "spotlight (changing the Shading Color)," which brightens the area, or "window masking (changing the texture map)," which makes only the area translucent. However, one shortcoming of this mode is that the amount of data in the display list is large, because two sets of ISP/TSP Parameters must be stored.

### Intensity Volume Mode

This mode is used to represent simple shadows with only one set of ISP/TSP Parameters. The parameters that are used for both areas are basically the same, but for area 1 the Base Color and Offset Color are multiplied by the 8-bit value that is specified in the FPU\_SHAD\_SCALE register.

This mode cannot be used correctly with Bump mapped polygons because the K1K2K3Q data changes.

Because the 8-bit data that is multiplied with the Shading Color data is set in a register, it is only possible to represent shadows with just one level of darkness on the screen, but if only simple shadows are needed, this mode permits them to be represented without increasing the amount of data in the display list.

## § 3.4.5.3 Modifier Volume Processing for Various Polygons

An Opaque Modifier Volume list is used for Modifier Volumes for Opaque polygons.

~~In HOLLY2,~~ Modifier Volume processing on Opaque polygons is performed together with Modifier Volume processing on Punch Through polygons.

An Opaque Modifier Volume list is used for Modifier Volumes for Punch Through polygons. The Punch Through polygon processing is first performed for "area 0," and then only those pixels that form "area 1" due to the Modifier Volume are drawn again. Therefore, if the texture data and UV coordinate values that are used for parameter 0 and parameter 1 in Parameter Selection Volume mode differ, inconsistencies such as a pixel that was opaque in area 0 being transparent in area 1 can arise, resulting in not being able to draw the polygons correctly. Also, graphics cannot be correctly drawn even if parameters 0 and 1 have different Base Color alpha values.

Therefore, for Punch Through polygons, it is normally possible to change only the base color and offset color RGB values in parameter 0 and parameter 1.

The processing time for an Opaque Modifier Volume is simply equivalent to the drawing time for that number of polygons.

A Translucent Modifier Volume list is used for Modifier Volumes for Translucent polygons. Because only Auto-sort mode is supported, Modifier Volume processing is performed for each layer while sorting the polygons, starting from the back. Therefore, because each Translucent polygon is processed once for each layer that they overlap, a great deal more processing time is required in comparison with an Opaque Modifier Volume.

### § 3.4.6 Flow of Texture Mapping and Shading Processing

The following diagram illustrates the flow of texture mapping and shading. Color clamp processing is performed Fog processing, and  $\alpha$  blend processing is performed after Fog processing. The pixel data that is drawn in units of Tiles is ultimately stored in the primary accumulation buffer, and from there it is transferred to the frame buffer in texture memory.

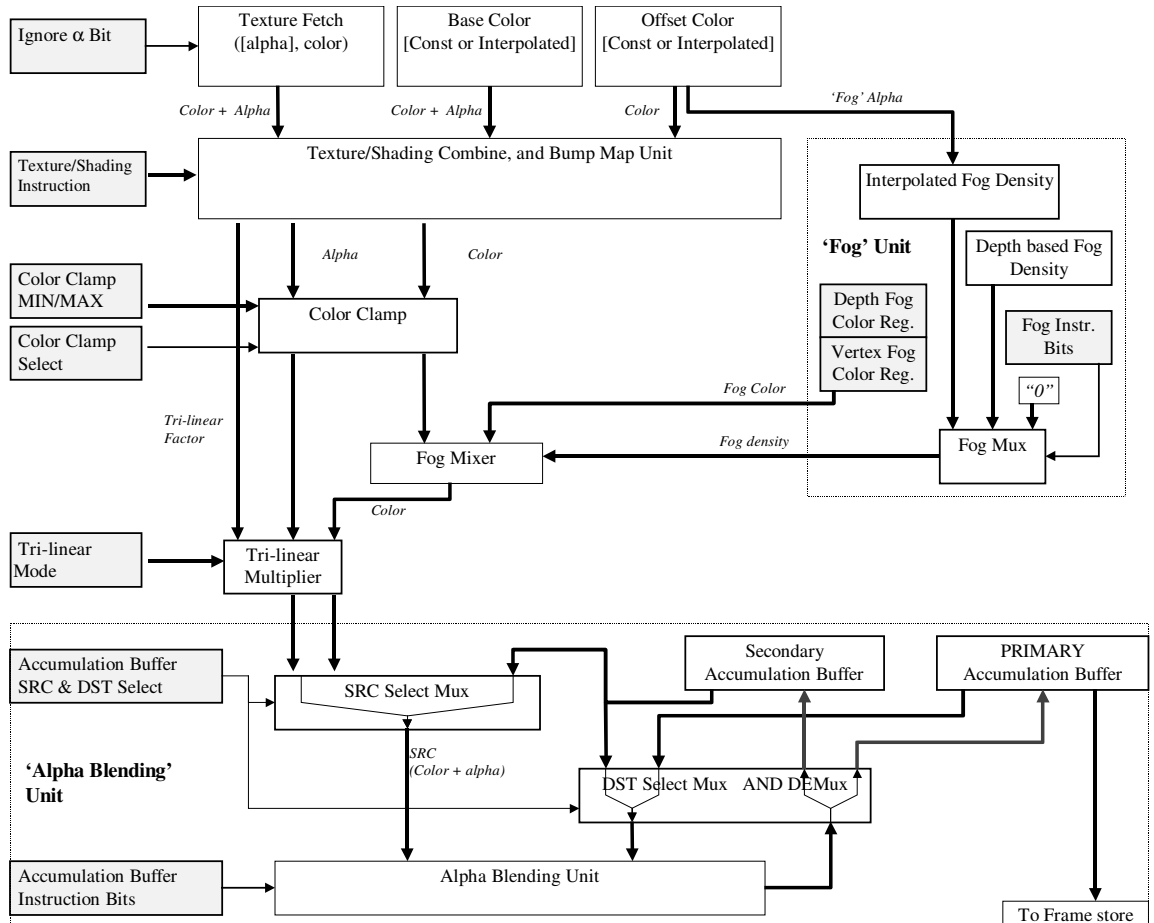


Fig.3-14

### § 3.4.6.1 Secondary Accumulation Buffer

Normally, drawing pixel data for individual Tiles is drawn in a buffer called the "Primary Accumulation Buffer." Another buffer, called the "Secondary Accumulation Buffer," is provided in order to permit the treatment of the result of overlapping multiple polygons as a single polygon.

The Secondary Accumulation Buffer is typically used for the following:

- Translucent polygons that have been subjected to trilinear filtering
- Translucent polygons that are Bump Mapped + Textured polygons

All that is necessary in order to be able to draw in the Secondary Accumulation Buffer is to set the DST Select bit (bit 24) in the TSP Instruction Word to "1". To use the results of the draw in the Secondary Accumulation Buffer as texture data, use a polygon that was drawn by drawing the data in the Secondary Accumulation Buffer to the Primary Accumulation Buffer (the Flush polygon), and set the SRC Select bit (bit 25) in the TSP Instruction Word to "1".

The pixel data that is stored when drawing to the Secondary Accumulation Buffer is ARGB 32-bit data of the same type that is normally drawn and stored in the Primary Accumulation Buffer. When drawing the result of alpha blending processing in the Secondary Accumulation Buffer to the Primary Accumulation Buffer as a Translucent polygon, it is essential to note that the pixel alpha values for the polygon in the Secondary Accumulation Buffer are used, so it is not possible to use the alpha value in the Base Color of the Flush polygon for control.

The Flush polygon is a polygon that is used to extract a shape specified by pixel data that was previously stored in the Secondary Accumulation Buffer and then draw that shape in the Primary Accumulation Buffer (Cut & Paste). It is necessary to once draw to the Secondary Accumulation Buffer for the pixel coordinates that are to be cut. The pixel data in the Secondary Accumulation Buffer is used as is, and texture mapping and shading processing (including Base Color, Offset Color, texture, and Texture/Shading Instructions) are ignored. Therefore, use normal Non-textured polygons for Flush polygons.

## § 3.4.7 Texture Mapping

### § 3.4.7.1 MIPMAP

When a polygon on which a texture has been mapped moves in the Z direction, the size of the polygon that is displayed changes. In this case, if the same texture is used, the appearance of the texture becomes distorted as it changes in conjunction with the movement of the polygon, and even flickering can occur in a texture that has been applied to a small polygon.

The solution for this type of case is to prepare textures of different sizes beforehand, and then perform processing that switches among these textures in accordance with the size of the polygon on which they are displayed. This processing is called "MIPMAP" processing. ("MIP" stands for the Latin phrase "Multim Im Parvo," or "many in a small space.") MIPMAP textures are prepared as square textures ranging in size from 1 x 1 to a specified size.

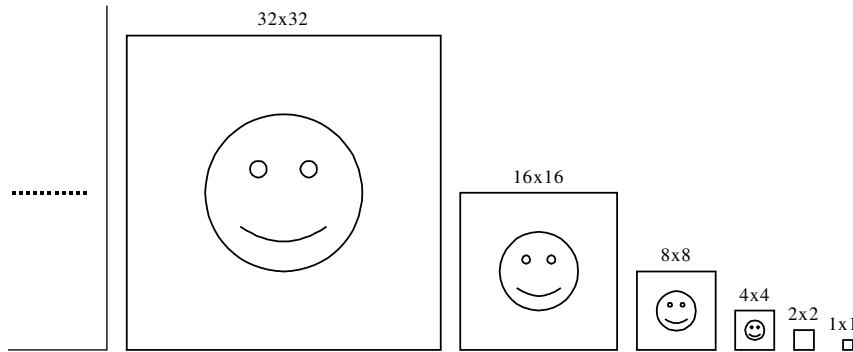


Fig. 3-15

The selection of a MIPMAP texture that accords with the displayed size of the polygon is performed according to a value, named "D," that is calculated by the CORE. For example, a texture of the specified size is used when  $0.0 < D < 2.0$ , and the next smaller sized texture is used when  $2.0 \leq D < 3.0$ .

The precision of the calculation of D (that is, the equation that is used) can be selected from among two types through "Dcalc Ctrl" in the ISP/TSP Instruction Word. Each of the equations is shown below. Note that "a," "b," "c," "d," "e," "f," "p," "q," and "r" are texture mapping coefficients, "X" and "Y" are screen coefficients, and "X'" and "Y'" are the screen coefficients of the first vertex.

$$D = \text{MAX}(dx, dy)$$

$$dx = \sqrt{dudx^2 + dvdx^2}, \quad dy = \sqrt{dudy^2 + dvdy^2}$$

When Dcalc Ctrl = 0

$$dudx = \frac{ar - pc}{(pX + qY + r)^2}$$

$$dvdx = \frac{dr - pf}{(pX + qY + r)^2}$$

$$dudy = \frac{br - qc}{(pX + qY + r)^2}$$

$$dvdy = \frac{er - qf}{(pX + qY + r)^2}$$

When Dcalc Ctrl = 1

$$dudx = \frac{a(pX' + qY' + r) - p(aX' + bY' + c)}{(pX + qY + r)^2}$$

$$dvdx = \frac{d(pX' + qY' + r) - p(dX' + eY' + f)}{(pX + qY + r)^2}$$

$$dudy = \frac{b(pX' + qY' + r) - q(aX' + bY' + c)}{(pX + qY + r)^2}$$

$$dvdy = \frac{e(pX' + qY' + r) - q(dX' + eY' + f)}{(pX + qY + r)^2}$$

The equations that are used when "Dcalc Ctrl = 1" offer greater precision for small polygons, but consume much more computing (drawing) time. Note also that the D value that is calculated by these equations can be adjusted through "MIP-MAP D adjust" in the TSP Instruction Word.

### § 3.4.7.2 Texture Filtering



There are three filtering modes (listed below) for texture mapping. The mode is specified through "Filter Mode" in the TSP Instruction Word.

- Point sampling
- Bi-linear filtering
- Tri-linear filtering

The polygon sampling position (x, y) that is used when calculating texture coordinates (u, v) can be selected through the HALF\_OFFSET register as either (0, 0) or (0.5, 0.5). Normally, (0.5, 0.5) is selected.

In addition, there is a function available, called "texture super-sampling," that enlarges the texture sampling point per pixel by a factor of four (by doubling the size in both the horizontal and vertical directions), thus increasing the image quality when the texture is compressed.

#### § 3.4.7.2.1 Point Sampling

Point sampling uses the data from the texture coordinates (u, v) that were derived from the sampling point (x, y) as texture data for the drawing pixel.

Although this mode entails the least processing load of the different types of texture mapping processing, the quality of the image deteriorates if it is enlarged or compressed.

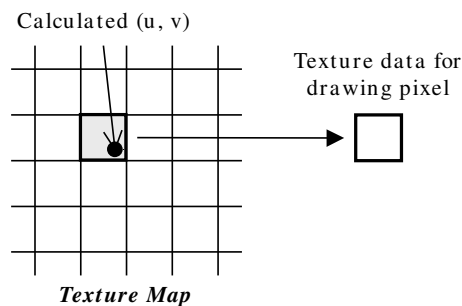


Fig. 3-16

### § 3.4.7.2.2 Bi-linear Filtering

Bi-linear filtering takes the weighted average of the data from the texture coordinates (u, v) that were derived from the sampling point (x, y) and the data from three adjacent texels (for a total of four texels), and uses the result as texture data for the drawing pixel.

Because the weighted average is taken from data for four texels, the quality of the image when expanded or compressed is superior to that produced by point sampling (although in some cases the image may appear to be out of focus). The processing time is practically the same as compared to point sampling when working with Twiddled-format textures, but when working with Non-Twiddled format textures, processing time can double in a worst-case instance.

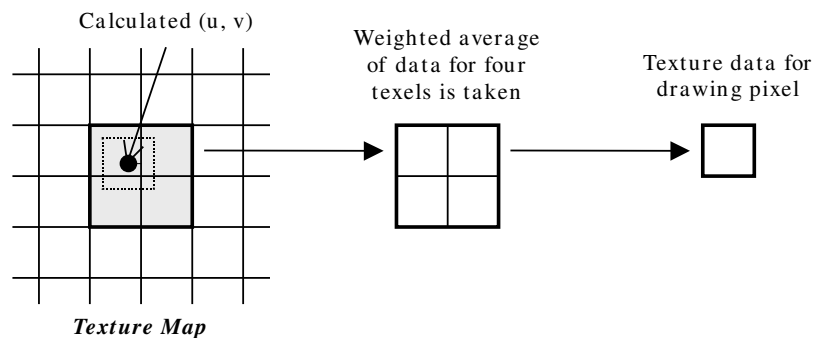


Fig. 3-17

Because data for four texels is used for bilinear filtering, in a case where the texture coordinates calculated from the sampling point lie on the edge of the texture map, then the adjacent texels (those that lie on opposite edges of the texture map) are used, which may result in an unexpected pixel color. (This problem can be avoided by using the Texture UV clamp function.) It is important to note that this problem also occurs when using an extracted portion of a larger texture map.

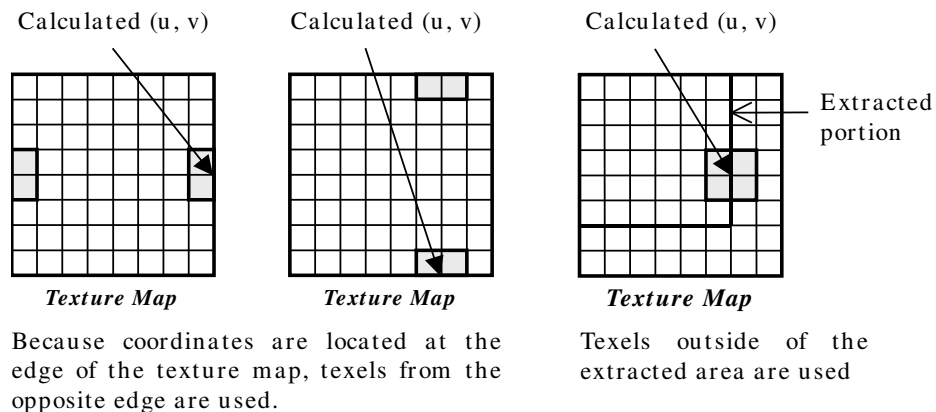


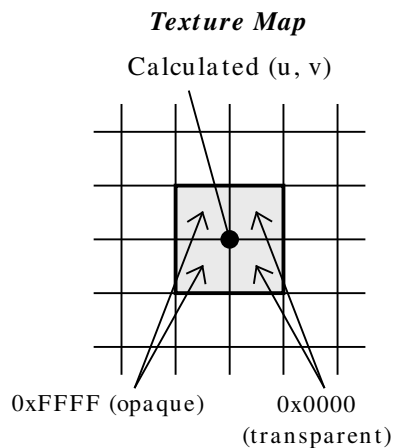
Fig. 3-18

When bilinear filtering has been performed on a texture that contains transparent texels, the transparent texel data (both alpha values and color values) is also used in calculating the weighted average of the four texels, with the result being used for the alpha value of the drawing pixel.

The translucent polygon has been drawn by alpha value of calculating result, but note that it may have an influence on transparent texel color data at boundaries between transparent and opaque pixels.

In the case of Punch Through polygons ~~in the HOLLYe~~, pixels are drawn only if their calculated alpha value is 1.0 (in other words, when the alpha value of all four texels is 1.0).

When drawing a polygon using a Punch through texture (a texture in which the alpha values are only 0.0 or 1.0), the drawing results at boundaries between transparent and opaque pixels differ, depending on whether the polygon is registered as a Punch Through polygon or a Translucent polygon. Although the boundary is neater in the case of a Translucent polygon, much more time is required to draw a Translucent polygon as opposed to a Punch Through polygon.



In the case of a Punch Through

In the drawing pixel data, the transparent texel data has an effect, so that the calculated alpha value = 0.5; therefore, the pixel is not drawn.

In the case of a Translucent

In the drawing pixel data, the transparent texel data has an effect, so that the as a result of the calculation so that each 8-bit color RGB = 0x7F and is drawn with an alpha value of 0.5.

Fig. 3-19

### § 3.4.7.2.3 Tri-linear Filtering

When using one texture map, it is possible to improve the quality of the image by using bi-linear filtering instead of point sampling. In addition, MIPMAP processing is used in order to improve the quality of the image when the compression factor is large (i.e., the image has moved away along the Z axis). However, even if bi-linear filtering is used in conjunction with MIPMAP processing, it is possible to clearly see the switchovers between MIPMAP textures of different sizes. Tri-linear filtering takes the weighted average of the results produced by bi-linear filtering of MIPMAP textures of two different sizes, and uses the result as texture data for the drawing pixel.

Because the weighted average is taken from data for eight texels in all, this approach offers the best quality in enlarged and compressed images, and the switchovers between MIPMAP textures appear smooth. However, because this method requires the most processing time, it is not recommended for use with all polygons.

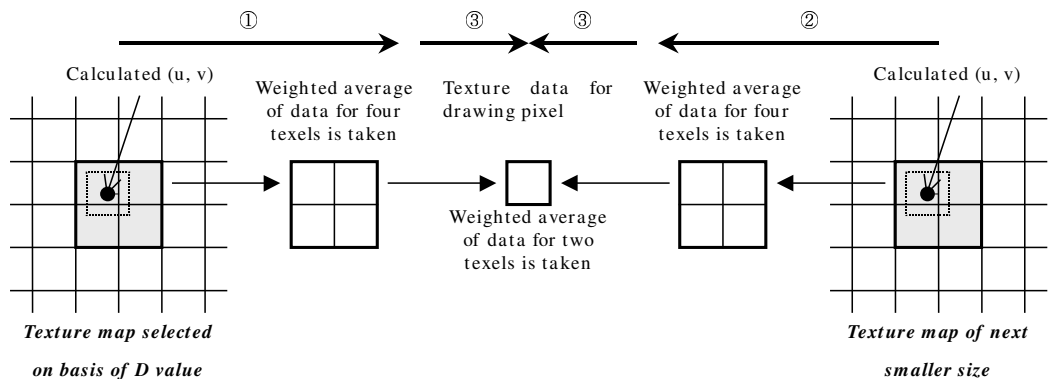


Fig. 3-20

As in the case of tri-linear filtering, it is important to note that unexpected pixel colors might be drawn, depending on the texture coordinates that are calculated. The same also applies to textures that include transparent texels.

When drawing a polygon with tri-linear filtering, the processing is performed twice for an Opaque polygon and three times for a translucent polygon. In other words, polygon parameters for two identically shaped polygons are required for an Opaque polygon, and polygon parameters for three identically shaped polygons are required for a translucent polygon.

#### <Opaque Polygons>

Drawing an Opaque polygon with tri-linear filtering is performed by performing the following two polygon processes:

- (1) Draw the polygon with "Tri-linear Pass A" specified for the Filter mode. In this case, the Blend Function should be  $DST := SRC \times "1" + DST \times "0"$  (SRC Alpha Instruction = "1", DST Alpha Instruction = "0"), and draw the polygon in the Primary Accumulation Buffer (SRC & DST select = "0"). The polygon is drawn with color data (as the SRC) produced by multiplying [1 - decimal portion of D] by the data produced through bi-linear filtering of the MIPMAP texture with the higher resolution.
- (2) Draw the polygon with "Tri-linear Pass B" specified for the Filter Mode. In this case, the Blend Function should be  $DST := SRC \times "1" + DST \times "1"$  (SRC Alpha Instruction = "1", DST Alpha Instruction = "1"), and draw the polygon in the Primary Accumulation Buffer (SRC & DST Alpha Instruction = "1"). The polygon is drawn with color data (as the SRC) produced by multiplying [decimal portion of D] by the data produced through bi-linear filtering of the MIPMAP texture with the lower resolution.

However, although the polygon for which "Pass A" was specified may be registered in an Opaque List, the polygon for which "Pass B" was specified must be registered in a translucent list.

### <Translucent Polygons>

A translucent polygon with tri-linear filtering is drawn by performing the following three polygon processes:

- (1) Draw the polygon with "Tri-linear Pass A" specified for the Filter Mode. In this case, the Blend Function should be  $DST := SRC \times "1" + DST \times "0"$  (SRC Alpha Instruction = "1", DST Alpha Instruction = "0"), and draw the polygon in the Secondary Accumulation Buffer (SRC select = "0", DST select = "1"). The polygon is drawn with color data (as the SRC) produced by multiplying [1 - decimal portion of D] by the data produced through bi-linear filtering of the MIPMAP texture with the higher resolution. At this point, [1 - decimal portion of D] is calculated for the pixel alpha values and stored in the Secondary Accumulation Buffer. Normally, it is sufficient to specify the alpha value of the final polygon for the alpha value of the Base Color.
- (2) Draw the polygon with "Tri-linear Pass B" specified for the Filter Mode. In this case, the Blend Function should be  $DST := SRC \times "1" + DST \times "1"$  (SRC Alpha Instruction = "1", DST Alpha Instruction = "0"), and draw the polygon in the Secondary Accumulation Buffer (SRC select = "0", DST select = "1"). The polygon is drawn with color data (as the SRC) produced by multiplying [decimal portion of D] by the data produced through bi-linear filtering of the MIPMAP texture with the lower resolution. At this point, [1 - decimal portion of D] is calculated for the pixel alpha values and stored in the Secondary Accumulation Buffer. Normally, it is sufficient to specify the alpha value of the final polygon for the alpha value of the Base Color.
- (3) Using the data that was produced by tri-linear filtering in the Secondary Accumulation Buffer as the SRC, draw the polygon in the Primary Accumulation Buffer (SRC select = "1", DST select = "0"). The Blend Function (SRC/DST Alpha Instruction) may be specified as desired in this case. Normally,  $DST := SRC \times "SRC\ Alpha" + DST \times "Inverse\ SRC\ Alpha"$  (SRC Alpha Instruction = 4, DST Alpha Instruction = 5). When the SRC data is stored in the Secondary Accumulation Buffer (SRC select = 1), the values in the Secondary Accumulation Buffer are used as is for the alpha and color values of the SRC pixels. The polygon's Shading Color and Texture/Shading instructions are ignored. Therefore, the alpha value of a tri-linear filtered Translucent polygon is specified by the polygon Base Colors from 1 and 2 above.

Naturally, all three polygons must be registered in a translucent list. ~~In the HOLLY2,~~  
Trilinear filtering cannot be specified for a Punch Through polygon.

### § 3.4.7.2.4 Texture Super-Sampling

Texture super-sampling can be used in combination with the three filtering modes. This function doubles the texture sampling points per pixel in both the horizontal and vertical directions [(x, y), (x + 0.5, y), (x, y + 0.5), (x + 0.5, y + 0.5)], compresses the texture and improves the quality of the portion that is drawn. However, because this quadruples the amount of texture data that is read, drawing takes about three or four times as long as compared to when this function is not used.

This function is recommended for use only when it is necessary to improve the image quality of a polygon that has a texture that has an intricate pattern or fine lines and that has been compressed. In addition, because this function yields few benefits if it is used at the same time as full-screen filtering that used the X scaler and Y scaler, it is recommended that only the full-screen filtering be used, due to the negative impact on drawing performance that the texture super-sampling function has.

### § 3.4.7.3 Bump Mapping

Bump Mapping is a method that is used to create the appearance of raised and lowered areas on a flat polygon surface by varying the brightness of the surface. The brightness of each pixel is determined by the hardware on the basis of the light source vector specified for each polygon and by the normal line vector specified for each texel (Bump Map texture).

The following two data items are specified as parameters for Bump Mapped polygons.

- (1) Bump Map parameters: K1K2K3Q (light source vector data)
- (2) Bump Map texture: SR (texel normal line vector data)

The K1K2K3Q data for Bump Mapped polygons is set in the location where the normal polygon Offset Color data is stored, and the data for the third vertex is valid (for example, as the Shading Color data for Flat Shading). In the case of a strip polygon, the data for the third and subsequent vertices is valid.

#### Bump Map parameters (specified for individual polygons)

bit 31-24	23-16	15-8	7-0
K1	K2	K3	Q

#### Bump Map textures (specified for individual texels)

bit 15-8	7-0
S	R

The RGB values of the texture data for Bump Mapped polygons are fixed to "white" (R = G = B = 0xFF), and the alpha value is the brightness (0x00 to 0xFF) that is calculated on the basis of the above parameters, where the darkest pixels are 0x00 and the brightest pixels are 0xFF. The color data for the drawing pixels is calculated from the texture data and the Base Color value according to the method specified in the Texture/Shading Instruction in the TSP Instruction Word.

When drawing the image of the Bump Mapped polygon itself, normally Decal Alpha is selected by the Texture/Shading instruction. In this case, the color of the darkest pixels is the color that is specified by the Base Color, and the color of the brightest pixels is white (the Bump Map Texel color). When using alpha blending, the pixel alpha color can be specified by the Base Color. for example, if a polygon is drawn with black (0xFF000000) specified for the Base Color for all pixels, a monochrome polygon with depressions and raised portions is produced.

#### *Bump Mapped ポリゴン (Texture/Shading Instruction = Decal Alpha)*



Fig. 3-21

### § 3.4.7.3.1 Bump Mapping Algorithm

The following section describes the operations that the hardware performs in order to derive the  $\alpha$  values for the texture data from the six 8-bit parameters (K1, K2, K3, Q, S, and R) that were specified.

The two angles that indicate the vector to a point on a hemisphere (refer to diagram below) are set in S and R, the parameters that specify the normal line vector for each texel.

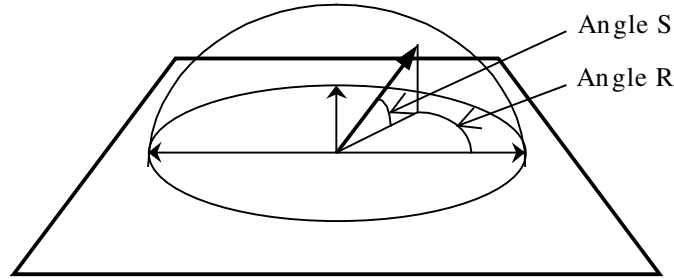


Fig. 3-22

The point indicated on the hemisphere ( $X_s, Y_s, Z_s$ ) is expressed through the following equations.

$$\begin{aligned} X_s &= \cos(s') * \cos(r') & s' &= \frac{\pi}{2} \frac{S}{256} \\ Y_s &= \sin(s') & \text{However,} & \\ Z_s &= \cos(s') * \sin(r') & r' &= 2\pi \frac{R}{256} \end{aligned}$$

In other words, the angles that express the normal line vector are specified with a value of 0 to 255, which in the case of S represents a range of angles from 0° to 90°, and in the case of R represents a range of angles from 0° to 360°. If "255" is specified, "256" (in other words, 90° or 360°) is assumed. Similarly, the light source vector can also be expressed by the following equations:

$$\begin{aligned} X_L &= \cos(t') * \cos(q') & t' &= \frac{\pi}{2} \frac{T}{256} \\ Y_L &= \sin(t') & \text{However,} & \\ Z_L &= \cos(t') * \sin(q') & q' &= 2\pi \frac{Q}{256} \end{aligned}$$

Because the brightness I of each texel is determined by the inner product of both vectors, the equation is as follows:

$$\begin{aligned} I &= X_s * X_L + Y_s * Y_L + Z_s * Z_L \\ &= \cos(s')\cos(r')\cos(t')\cos(q') + \sin(s')\sin(t') + \cos(s')\sin(r')\cos(t')\sin(q') \\ &= \sin(s')\sin(t') + \cos(s')\cos(t')\cos(r'-q') \end{aligned}$$

The alpha values for the drawing pixels are calculated by the hardware according to the following equations that allow the amount of change in the brightness to be specified so that various effects can be obtained. The Bump Map parameters K1, K2, and K3 are calculated according to the above equations, and set accordingly. ("0" is set for "0.0," and "255" is set for "1.0.")

$$\begin{aligned}\alpha &= (1-strength) + strength*I \\ &= (1-strength) + strength*\sin(s')\sin(t') + strength*\cos(s')\cos(t')\cos(r'-q') \\ &= K1 + K2*\sin(s') + K3*\cos(s')\cos(r'-q')\end{aligned}$$

However,  $K1 = 1-strength$  [ $strength = 0.0 \sim 1.0$ ]

$K2 = strength*\sin(t')$

$K3 = strength*\cos(t')$

### § 3.4.7.3.2 Bump Mapped + Textured Polygons

Bump Mapped polygons are not normally used by themselves; instead, they are used in combination with Textured polygons. Three examples of how to combine these polygons are described below. Normally, Method A is used. Although the result (the RGB value) produced by Method A and Method B is the same, the alpha value of the pixels that are drawn can be controlled for the whole polygon through method A (the alpha value is calculated from the alpha value in the Base Color of the Textured polygon and the alpha value of the texel), while method B does not permit control of the alpha value because alpha values in a Bump Mapped polygon are reflected on a pixel by pixel basis. The polygon Shading Color is specified on the Textured polygon side. In addition, both the Bump Mapped polygon and the Textured polygon are registered in a Translucent polygon list.

#### Bump Mapped polygon and Textured polygon

##### Method A

###### ① Bump Mapped polygon

Decal Alpha

SRC = One

DST = Zero

Base Color = 0xFF000000

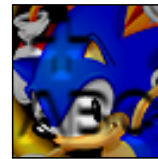
###### ② Textured polygon

Modulate Alpha

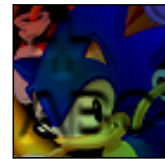
SRC = Other Color

DST = Zero

Base Color = 0xFFFFFFFF



Flat Shading



Gouraud Shading

##### Method B

###### ① Textured polygon

Modulate Alpha

SRC = One

DST = Zero

Base Color = 0xFFFFFFFF

###### ② Bump Mapped polygon

Decal

SRC = Zero

DST = SRC Alpha

Base Color = 0xFFFFFFFF

Results are the same as method A

##### Method C

###### ① Bump Mapped polygon

Decal Alpha

SRC = One

DST = Zero

Base Color = 0xFF000000

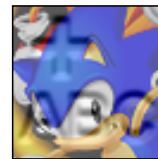
###### ② Textured polygon

Modulate Alpha

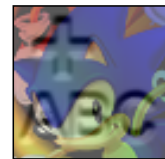
SRC = SRC Alpha

DST = Inverse SRC Alpha

Base Color = 0x80FFFFFF



Flat Shading



Gouraud Shading

Fig. 3-23



To make an image that was formed by combining a Bump Mapped polygon with a Textured polygon into a Translucent polygon, use the Secondary Accumulation Buffer. When using method A above, specify the drawing buffer in the Secondary Accumulation Buffer, and then draw a polygon with the same shape (a Flush polygon) in the Primary Accumulation Buffer. In other words, three polygons are required: (1) Bump Mapped polygon, (2) Textured polygon, and (3) Flush polygon.

Example of a Translucent polygon formed by a Bump Mapped polygon + Textured polygon

**Example of a Translucent polygon formed by a Bump Mapped polygon + Textured polygon**

<u>(1) Bump Mapped polygon</u>	<u>(2) Textured polygo</u>	<u>(3) Flush polygon</u>	<u>Final drawing polygon</u>
Decal Alpha	Modulate Alpha	[No effect]	The alpha value of the polygon pixels is calculated from the alpha value in the Base Color of the Textured polygon and the alpha value of the texel. Therefore, the transparent texels in the texture are processed as is.
SRC = One	SRC = Other Color	SRC = SRC Alpha	
DST = Zero	DST = Zero	DST = Inverse SRC Alpha	
SRC Select = 0	SRC Select = 0	SRC Select = 1	
DST Select = 1	DST Select = 1	DST Select = 0	
Base Color = 0xFF000000	Base Color = 0xFFFFFFFF	[No effect]	

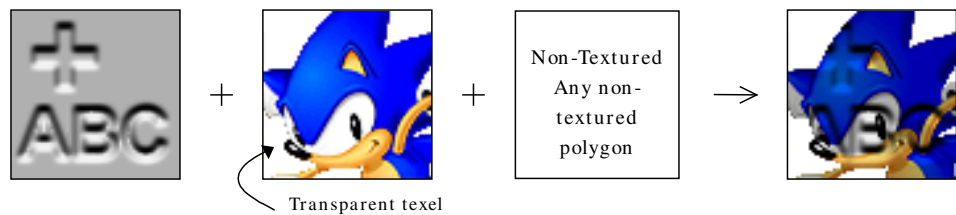


Fig. 3-24

### § 3.4.8 Fog Processing

Fog processing can be specified for each polygon individually. There are two types of Fog processing: "Look Up table mode" and "Per Vertex" mode. These modes are specified through "Fog control" in the TSP Instruction Word. The two modes can both be used within the same screen, and the Fog Color for each can be specified independently (in the FOG\_COL\_PAL register or the FOG\_COL\_VERT register). In addition, Fog processing is performed prior to the  $\alpha$  blend processing.

The equation that is used to calculate the color in Fog processing is as follows:

$$\text{Fogged\_pixel} = (1.0 - \text{Fog\_alpha}) \times \text{pixel\_col} + \text{Fog\_alpha} \times \text{Fog\_col}$$

Fogged_pixel:	Color data after Fog processing
Fog_alpha:	Fog coefficient (8-bit value)
pixel_col:	Pixel color
Fog_col:	Fog Color

#### § 3.4.8.1 Look-up Table Mode

128 Fog coefficients can be specified in the Fog table. The value that is obtained by interpolating between the two values that are retrieved from the table according to the pixel's Z (1/W) value becomes the Fog coefficient for the drawing pixel.

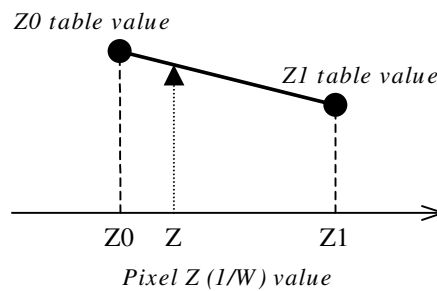


Fig. 3-25

The 1/W value, which is used as the Fog table address, is obtained by multiplying the actual Z value for the drawing pixel by the value specified in the FOG\_DENSITY register, clamped between 1.0 and 255.9999. The 7-bit Fog table address is formed as follows from the 1/W value that was calculated.

bit 6-4	3-0
Lower 3 bits for the 1/W index	Upper 4 bits for the 1/W mantissa (the sign bit and "1.0" bit are ignored)

The bit configuration of the FOG\_DENSITY register is as shown below. For example, if specifying 255.0, set 0xFF07. In this case, if the Z value of the actual drawing pixel is 1/255.0, then 1/W = 1.0.

bit 15-8	7-0
8-bit mantissa (bit 15 is the "1.0" bit)	8-bit index (two's complement)

In addition, the following equation is used to derive the 1/W value from the table address value (index):

$$1/W = (\text{pow}(2.0, \text{Index} \gg 4) \times ((\text{Index} \& 0xF) + 16) / 16.0) / \text{FogDensity};$$

The coefficient for when 1/W = 1.0 is stored at Fog table address 0 (the start of the table), and the coefficient for when 1/W = 256.0 is stored in table address 127. The 16-bit data in the Fog

table consists of two 8-bit Fog coefficients. The Fog coefficient that is stored in the upper 8 bits is the coefficient where the value of  $1/W$  is equal to that address, while the Fog coefficient that is stored in the lower 8 bits is the Fog coefficient that is used for interpolation when the Fog coefficient is larger than the address value (in other words, the Fog coefficient in the next address).

Address	bit 15-8	bit 7-0
0x00	Coefficient when address = 0x00	Coefficient when address = 0x01
0x01	Coefficient when address = 0x01	Coefficient when address = 0x02
0x02	Coefficient when address = 0x02	Coefficient when address = 0x03
.....	.....	.....
0x7E	Coefficient when address = 0x7E	Coefficient when address = 0x7E
0x7F	Coefficient when address = 0x7F	Coefficient when address = 0x7F

Look-up table mode includes processing called "Mode 2." For a polygon for which this mode is specified, the Base Color  $\alpha$  value and RGB value are replaced as follows:

Base Color  $\alpha$  value = Fog coefficient  
Base Color RGB value = Fog Color value

This mode is used for polygons for which Fog processing is to be performed after  $\alpha$  blend processing. For example, when applying a color filter (which controls the transmission ratio of each color) to a textured polygon, the textured polygon is drawn first, and then a color filter polygon is blended on top of the first polygon with "other color" or "Inverse Other Color" specified. If Fog processing is performed on each polygon individually, the resulting image will not be correct.

To draw such a polygon, first blend and draw the textured polygon and the color filter polygon with no Fog processing. Then blend a third polygon, for which Mode 2 Fog processing is specified, on top of the other two polygons with "SRC Alpha" or "Inverse SRC Alpha" specified. This approach will yield the correct image if Fog processing is applied after the two polygons are blended.

### § 3.4.8.2 Per Vertex Mode

A Fog coefficient is specified for the  $\alpha$  value of the Offset Color data for each vertex of a polygon. In the case of a polygon for which Gouraud shading was specified, the Fog coefficient for each drawing pixel is derived by interpolating from the  $\alpha$  value of the Offset Color for each vertex. When Flat Shading is specified, the Fog coefficient is also constant.

The only difference between this mode and Look Up table mode is that the Fog coefficient is not retrieved from a table according to the Z value; instead, it is derived from the value specified for each vertex. The color operation equations that use the resulting Fog coefficient are completely identical in the two modes. In addition, with the normal Look Up table mode, once the table has been set, the hardware performs Fog processing automatically. In "Per Vertex" mode, however, the CPU has to calculate and set the Fog coefficient ( $\alpha$  value) for each vertex each time, according to the polygon's position. This increases the load on the CPU. However, there are some effects that can be implemented in "Per Vertex" mode that cannot be implemented in Look Up table mode (for example, creating a Fog effect based on the Y value instead of the Z value).

Polygons for which this mode is specified must be set up so that an Offset Color is used (Offset bit = 1). If the polygon is not set up to use an Offset Color, Fog processing is not performed.

### § 3.4.9 Clipping

There are two types of clipping: Tile Clipping, which is performed on individual Tiles by the TA; and pixel clipping, which is performed on individual pixels when they are written to the frame buffer.

#### § 3.4.9.1 Tile Clipping

Polygon data that is input to the TA can be clipped at the individual Tile level, so that polygon data (object) that is completely outside of the specified clipping area is not stored in texture memory. the Tile Clipping area in the TA is determined by the "Global Tile Clip area" (which is specified by the TA\_GLOB\_TILE\_CLIP register), and the "User Tile Clip area" (which is specified by the User Tile Clip Control Parameters). Because the Global Tile Clip specification is a register specification, it can only be specified for an entire screen. The User Tile Clip specification can be selected for individual objects as either "off," "enabled inside area," or "enabled outside area." The size of the area can also be set individually for each object. (Refer to section 3.7.3.3.)

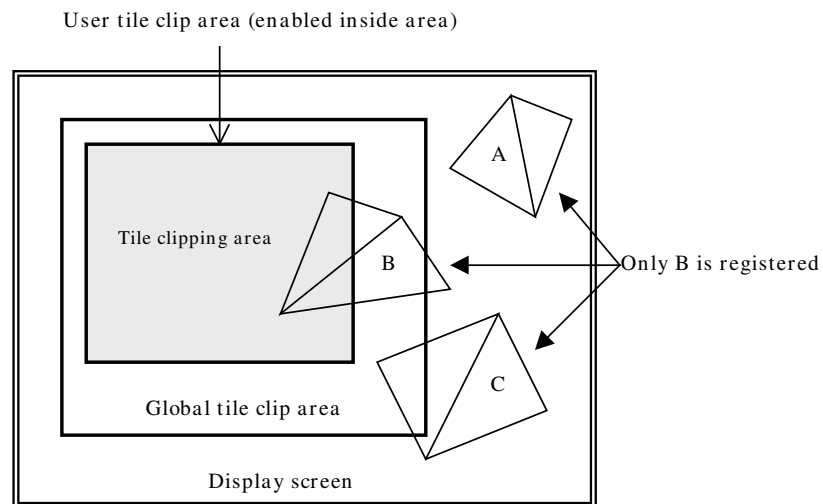


Fig. 3-26

For example, when drawing a screen that is divided into two Tiles as shown below, Tile Clipping only needs to be performed when inputting polygon for each data into the TA.

*Example of parameter input for TA*

User Tile Clip parameter that specified the left side of the screen
Opaque polygon data needed for the left side of the screen
User Tile Clip parameter that specified the right side of the screen
Opaque polygon data needed for the right side of the screen
Opaque end of list
User Tile Clip parameter that specified the left side of the screen
Opaque polygon data needed for the left side of the screen
User Tile Clip parameter that specified the right side of the screen
Opaque polygon data needed for the right side of the screen
Translucent end of list

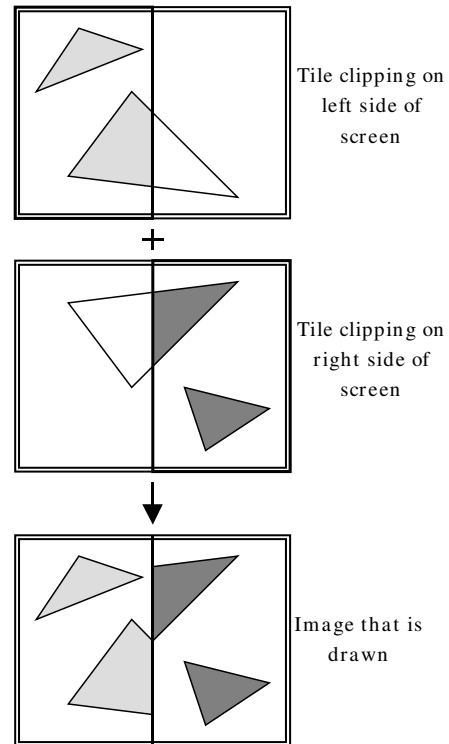


Fig. 3-27

### § 3.4.9.2 Pixel Clipping

Pixel data that is transferred from the accumulation buffer in the CORE to the frame buffer in texture memory can be clipped at the individual pixel level so that data that is outside of the specified clipping area is not stored in the frame buffer (but the polygon is drawn). Because the pixel clipping area is specified by the FB\_X\_CLIP register and the FB\_Y\_CLIP register, only one area can be specified on one screen. Note that pixels that are in the positions specified by the register are deemed to be inside of the area, and are therefore stored in the frame buffer.

If the size of the display screen is larger than that of the clipping area, any data that remains in frame buffer is displayed as is in the portion of the screen that lies outside of the clipping area.

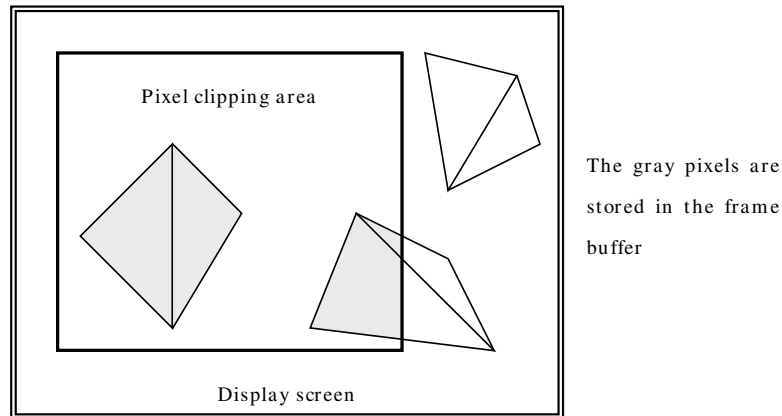


Fig. 3-28

For example, when drawing a screen such as the one shown below where the window area does not coincide with Tile boundaries, use pixel clipping and draw twice within one frame.

*Example of parameter input and drawing processing*

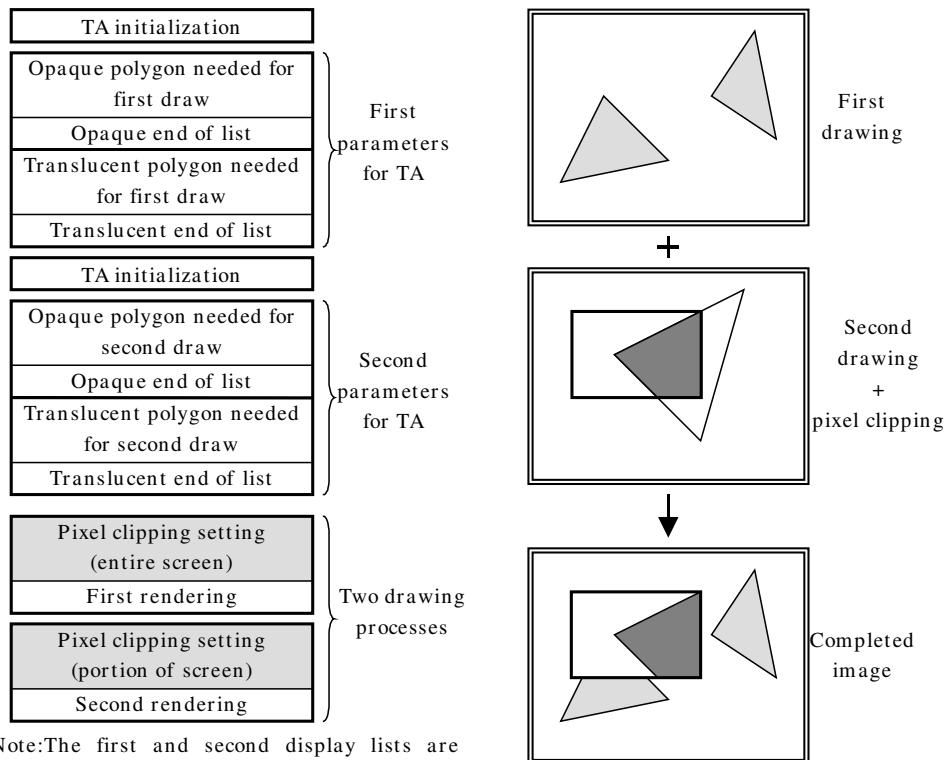


Fig. 3-29

### § 3.4.10 Drawing to a Texture Map

The pixel data that is drawn in the accumulation buffer in the CORE is transferred to the texture memory address specified by the FB\_W\_SOF1 register and the FB\_W\_SOF2 register. These registers can be used to specify whether to store the screen data that has been drawn as texture data for subsequent use, or as frame buffer data for a TV display.

Register	Specified addresses	Access area
FB_W_SOF1,	0x00000000~0x0FFFFFFC	32-bit (frame buffer)
FB_W_SOF2	0x10000000~0x1FFFFFFC	64-bit (texture data)

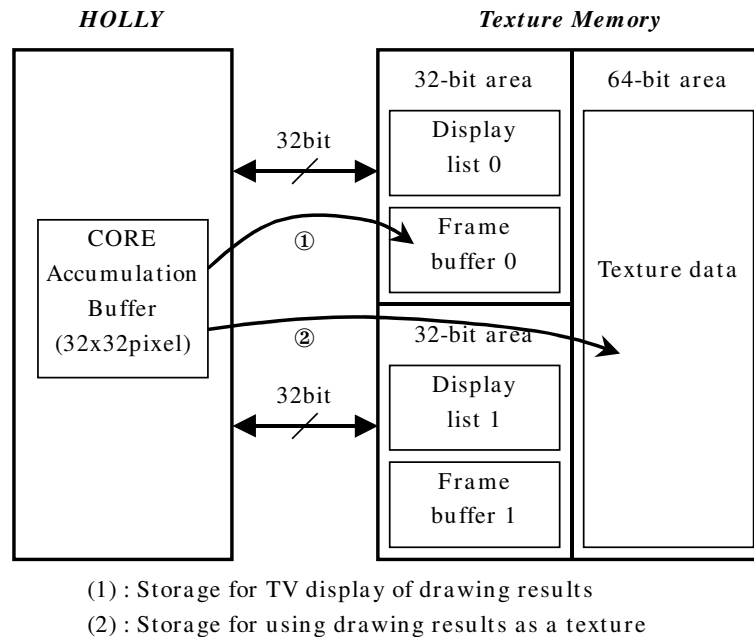


Fig. 3-30

The data that is stored in the 32-bit area is separate from the data that is stored in the 64-bit area, and frame buffer data cannot be used as texture data and texture data cannot be used for TV display. Therefore, when performing environment mapping using the drawing results, the first drawing results are stored in a 64-bit area, and then the results of the second drawing that was done using the texture data is stored in the 32-bit area.

When texture data is stored in a 64-bit area, the data is stored according to the same frame buffer-related register settings as when stored in a frame buffer, so it is necessary to set the registers in a way that will produce correct texture data. Furthermore, when the drawing results that are stored in the 64-bit area are to be used for a texture, the texture format will be either Non-Twiddled Rectangular format or Stride format.

FB_W_CTRL bit 2-0	Pixel format	When drawing to a texture map
0	0555 KRGB 16 bit	Can be used for drawing to a texture map
1	565 RGB 16 bit	
2	4444 ARGB 16 bit	
3	1555 ARGB 16 bit	
4	888 RGB 24 bit	Cannot be used for drawing to a texture map
5	0888 KRGB 32 bit	
6	8888 ARGB 32 bit	
7	Reserved	Cannot be used

### § 3.4.11 X Scaler & Y Scaler

The X scaler and the Y scaler perform filtering and scaling in the X and Y directions when transferring pixel data from the accumulation buffer in the CORE to the frame buffer in texture

memory. Because this filtering and scaling is not performed when the pixel data is transferred from the frame buffer to the DAC, the image that is displayed on the screen is identical to the pixel data in the frame buffer.

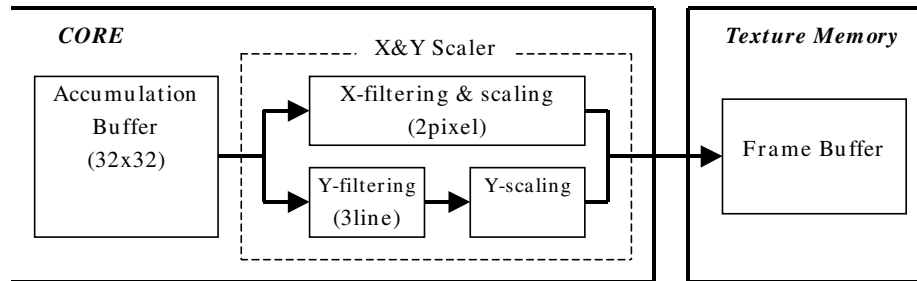


Fig. 3-31

### § 3.4.11.1 X Scaler

The X scaler filters every two pixels of pixel data in the X direction that is being drawn, scaling down the image by 1/2. Whether or not to use this function can be specified in the SCALER\_CTL register.

The filtering coefficient is fixed at 0.5, and the averaged data of two adjacent pixels in the accumulation buffer is stored in the frame buffer as data for one pixel. Therefore, when using X filtering, it is necessary to draw with double the display resolution in the X direction.

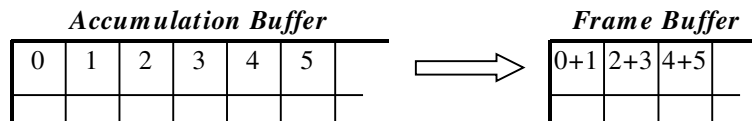


Fig. 3-32



### § 3.4.11.2 Y Scaler

The Y scaler filters three lines of pixel data in the Y direction that is being drawn, scaling the image as specified. Filtering is performed only when scaling an image down, not when scaling an image up. When scaling an image down, the filtering coefficient is specified in the Y\_COEFF register. The scaling coefficient is specified in the SCALER\_CTL register.

When using Y filtering at the drawing resolution in the Y direction, it is sufficient to specify a reduction coefficient as close to 1.0x as possible (0x0401).

Y scaling produces an image by interpolation of lines above and below according to the results of line position calculation.

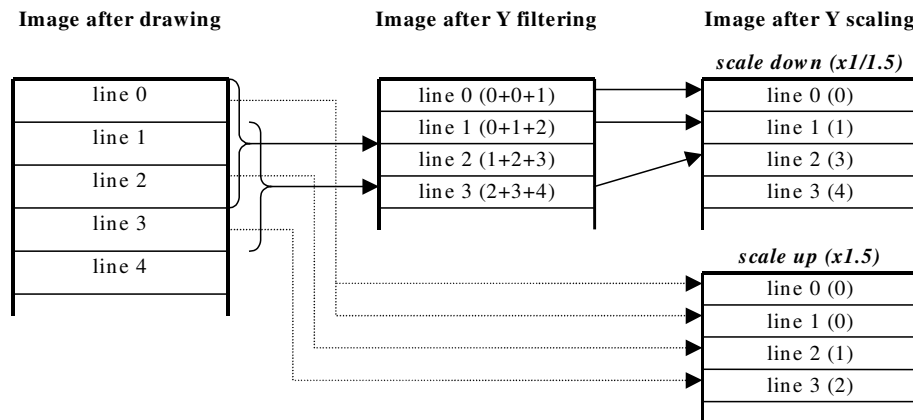


Fig. 3-33

### § 3.4.12 Flicker-free Interlacing

With an interlaced display, the Y scaler can be used to implement flicker-free filtering. However, because the Tiles must be drawn in sequence in the Y direction, the Region array data (refer to section 3.7) must be stored (arranged vertically) so that it is drawn in the Y direction.

There are two methods for implementing flicker-free filtering; the screen drawing intervals, the frame buffer memory size, and read control for display differ for each method.

Type	Screen drawing interval	Frame buffer memory size	Read control for display
A	1/30 second (NTSC) or 1/25 second (PAL)	480 lines	Shift the start address one line for each field, skipping one line at a time when reading
B	1/60 second (NTSC) or 1/50 second (PAL)	240 lines	Display as is

### § 3.4.12.1 Type A

First, set the Y scaler scaling coefficient so that the reduction coefficient is as close to 1.0x (0x0401) as possible, and enable Y filtering. The result of filtering three lines of data is then stored in the frame buffer. In other words, the result of drawing 480 lines is stored in the frame buffer as 480 lines. Therefore, the screen needs to be drawn for each frame (in units of 1/30 or 1/25 seconds).

When displaying the image, shift the start address for the read from the frame buffer one line for each field, skipping one line at a time.

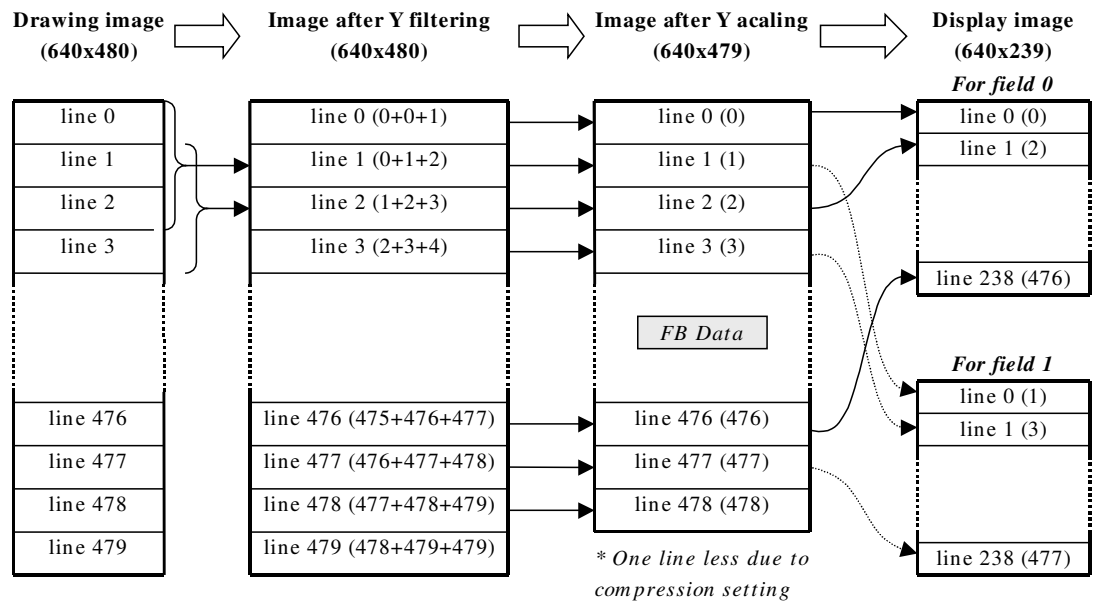


Fig. 3-34

Make the following settings in order to implement type A flicker-free filtering:

- (1) Set the scaling coefficient for the Y direction so that the reduction coefficient is as close to 1.0x as possible.  
(Set the "Vertical Scale Factor" in the SCALER\_CTL register to "0x0401".)
- (2) Control reads from the frame buffer for display one field at a time.

### § 3.4.12.2 Type B

First, perform filtering with 3 lines of data in the Y scaler, scale the data by 1/2, and then store in the frame buffer only that line data that is needed for the specified display field. In other words, although drawing must be performed with 480 lines, only 240 lines are needed for the pixel data that is stored in the frame buffer. However, because this processing is performed for individual Tiles when the pixel data that was drawn is transferred to the frame buffer, the data must be drawn in every field (in units of 1/50 or 1/60 second).

For display, the data only needs to be read as is from the frame buffer.

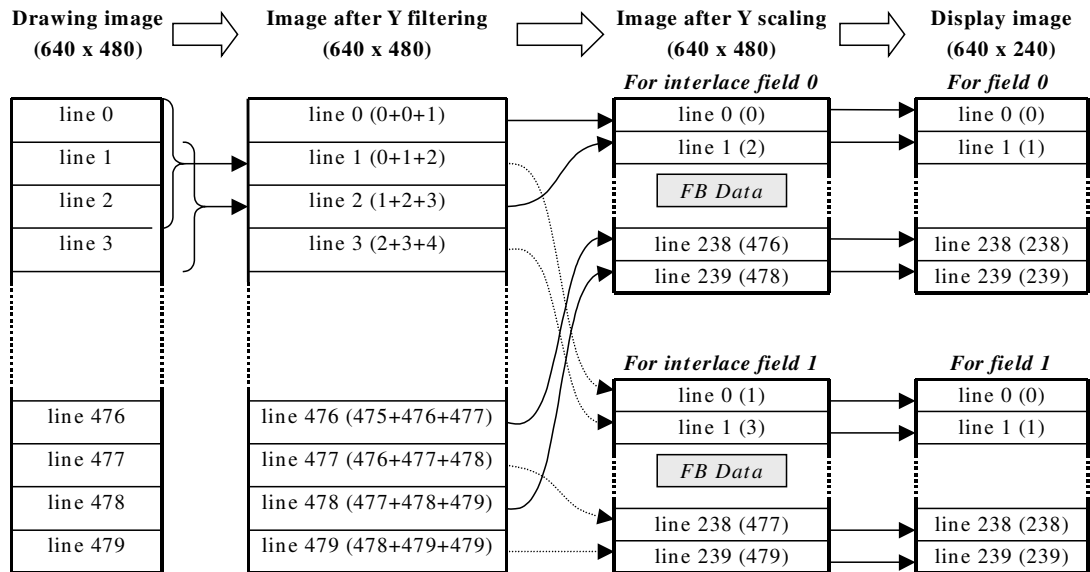


Fig. 3-35

Make the following settings in order to implement type B flicker-free filtering:

- (1) Set the scaling coefficient for the Y direction for 1/2 reduction. (Set the "Vertical Scale Factor" in the SCALER\_CTL register to "0x0800".)
- (2) Set the Y scaler to interlace mode. (Set "Interlace" in the SCALER\_CTL register to "1".)
- (3) Switch the Y scaler setting back and forth between field 0 and field 1 in accordance with the screen display. (Toggle "Field Select" in the SCALER\_CTL register.)

### § 3.4.13 Strip Buffers

A strip buffer is a pixel data buffer that retains pixel data that has been drawn in Tile units for the specified number of lines, rather than an entire screen; in other words, a compressed frame buffer. Area for two strip buffers is allocated in texture memory; the size of one buffer can be specified over a range of 32 to 1024 lines, in units of 32 lines. The starting address of the strip buffer can be specified in the FB\_W\_SOF1 register and the FB\_W\_SOF2 register, and the strip buffer size can be specified in the FB\_R\_CTRL register.

Strip buffer processing is synchronized with the TV display, and performs the following operations:

- (1) Stores 32-pixel × 32-pixel data drawn in the accumulation buffer into strip buffer 1 in texture memory.
- (2) Once strip buffer 1 has been filled with pixel data, the process of displaying the contents of strip buffer 1 on the screen begins, and the drawing pixel data for the next Tile is stored in strip buffer 2.
- (3) Once strip buffer 2 has been filled with pixel data, the process waits until all of the pixel data in strip buffer 1 has been displayed on the screen. Once this happens, strip buffer 2 becomes the screen display strip buffer, and strip buffer 1 becomes the pixel data storage strip buffer again.
- (4) Steps (1) through (3) are repeated until all of the lines on the display screen have been displayed.

When the strip buffers are used, less memory is required when compared to the frame buffer. However, the strip buffers are effective only in the following cases:

- When polygons are uniformly positioned over the entire screen. In other words, when the difference between the drawing time and the display time for the strip buffer size is small.
- When the number of polygons being drawn is small.

Because the strip buffers must operate in synchronization with the TV screen display, the drawing time required for the buffer size must not be longer than the corresponding screen display time. In other words, it is also essential that the strip buffer size be specified so that "drawing time < display time." The timing of drawing must be such that drawing is completed before the strip buffer is displayed at the beginning of the screen. In other words, drawing must be started before the start of screen display by at least a margin equal to the time needed to display data equivalent to the size of the strip buffer. For example, when the strip buffer contains 64 lines, drawing must start at least 64 lines before the start of screen display. When drawing to the strip buffer is not completed in time for the screen display, the strip buffer is forcibly switched, drawing is halted, and an interrupt is generated. Naturally, the screen is not displayed correctly in this case.

When polygons are concentrated in a certain portion of the screen, the size of the strip buffer must be increased in order to maintain the relationship "drawing time < display time," which is a restriction of the strip buffer. However, doing this will reduce drawing performance for processing of screen areas with few polygons, as processing will not proceed to the next drawing even though the previous one is quickly completed until termination of display.

When using the strip buffers, the region data array (see section 3.7) must be stored in such a manner that drawing proceeds in the X direction (horizontally). In addition, when there are 240 display screen lines, 256 lines are drawn and stored in the strip buffer, but the last 16 lines are not displayed.

The value that is specified for the screen buffer size must yield an even number when the number of display screen lines is divided by that value. Normally, in the case of NTSC and PAL (both interlaced and non-interlaced), the number of display screen lines is 240, so the strip buffer size should be either 32, 64, or 128. In the case of VGA, the number of display screen lines is 480, so the strip buffer size should be either 64, 128, or 256.

Furthermore, the X clipping function cannot be used. In other words, the size of the display screen in the horizontal direction must be specified for the X clipping values in the FB\_X\_CLIP register. For example, when the size of the display screen in the horizontal direction is 640 pixels, specify FB x clipping max = 639 and FB x clipping min = 0.

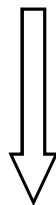
### § 3.4.14 Frame Buffer Drawing Data and Display Data

When drawing in Tile units in the CORE, the pixel data is 8-bit ARGB data, and ARGB are each processed as 8 bits for shading and alpha blending. When this pixel data that has been drawn is transferred to the frame buffer in texture memory, it is converted into the pixel format specified in the FB\_W\_CTRL register. The "4444 ARGB 16-bit" format that can be specified in FB\_W\_CTRL is a special format for drawing to the texture map only, and cannot be used to draw to the frame buffer.

Furthermore, when outputting to the DAC for the purpose of display, the data is read from the frame buffer in the pixel format that was specified in the FB\_R\_CTRL register, and is converted into Chroma + RGB 8-bit format. The chroma bit is used for forming a composite with an external screen, and is not normally used.

#### Pixel data in CORE

Alpha:8bit	Red:8bit	Green:8bit	Blue:8bit
------------	----------	------------	-----------



**fb\_kval value in the FB\_W\_CTRL register**

K:8bit
--------

**fb\_alpha\_threshold value in the FB\_W\_CTRL register**

A:8bit
--------

#### Pixel data in the frame buffer (select from among the following six types)

<i>0555 KRGB 16bit</i>			
K:1bit	R Upper 5bit	G Upper 5bit	B Upper 5bit

K: Value of uppermost bit in fb\_kval

RGB: Can turn dither processing on/off

<i>565 RGB 16bit</i>			
R Upper 5bit	G Upper 6bit	B Upper 5bit	

RGB: Can turn dither processing on/off

<i>1555 ARGB 16bit</i>			
A:1bit	R Upper 5bit	G Upper 5bit	B Upper 5bit

A: Determined by comparison with fb\_alpha\_threshold

RGB: Can turn dither processing on/off

<i>888 RGB 24bit packed</i>			
R:8bit	G:8bit	B:8bit	

RGB: CORE values, as is

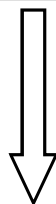
<i>0888 KRGB 32bit</i>			
K:8bit	R:8bit	G:8bit	B:8bit

K: fb\_kval value

RGB: CORE values, as is

<i>8888 ARGB 32bit</i>			
A:8bit	R:8bit	G:8bit	B:8bit

Data in CORE, as is



**fb\_concat value in FB\_R\_CTRL register**

3bit
------

**fb\_chroma\_threshold value in FB\_R\_CTRL register**

8bit
------

#### Pixel data during display

Chroma:1bit	Red:8bit	Green:8bit	Blue:8bit
-------------	----------	------------	-----------

Chroma is determined by comparison with fb\_chroma\_threshold

When the RGB data in the frame buffer is less than 8 bits, the fb\_concat value is added at the lower end of the data.

Fig. 3-36

When the pixel format in the frame buffer is 16 bits, it is important to note that the lower bit values of RGB that are discarded when the data is transferred from the CORE to the frame buffer differ from the lower bit values of RGB that are added when the data is output from the frame buffer to the DAC.

## § 3.5 Display Function Details

### § 3.5.1 Sync Pulse Generator

HOLLY supports display on both NTSC and PAL TVs and monitors. HOLLY includes a block called the "SPG" (Sync Pulse Generator) that generates the sync signals. Certain registers must be set in accordance with the display standard. For details on the registers, refer to section 8.4.2.

The settings for each of the registers in the SPG block for different display modes are listed below.

Register name	NTSC Non-interlace	NTSC Interlace	PAL Interlace	PAL Interlace	VGA
	320x240 640x240	320x240 640x240 640x480	320x240 640x240	320x240 640x240 640x480	640x480
SPG_LOAD	0x01060359	0x020C0359	0x0138035F	0x0270035F	0x020C0359
SPG_HBLANK	0x007E0345	0x007E0345	0x008D034B	0x008D034B	0x007E0345
SPG_VBLANK	0x00120102	0x00240204	0x002C026C	0x002C026C	0x00280208
SPG_WIDTH	0x03F1933F	0x07D6C63F	0x07F1F53F	0x07D6A53F	0x03F1933F
SPG_CONTROL	0x000000140	0x000000150	0x000000180	0x000000190	0x000000100
VO_STARTX	0x000000A4	0x000000A4	0x000000AE	0x000000AE	0x000000A8
VO_STARTY	0x00120011	0x00120012	0x002E002E	0x002E002D	0x00280028
VO_CONTROL	In case of 320 : 0x00160100 In case of 640 : 0x00160000	In case of 320 : 0x00160100 In case of 640 : 0x00160000	In case of 320 : 0x00160100 In case of 640 : 0x00160000	In case of 320 : 0x00160100 In case of 640 : 0x00160000	0x00160000

Note: When interlaced, the 240 lines are single-interlaced.

The screen display positions for the sync signals are specified in the VO\_STARTX and VO\_STARTY registers.

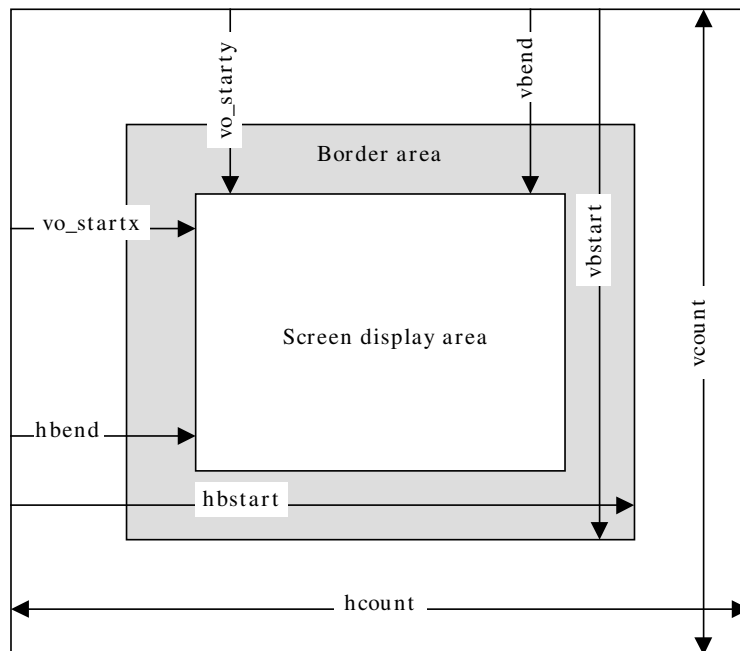


Fig. 3-37

### § 3.5.2 Frame Buffer Settings

The following eight registers are used for the frame buffer settings. For details on the contents of each register, refer to section 8.4.2.

Register name	Description of settings
FB_R_CTRL	This register is used for settings concerning reads from the frame buffer. <i>vclk_div</i> : Pixel clock setting <i>fb_strip_buf_en</i> : Strip buffer enable <i>fb_stripsize</i> : Strip buffer size <i>fb_chroma_threshold</i> : Comparison $\alpha$ value for chroma output <i>fb_concat</i> : Lower bit value for concatenation in RGB output <i>fb_depth</i> : Frame buffer pixel format <i>fb_line_double</i> : Line double read enable <i>fb_enable</i> : Frame buffer read enable
FB_W_CTRL	This register is used for settings concerning writes to the frame buffer. <i>fb_alpha_threshold</i> : Comparison value for $\alpha$ value writes <i>fb_kval</i> : Upper bit value for concatenation during a write <i>fb_dither</i> : Dithering enable <i>fb_packmode</i> : Frame buffer pixel format
FB_W_LINESTRIDE	Specifies the line width (in units of 64 bits) for writes to the frame buffer.
FB_R_SOF1	Specifies the starting address for reads from the frame buffer for field 1.
FB_R_SOF2	Specifies the starting address for reads from the frame buffer for field 2.
FB_R_SIZE	Specifies the size when reading from the frame buffer. <i>FB modulus</i> : Amount of data from the end of a line to the data for the next line <i>FB y size</i> : Number of lines in the frame buffer <i>FB x size</i> : Number of pixels in the frame buffer
FB_W_SOF1	Specifies the starting address for writes to the frame buffer for field 1.
FB_W_SOF2	Specifies the starting address for writes to the frame buffer for field 2.

The following diagram illustrates the settings for reading from the frame buffer.

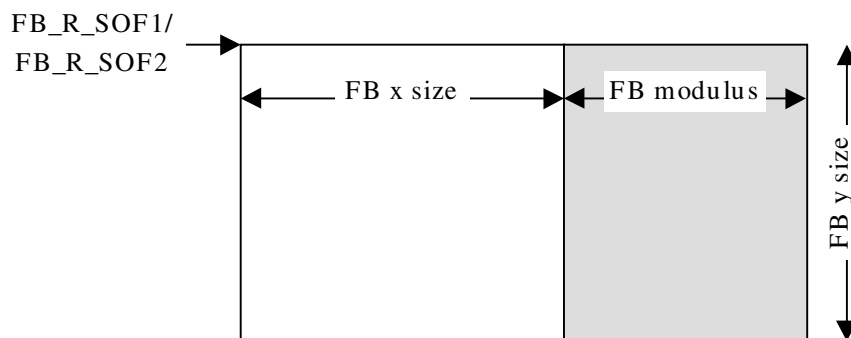


Fig. 3-38



### § 3.6 Texture Definition

The textures coordinates U and V are both normally specified in a range from (0.0 , 0.0) to (1.0, 1.0) with 32-bit IEEE floating-point values. It is also possible to discard the lower 16 bits and specify the coordinates with 16-bit floating-point values. When using 16-bit values, both coordinates are specified as a 32-bit value, with the upper 16 bits representing U and the lower 16 bits representing V.

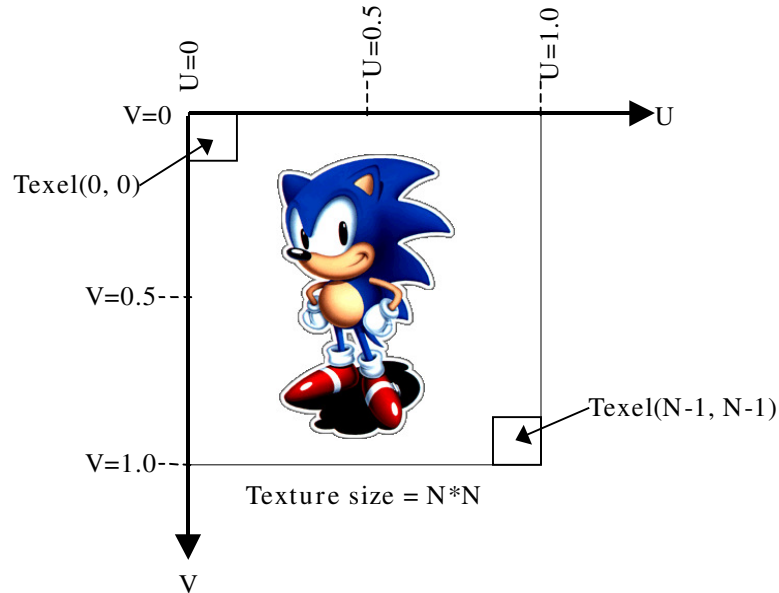


Fig. 3-39 Texture Definition

### § 3.6.1 Texture Pixel Format

The texture pixel formats that can be used are listed below. These formats are specified through "Pixel Format" in the Texture Control Word.

Type	Bit configuration	Description
RGB	1555	$\alpha$ value: 1 bit; RGB values: 5 bits each
	565	R value: 5 bits; G value: 6 bits; B value: 5 bits
	4444	$\alpha$ value: 4 bits; RGB values: 4 bits each
YUV	32bit/2texel	YUV 422 data, 8 bits each
Bump Map	16bit/texel	S value: 8 bits; R value: 8 bits
Palette	4bit/texel	16 colors per texture
	8bit/texel	256 colors per texture

Table 3-3 Pixel Formats

#### § 3.6.1.1 RGB Textures

RGB textures are expressed by 16 bits per texel. There are three different color formats.

##### RGB1555 Texture

bit 15	14-10	9-5	4-0
Alpha	Red	Green	Blue

##### RGB565 Texture

bit 15-11	10-5	4-0
Red	Green	Blue

##### RGB4444 Texture

bit 15-12	11-8	7-4	3-0
Alpha	Red	Green	Blue

#### § 3.6.1.2 YUV Textures

YUV textures are expressed by 16 bits per texel, and one data item (YUV422 data) corresponds to two adjacent texels in the horizontal direction. The Y data for the left texel and the U data for both texels are collectively referred to as "YoU" data, and the Y data for the right texel together with the V data for both texels are referred to as "Y1V" data. Each YUV data element is specified by an unsigned 8-bit value from 0 to 255.

MPEG data (YUV420 data in macro block units) is converted into YUV texture data by passing the data through the YUV data converter in the Tile Accelerator. (Refer to section 3.8.1.)

##### YoU-data

bit 15-8	7-0
Yo	U

##### Y1V-data

bit 15-8	7-0
Y1	V

The YUV texture data is converted within the CORE according to the following equations into RGB values for drawing. Note that the RGB values that are computed are clamped in the range 0 to 255.

$$R = Y + (11/8) \times (V-128)$$

$$G = Y - 0.25 \times (11/8) \times (U-128) - 0.5 \times (11/8) \times (V-128)$$

$$B = Y + 1.25 \times (11/8) \times (U-128)$$

$$\alpha = 255$$

### § 3.6.1.3 Bump Map Textures

Bump Map textures are expressed by 16 bits per texel; two 8-bit parameters are specified to express the normal line vector for each texel.

#### Bump Map Texture

bit 15-8	7-0
S	R

The 8-bit parameters that are specified, S and R, set the two angles that define the vector to a point on a hemisphere, as shown in the illustration below.

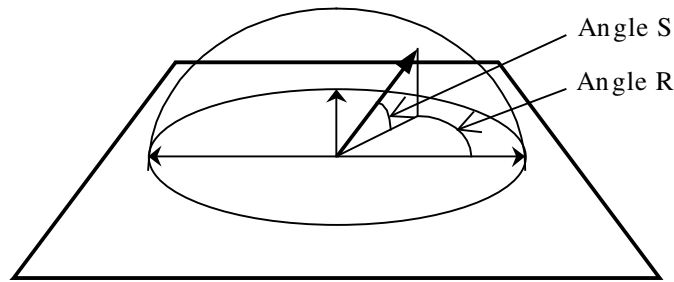


Fig. 3-40

A point (x, y, z) on the hemisphere is expressed through the following equations:

$$\begin{aligned}
 x &= \cos(s') * \cos(r') & s' &= \frac{\pi}{2} \frac{S}{256} \\
 y &= \sin(s') & \text{However,} & \\
 z &= \cos(s') * \sin(r') & r' &= 2\pi \frac{R}{256}
 \end{aligned}$$

In other words, the angles that express the normal line vector are specified with a value of 0 to 255, which in the case of S represents a range of angles from 0° to 90°, and in the case of R represents a range of angles from 0° to 360°. If "255" is specified, "256" (in other words, 90° or 360°) is assumed.

For details on the Bump Mapping algorithm, refer to section 3.4.7.3.1.

#### § 3.6.1.4 Palette Textures

Palette textures are expressed by four or eight bits per pixel ("4BPP" or "8BPP," hereafter), which indicate the low-order address byte in palette RAM in the CORE. When the value specified by the palette selector in the Texture Control Word is added as the high-order address byte, the result is the address in palette RAM. In the case of 8BPP format, only the upper two bits of the palette selector are valid. Up to 1024 colors can be set in palette RAM.

##### 4BPP Palette Texture

bit 9-4	3-0
Palette Selector (bit26-21)	Texture Data

##### 8BPP Palette Texture

bit 9-8	7-0
Palette Selector (bit26-25)	Texture Data

The following table lists the four color data formats that can be specified in palette RAM. Only one format can be specified per screen, in the PAL\_RAM\_CTRL register. Multiple color data formats cannot co-exist. It is important to note that if a Filter Mode other than point sampling is set with the ARGB8888 format, drawing performance will be only about 50% of normal drawing performance.

Format	Description
ARGB1555	$\alpha$ value: 1 bit; RGB values: 5 bits each
RGB565	$\alpha$ value: none; R value: 5 bits; G value: 6 bits; B value: 5 bits
ARGB4444	$\alpha$ value: 4 bit; RGB values: 4 bits each
ARGB8888	$\alpha$ value: 8 bit; RGB values: 8 bits each

##### ARGB1555 Palette

bit 15	14-10	9-5	4-0
Alpha	Red	Green	Blue

##### RGB565 Palette

bit 15-11	10-5	4-0
Red	Green	Blue

##### ARGB4444 Palette

bit 15-12	11-8	7-4	3-0
Alpha	Red	Green	Blue

##### ARGB8888 Palette

bit 31-24	23-16	15-8	7-0
Alpha	Red	Green	Blue

### § 3.6.2 Texture Formats

The texture shapes that can be used are square and rectangular. There are eight sizes (represented by values of  $2^n$ , ranging from 8 to 1024) that can be set for the U size and the V size, each, in the TSP Instruction Word. If the same size is specified for U and V, the texture shape is square; if different sizes are specified, the texture shape is rectangular.

One type of rectangular texture is called a "stride texture," for which a multiple of 32 (from 32 to 512) is specified for the U size. This type of texture can be used when the result of drawing is to be used as a texture. Stride textures only support Non-Twiddled format. Because the U size is specified in the TEXT\_CONTRL register, only one U size can be specified on one screen.

There are two formats for storing texture data in texture memory: Twiddled format and Non-Twiddled format. Furthermore, Twiddled format can be either compressed format or non-compressed format. In addition, among Twiddled format textures, there are textures known as MIPMAP textures, which store multiple textures that are switched according to the Z value of the polygon.

Storage format	Compressed/ non-compressed	MIPMAP	Texture format
Twiddled format	Compressed	MIPMAP	Square
		Individual	Square
	Non-compressed	MIPMAP	Square
		Individual	Square Rectangular
Non-Twiddle	Non-compressed	Individual	Square
			Rectangular
			Stride

#### § 3.6.2.1 Twiddled Format

Twiddled-format texture data is stored in a special order (a reverse "N") shown in the diagram below in order to minimize performance loss when reading texture data for drawing. Normal textures use this format. The Twiddled format specification is made through "scan order" in the Texture Control Word. (Refer to section 3.7.9.3.)

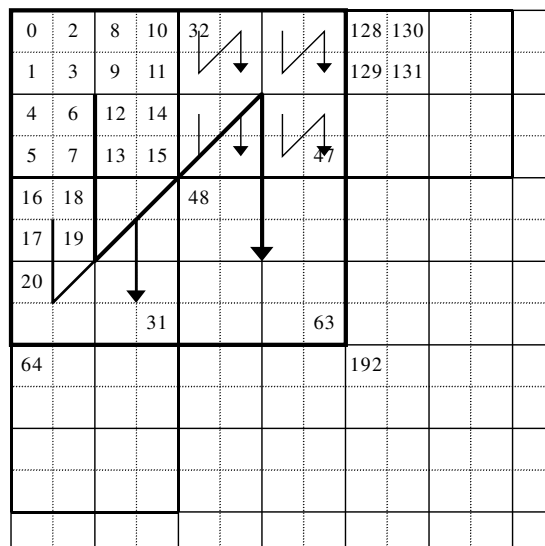


Fig. 3-41 Twiddle Format

Twiddled format textures can be either square or rectangular. The relationship between the texture data storage address and the UV coordinates is shown below.

**<Squares>**

The bits of the storage address are configured so that each bit of the UV coordinates alternate, starting from the low-order end. The least significant bit is bit 0 of the V coordinate (Vo).

Example:                ..... U4 V4 U3 V3 U2 V2 U1 V1 U0 Vo

**<Rectangles>**

The bits of the storage address are configured so that each bit of the UV coordinates alternate, starting from the low-order end. The least significant bit is bit 0 of the V coordinate (Vo). Any extra bits for one coordinate are positioned in order at the high end.

Example:                ..... V5 V4 U3 V3 U2 V2 U1 V1 U0 Vo

Twiddled format textures support all pixel formats. The data configuration for each type of data is listed below.

**RGB & Bump Map Texture**

bit 63-48	47-32	31-16	15-0
Texel (1,1)	Texel (1,0)	Texel (0,1)	Texel (0,0)

**YUV Texture**

bit 63-48	47-32	31-16	15-0
Y1V (1,1)	Y1V (1,0)	YoU (0,1)	YoU (0,0)

**4BPP Palette Texture**

bit 63-60	59-56	55-52	51-48	47-44	43-40	39-36	35-32
Texel (3,3)	Texel (3,2)	Texel (2,3)	Texel (2,2)	Texel (3,1)	Texel (3,0)	Texel (2,1)	Texel (2,0)
bit 31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Texel (1,3)	Texel (1,2)	Texel (0,3)	Texel (0,2)	Texel (1,1)	Texel (1,0)	Texel (0,1)	Texel (0,0)

**8BPP Palette Texture**

bit 63-56	55-48	47-40	39-32	31-24	23-16	15-8	7-0
Texel (1,3)	Texel (1,2)	Texel (0,3)	Texel (0,2)	Texel (1,1)	Texel (1,0)	Texel (0,1)	Texel (0,0)

<Note> The numbers in parentheses are the UV coordinates.

### § 3.6.2.2 Non-Twiddled Format

Non-Twiddled format texture data is stored in sequence, similar to bitmapped data. This format is used when the drawing results are to be used as texture data.

However, the drawing performance for this format is low compared to Twiddled format.

Example: When U = 16 pixels and V = 8 pixels

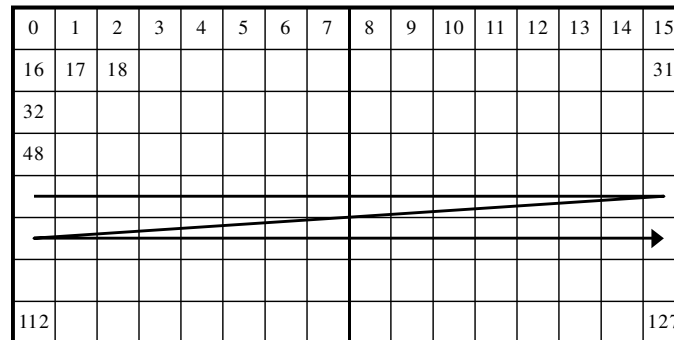


Fig. 3-42 Non-Twiddled Format

Non-Twiddled format textures support all shapes: square, rectangular, and stride. The relationship between the texture data storage address and the UV coordinates is shown below.

#### <Squares and rectangles>

$$(\text{texture data storage address}) = (\text{V size}) \times (\text{V coordinate}) + (\text{U coordinate})$$

#### <Stride>

$$(\text{texture data storage address}) = (\text{stride value}) \times 32 \times (\text{V coordinate}) + (\text{U coordinate})$$

However, "stride" corresponds to bits 4 through 0 of the TEXT\_CONTROL register.

Non-Twiddled formats support all pixel formats, except for palette textures. For example, the data configuration (64 bits) of texture data with a size of 128 x 128 is as follows.

#### RGB & Bump Map Texture

Address	bit 63-48	47-32	31-16	15-0
0x00	Texel (3,0)	Texel (2,0)	Texel (1,0)	Texel (0,0)

.....

Address	bit 63-48	47-32	31-16	15-0
0x80	Texel (3,1)	Texel (2,1)	Texel (1,1)	Texel (0,1)

#### YUV Texture

Address	bit 63-48	47-32	31-16	15-0
0x00	Y1V (3,0)	YoU (2,0)	Y1V (1,0)	YoU (0,0)

.....

Address	bit 63-48	47-32	31-16	15-0
0x80	Y1V (3,1)	YoU (2,1)	Y1V (1,1)	YoU (0,1)

<Note> The numbers in parentheses are the UV coordinates.

### § 3.6.2.3 VQ Textures

One type of Twiddled format texture is a compressed texture data format that is compressed from 1/3 to 1/8 the normal size by a method called "VQ (Vector Quantization) compression." Textures stored in this format are called "VQ textures." This format is supported only for square RGB pixel format. The VQ texture specification is made through "VQ compressed" in the Texture Control Word. (Refer to section 3.7.9.3.)

A VQ texture consists of two types of data, an "index" and a "code book." The relationship between the index and the code book is similar to the relationship between palette texture data and palette data. The index indicates 2 texels (H) × 2 texels (V) of the texture prior to compression, through a code book number. The code book is a grouping of units of data for four texels (64 bits), and usually consists of 256 × 64 bits. The four-texel data of the code book is expanded in a reverse "N" shape, similar to Twiddled format.

The UV size of a texture in the TSP Instruction Word specifies the size of the texture before compression. In other words, the Index is one-half the specified UV size in the horizontal and vertical directions.

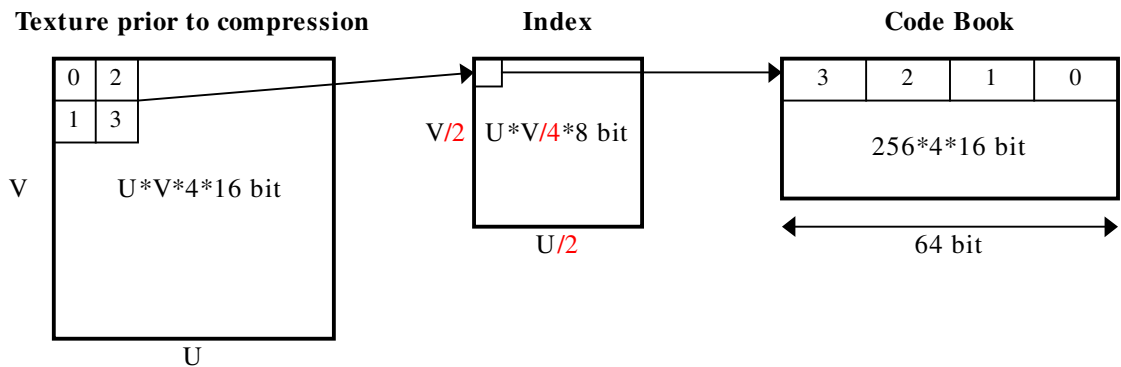


Fig. 3-43

The texture data sizes before and after compression are listed in the table below.

Texture size U × V	Amount of data prior to compression (bytes)	Amount of data after compression (bytes)	Compression ratio (%)	Amount of data in index (bytes)	Amount of data in code book (bytes)
16 × 16	512	2,176	425.00	64	256 × 8
32 × 32	2,048	2,304	112.50	256	256 × 8
64 × 64	8,192	3,072	37.50	1,024	256 × 8
128 × 128	32,768	6,144	18.75	4,096	256 × 8
256 × 256	131,072	18,432	14.06	16,384	256 × 8
512 × 512	524,288	67,584	12.89	65,536	256 × 8
1,024 × 1,024	2,097,152	264,192	12.60	262,144	256 × 8
2,048 × 2,048	8,388,608	1,050,624	12.52	1,048,576	256 × 8

It is predetermined that there are normally 256 code book elements per texture. The number of code book elements indicated for texture sizes of 32 × 32 or smaller in the above table are the values at which data is not compressed, but instead increases in size. Normally, when dealing with textures that are 32 × 32 or smaller, it is necessary to group several into a size of at least 64 × 64 before compressing them.



For VQ textures, the two types of data, the index and the code book, are stored in texture memory. The index data is stored in the same manner as an 8BPP palette texture in Twiddled format, while for the code book 256 data elements, each corresponding to four texels, are stored. In addition, the data must be stored contiguously in texture memory (as shown below), with the code book in the lower addresses.

The texture address that is specified in the Texture Control Word is the starting address of the code book.

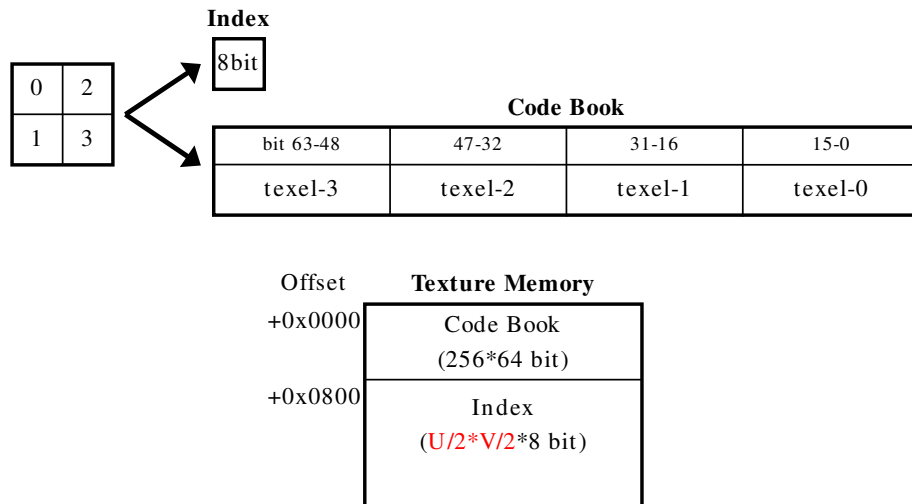


Fig. 3-44

The hardware determines that the texture address specified in the Texture Control Word is the start of the code book data, and uses the address produced by adding 256 x 64-bits to that address as the start of the index data. Therefore, in order to have a code book with less than 256 elements, use an address in the middle of the previous texture data as the texture address that is specified in the Texture Control Word, and then store index data only for the values that correspond to the code book data that was stored.

**Example: Creating a code book with 128 elements:**

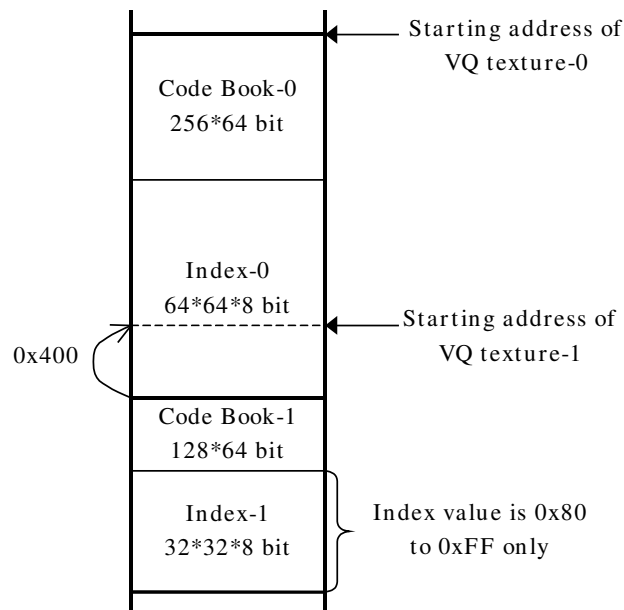


Fig. 3-45

It is possible to use the method for creating a code book with less than 256 elements in order

to compress and store as individual elements data with a texture size of  $32 \times 32$  or less prior to compression. When doing so, the interval between the code book data starting address that is to be used and the corresponding index data starting address must be 2Kbytes (=  $256 \times 64$  bits), so the size of the code book data and the size of the index data must be identical as shown in the table below. (If they are not the same size, memory space will be wasted.)

The compression rate for data with a  $64 \times 64$  texture size before compression can be increased by reducing the code book data in the same manner.

Texture size before compression U x V	Amount of data before compression (byte)	Amount of data after compression (byte)	Compression factor (%)	Index data amount (byte)	Code book data amount (byte)
16 x 16	512	128	25.00	64	8 x 8
32 x 32	2,048	512	25.00	256	32 x 8
64 x 64	8,192	2,048	25.00	1,024	128 x 8

**Example: Storing small VQ textures independently**

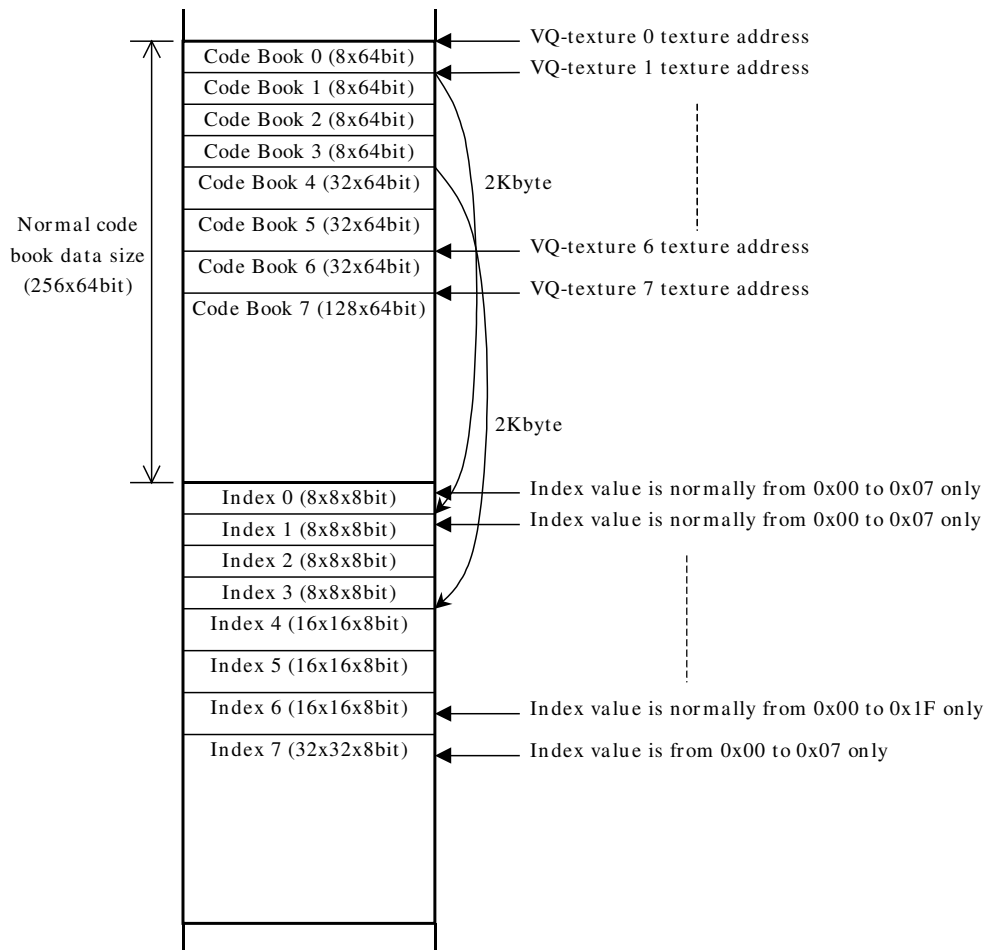


Fig. 3-46

When a VQ texture is stored in this way, it is possible for one index data element to be common to several code book data elements. For example, in the case illustrated above, Index 1 can use Code books 1 through 7, and Index 4 can use Code Books 4 through 7. This is because the hardware regards 256 elements starting from the specified texture address as code book data.

#### § 3.6.2.4 MIPMAP Texture

A MIPMAP texture stores several textures, from  $1 \times 1$  up to a specified size, in texture memory, in order from small to large. However, because YUV textures have one data item per two texels, the  $1 \times 1$  size texture (only) is stored in RGB565 format. MIPMAP textures are only

supported for Twiddle format squares; whether a texture is a MIPMAP texture or not is specified through "MIP Mapped" in the Texture Control Word.

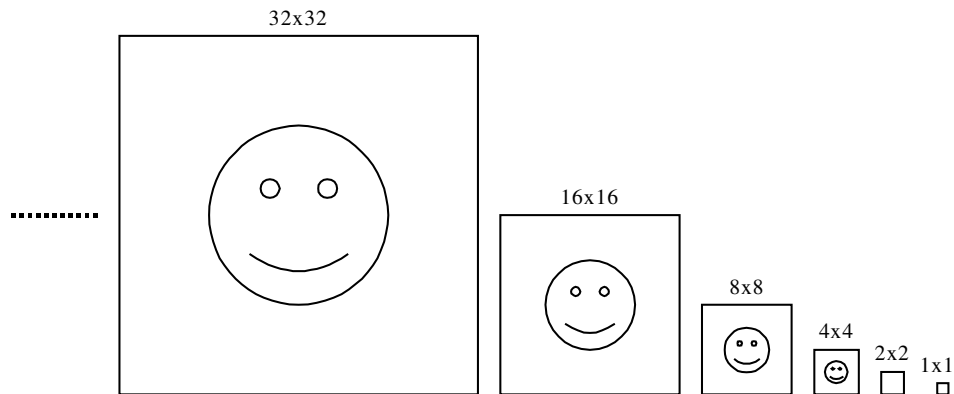


Fig. 3-47

The texture address that is specified in the Texture Control Word is the starting address of the  $1 \times 1$  texture data. In the case of a VQ texture, the starting address of the code book data is specified.

In addition, the data in texel-3 is used for the code book data for the minimum size MIPMAP texture for VQ textures.

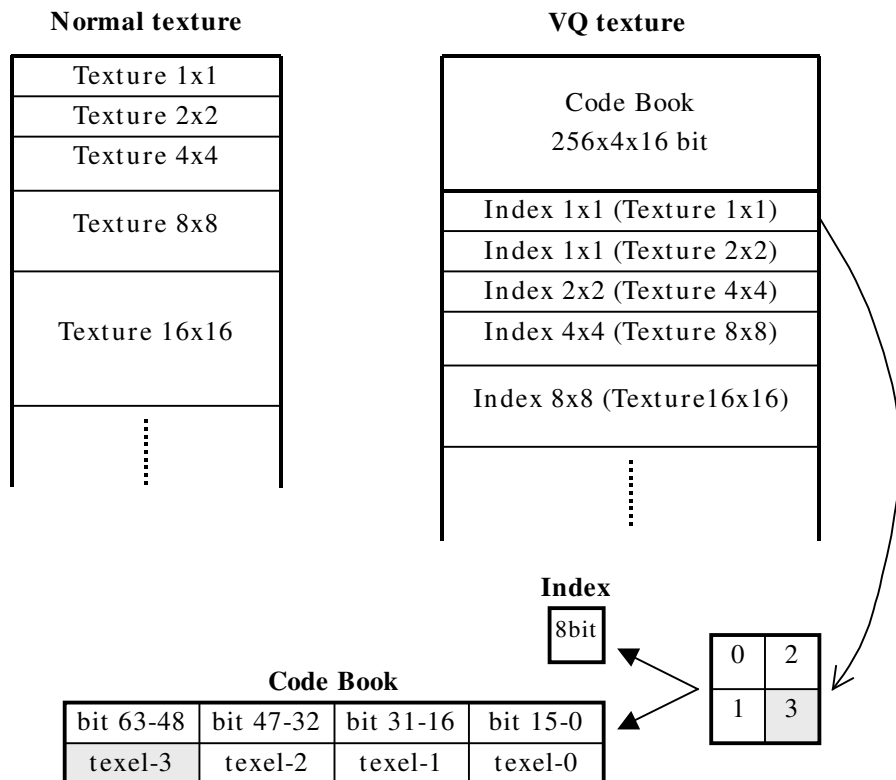


Fig. 3-48

The following tables list the offset values for the starting addresses where texture data is stored for each size of texture. In the case of a VQ texture, however, these values are the offset values for the starting address of the index data.

4BPP palette textures	
Texture size	4-bit offset value for starting address
1x1	0x00003
2x2	0x00004
4x4	0x00008
8x8	0x00018
16x16	0x00058
32x32	0x00158
64x64	0x00558
128x128	0x01558
256x256	0x05558
512x512	0x15558
1024x1024	0x55558

8BPP palette textures	
Texture size	Byte offset value for starting address
1x1	0x00003
2x2	0x00004
4x4	0x00008
8x8	0x00018
16x16	0x00058
32x32	0x00158
64x64	0x00558
128x128	0x01558
256x256	0x05558
512x512	0x15558
1024x1024	0x55558

Non-palette textures	
Texture size	Byte offset value for starting address
1x1	0x00006
2x2	0x00008
4x4	0x00010
8x8	0x00030
16x16	0x000B0
32x32	0x002B0
64x64	0x00AB0
128x128	0x02AB0
256x256	0x0AAB0
512x512	0x2AAB0
1024x1024	0xAAB0

VQ textures	
Texture size	Byte offset value for starting address
1x1	0x00000
2x2	0x00001
4x4	0x00002
8x8	0x00006
16x16	0x00016
32x32	0x00056
64x64	0x00156
128x128	0x00556
256x256	0x01556
512x512	0x05556
1024x1024	0x15556

### § 3.6.3 Color Data Extension

Texture data that is loaded is handled within the CORE as 8-bit values for  $\alpha$ , R, G, and B, respectively.

#### <In Twiddled format>

In the case of a Twiddled format texture, the deficiency in the number of bits is made up by appending the high-order bits (starting from the MSB) of the value at the low-order end of the value so that there are 8 bits present, as shown in the diagram below. An  $\alpha$  value of 0x00 indicates complete transparency, while an  $\alpha$  value of 0x00 indicates complete opacity.

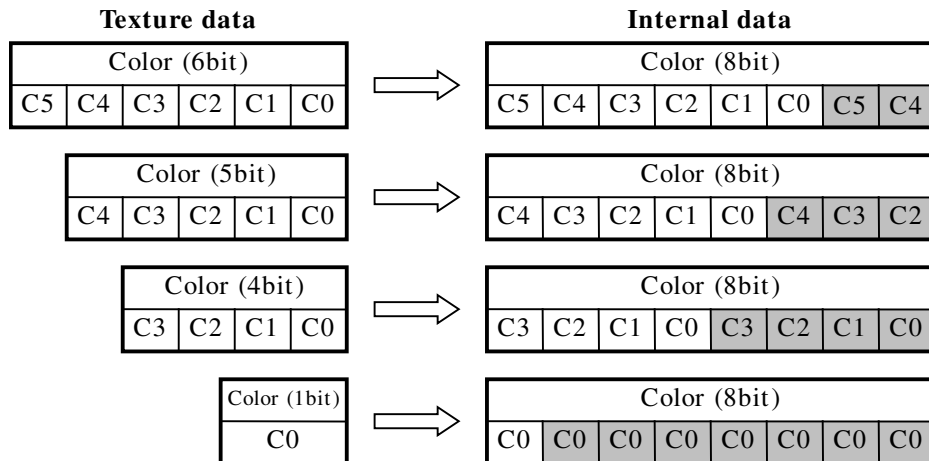


Fig. 3-49

#### <In Non-Twiddled format>

In the case of a Non-Twiddled format texture, zeroes are appended at the low-order end of each value, as shown in the diagram below. However, when there is only one bit, that bit is repeated for the remaining seven bits similar to the case for Twiddled format.

Non-Twiddled format textures are used in order to use as a texture an image that was drawn by the CORE. If the dithering function is used when the image that was drawn is stored in texture memory, the data that is drawn may be the original texture data "+ 1." If this is repeated, the "+ 1" error accumulates in the data for the drawn image, with the possibility that the result will be completely different from the original texture data. Therefore, when color data for a Non-Twiddled format texture is extended, adding zeroes at the low-order end of the data minimizes this color data error.

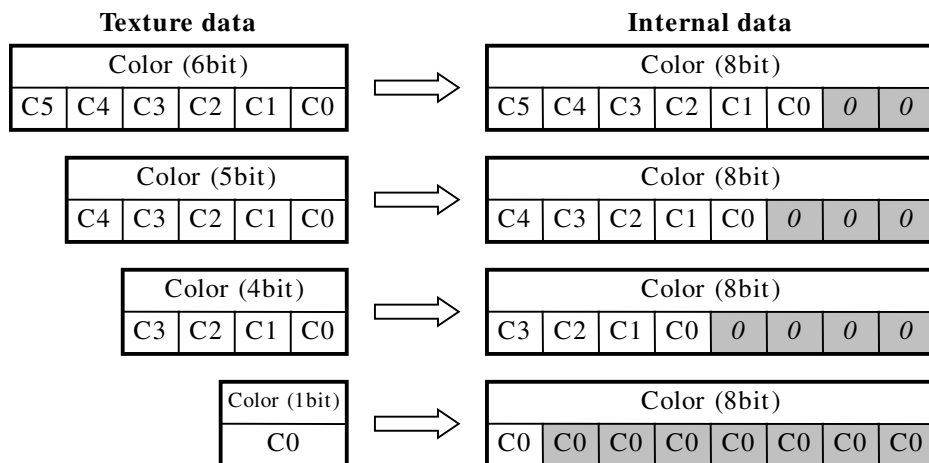


Fig. 3-50

### § 3.6.4 Texture Format Combinations

Texture Control Word					Supplement
bit29-27	bit 31	30	26	25	
Pixel Format	MIP Mapped	VQ Compressed	Scan Order	Stride Select	
Any of RGB1555, RGB565, RGB4444, YUV422, or Bump Map	0	0	0	0	
	0	0	1	0	
	0	0	1	1	
	0	1	0	0	RGB only
	1	0	0	0	Square only
4BPP or 8BPP palette	1	1	0	0	RGB Square only
	0	0	-	-	Twiddled format
	0	1	-	-	Twiddled format
	1	0	-	-	Twiddled format & square
	1	1	-	-	Twiddled format & square

<Notes>

- When "scan order" is "0," "stride select" is ignored.
- When "scan order" is "1," "MIP mapped" is ignored.
- When "scan order" is "1" and "stride select" is "1," the texture U size is specified by the stride value (bits 4 to 0) in the TEXT\_CONTROL register.
- When "MIP mapped" is "1," "V size" in the TSP Instruction Word is ignored and the texture is square.

### § 3.6.5 Efficient Storage in Texture Memory

The storage status of texture data in texture memory has a major impact on drawing performance. Texture memory is divided into hardware "pages" (2048-byte areas), and in order to avoid having a negative effect on drawing performance it is important to store texture data within one page. 2048 bytes of data is equivalent to one 16-bit texture with a size of  $32 \times 32$ .

When storing textures of varying sizes, drawing performance can be kept at a maximum by storing combinations of sizes in a well-planned manner so that no texture spans a page boundary. Even if one texture has 2K or more of data, a deterioration of drawing performance can be prevented by avoiding having texture data span page boundaries as much as possible.

In the case of VQ textures, the code book size is 2K, so it is most efficient to locate the starting address of a code book at a 2K boundary.

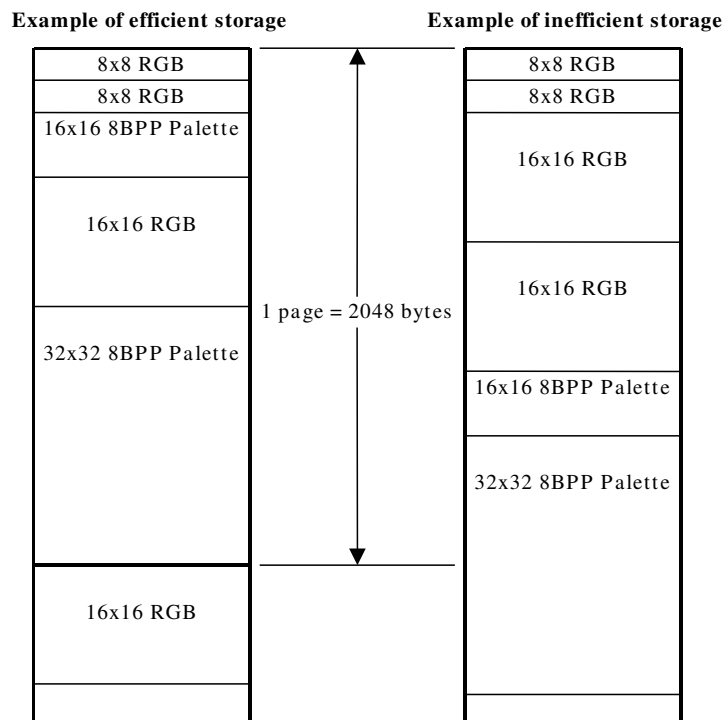


Fig. 3-51

### § 3.7 Display List Details

HOLLY includes two drawing blocks among its rendering blocks: the Tile Accelerator (TA), which assists in display list generation, and the CORE, which draws polygons in individual  $32 \times 32$ -pixel Tiles.

HOLLY polygon drawing display lists include three types of data for the CORE ("Region Array," "Object List," and "ISP/TSP Parameters"), and three types of data for the TA ("Control Parameters," "Global Parameters," and "Vertex Parameters"). The data for the CORE is stored in the texture memory that is connected to the HOLLY. From its three types of data, the TA generates the Object List and ISP/TSP Parameters for the CORE and stores them in texture memory.

Therefore, four types of data are normally required: the Control Parameters, Global Parameters, and Vertex Parameters that are input to the TA, and the Region Array that the CPU stores directly in texture memory. Furthermore, the texture data is stored in texture memory in the format specified by the CORE.

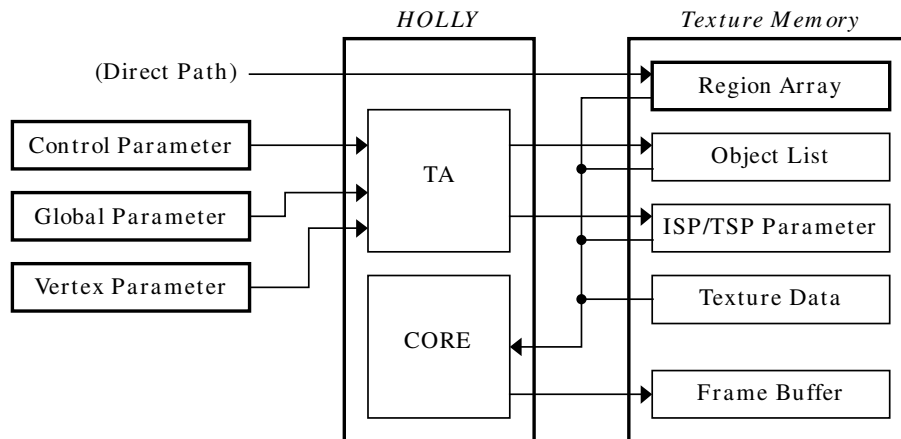


Fig. 3-52

The ISP/TSP Parameters include polygon vertex data and shading data, and the Object List is a collection of the starting address of the data (ISP/TSP Parameters) for the polygons that are included within the same Tile. The Region Array specifies the positions of the Tiles on the screen and the starting addresses in the Object List that correspond to those Tiles.

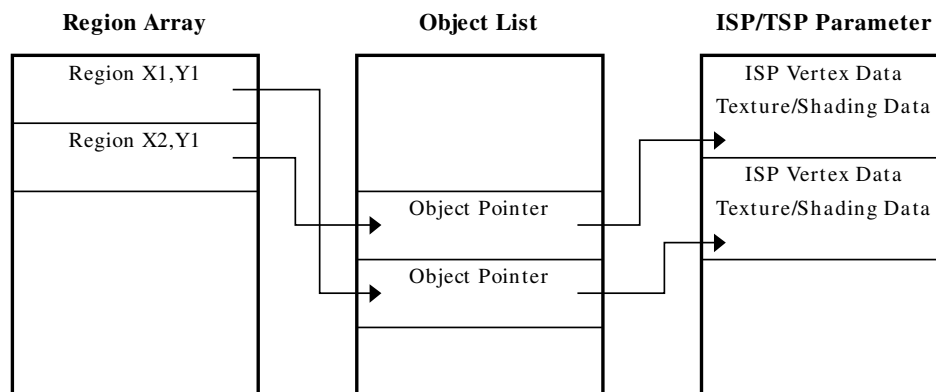


Fig. 3-53

Normally, a display list that is stored in texture memory is used by switching between two buffers: one for the current screen, which is read by the CORE in order to draw, and one for the next screen, which is written by the CPU through the TA. The data for these two buffers can be switched for each frame through the REGION\_BASE register and the PARAM\_BASE register for the CORE and the TA\_OL\_BASE register and the TA\_ISP\_BASE register for the TA.

In addition, because the data values in the Region Array are determined uniquely on the basis of the number of screen Tiles, etc., they only need to be overwritten when the scene that is displayed changes, for example. In other words, once the initial two buffers' worth of Region Array data have been stored, they normally can be left as is until there is a need to overwrite them.



There are three types of data that are stored in the texture memory: the CORE display list, the texture data, and the frame buffer or strip buffer. Some of this data is used on the 64-bit bus, and some is used on the 32-bit bus. Therefore, it is necessary to store the data in texture memory in either the 64-bit access area or the 32-bit access area, whichever is appropriate.

Data being stored	Mapping area where stored
Display list	32-bit access area
Texture data	64-bit access area
Frame buffer or strip buffer	32-bit access area (64-bit access area*)

\* Used when using data that was already drawn as texture data.

There are two types of paths from the CPU bus to texture memory: one through the TA and one through a circuit called the PVR I/F. The paths through the TA permit only 32-byte burst writes through the SH4 store queue or through DMA; reading is not possible. Although reading and writing are both possible with the path through the PVR interface, this path is slower than the paths through the TA. Therefore, data transfers to texture memory are normally performed on the paths through the TA. There are four paths through the TA:

- (1) A path that generates the Object List and ISP/TSP Parameters, and stores them in the 32-bit access area
- (2) A path that converts YUV data into YUV-422 data and stores the result in the 64-bit access area
- (3) A direct path to the 64-bit access area
- (4) A direct path to the 32-bit access area

Path 1 is used to generate the Object List and ISP/TSP Parameters from the three types of TA input parameters (Control Parameters, Global Parameters, and Vertex Parameters), and then store the results in texture memory. Polygon data is normally transferred on this path. Note that the CPU creates the Region Array, and transfers it to texture memory through path 4. Path 2 is used to convert YUV data that was input in macro block units (16 pixels × 16 pixels) into YUV-422 texture data for the CORE, and then store the results in texture memory. MPEG data is transferred on this path. Other texture data is transferred to texture memory through path 3. This path specification is made through the address of the transfer destination. For details, refer to "2.6 Data Transfers."

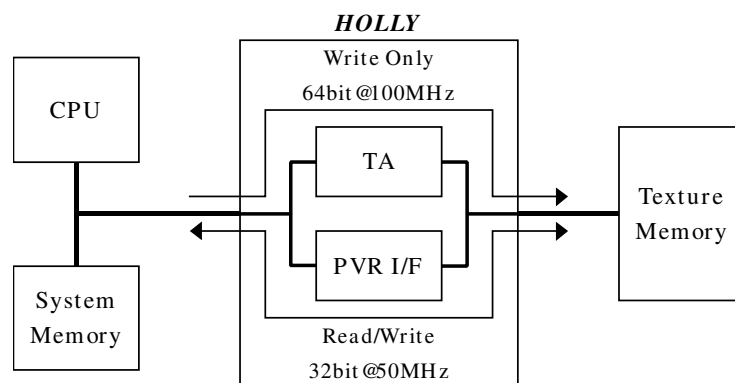


Fig. 3-54

Three types of data are input to the TA:

- (1) Polygon data: TA input parameters for the display list
- (2) YUV data: YUV data for individual macro blocks
- (3) Direct data: Data that is written directly into texture memory (64 bits or 32 bits)

Distinctions are made between each type of data through the address (mapping area) indicated when the data is input to the TA. Regarding the input order, polygon data and other data (YUV data and direct data) can be combined freely, in units of 32 bytes. However, if YUV data and direct data are to be input together, the direct data must not be input until a macro block of YUV data (384 bytes of YUV420 data or 512 bytes of YUV422 data) has been input.

**Example**

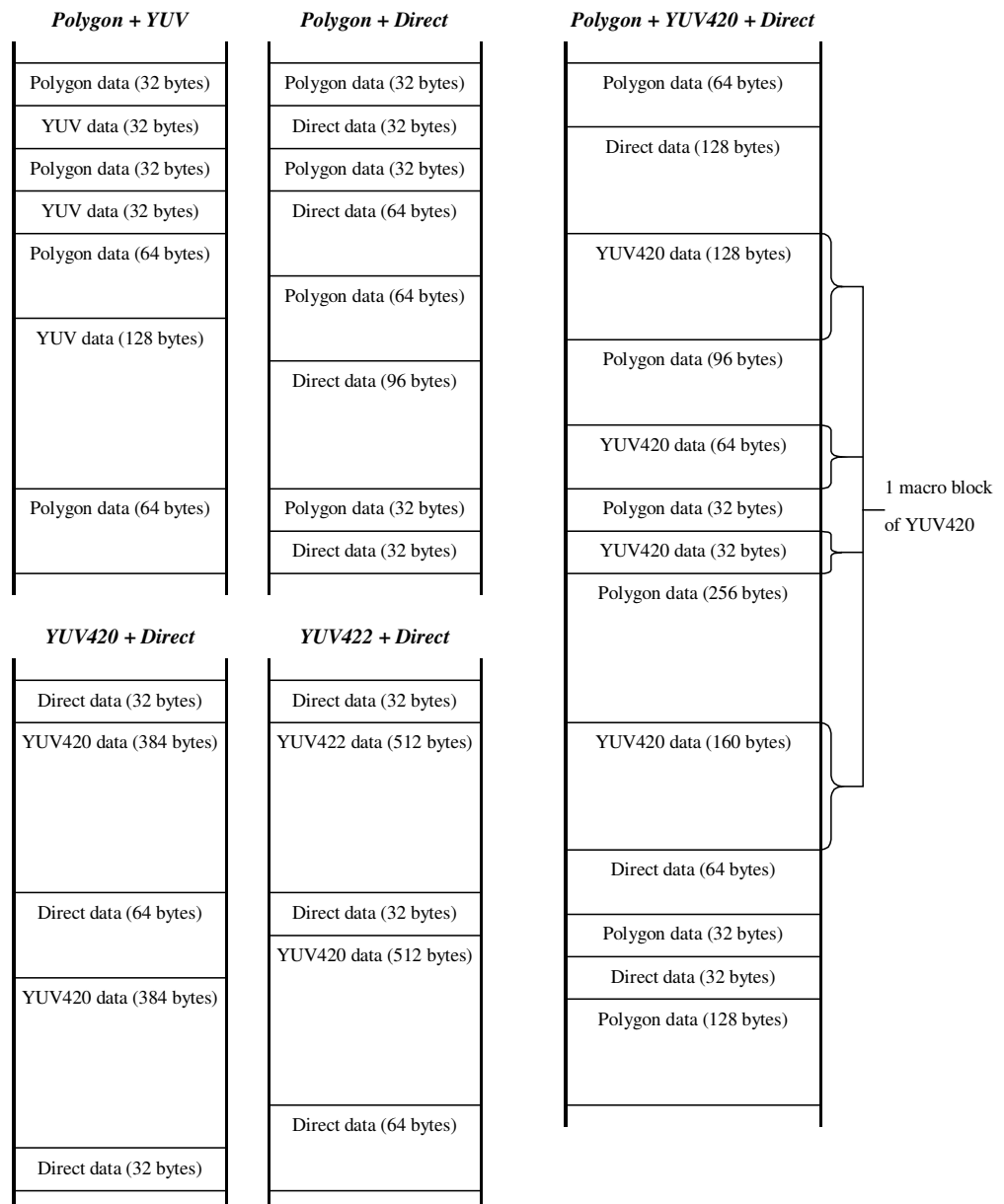


Fig. 3-55

### § 3.7.1 Polygon List Input

HOLLY utilizes the following five polygon lists. "Punch Through" is an Opaque polygon that uses texture data that only has texels with an alpha value of either 0.0 (transparent) or 1.0 (opaque)).

- |                                  |  |
|----------------------------------|--|
| (1) Opaque:                      | Opaque polygon   |
| (2) Opaque Modifier Volume:      | Opaque polygon & Punch Through Modifier Volume         |
| (3) Translucent:                 | Translucent polygon                                    |
| (4) Translucent Modifier Volume: | Translucent polygon Modifier Volume                    |
| (5) Punch Through*:              | Punch Through polygon ( <del>*added for HOLLY2</del> ) |

When inputting the polygon data to the TA, it is necessary to first perform a "TA reset" or a "list initialization," and then input the polygons grouped, by type. Only those lists of the necessary types need to be input; it is not necessary to input polygon lists for all five (~~five, in the case of HOLLY2~~) types. The order in which each list is input does not matter. However, each list can only be input once; a list of the same type of polygons cannot be input twice or more.

**Polygon list input example** (~~\*Punch Through input is for HOLLY2~~)

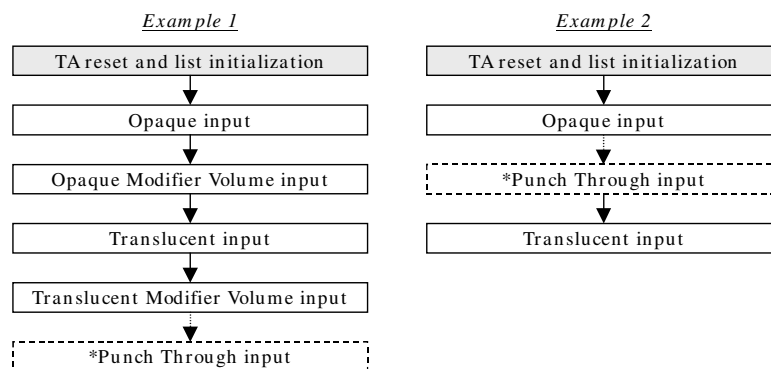
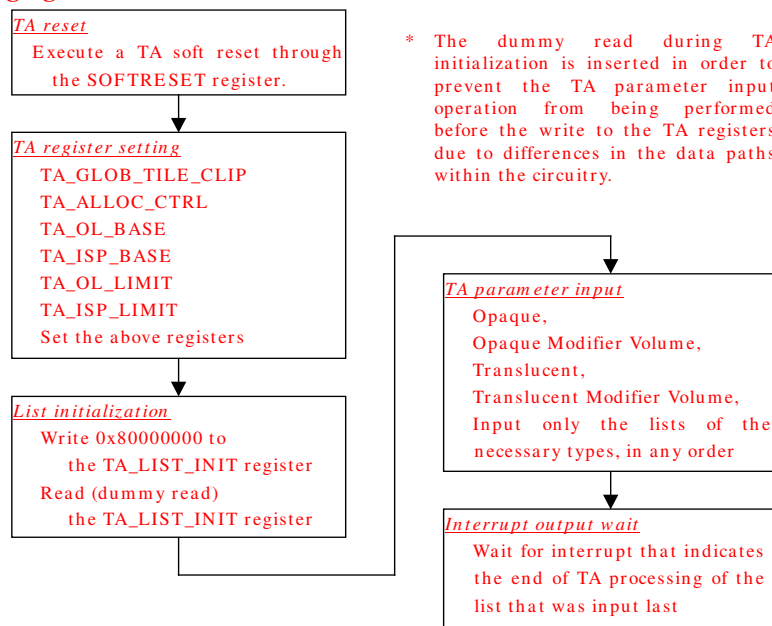


Fig. 3-56

~~The flow for inputting polygon lists to the TA (and parameters for the TA) is shown below (HOLLY1 only):~~

~~Following figure is deleted.~~



~~Fig. 3-57~~

~~(The following description, and the description from section 3.7.1.1 to 3.7.1.4 apply to HOLLY2.)~~

~~HOLLY2~~ HOLLY supports "multipass operation," in which polygon lists are input several times in succession. In multipass operation, "list continuation processing" is inserted after a list is input, and then processing continues with the input of the next list. This allows a list of the same type of polygon to be divided into several lists and input as more than one list of the same type.

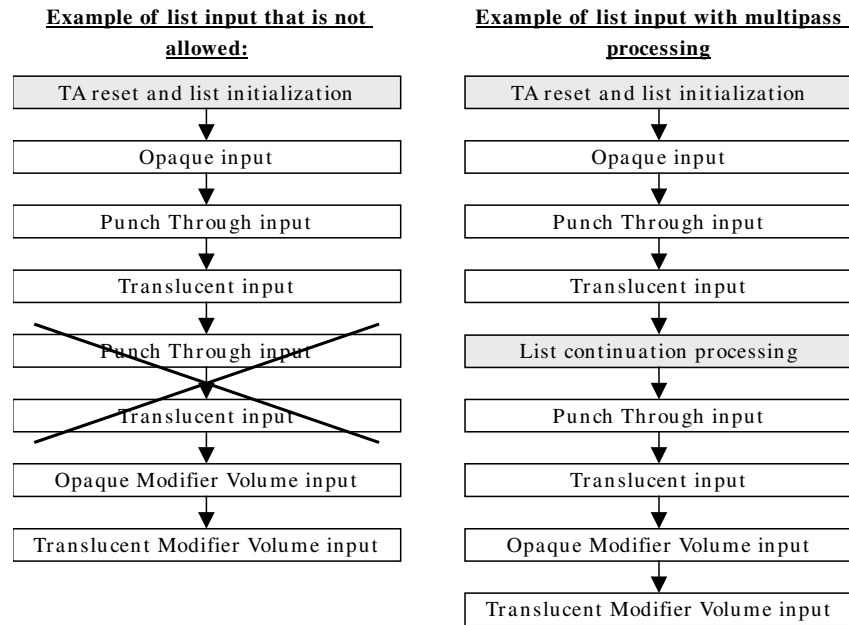


Fig. 3-58

### § 3.7.1.1 TA Parameter Input Flow

The flow of polygon list input to the TA (TA parameter input) is shown below.

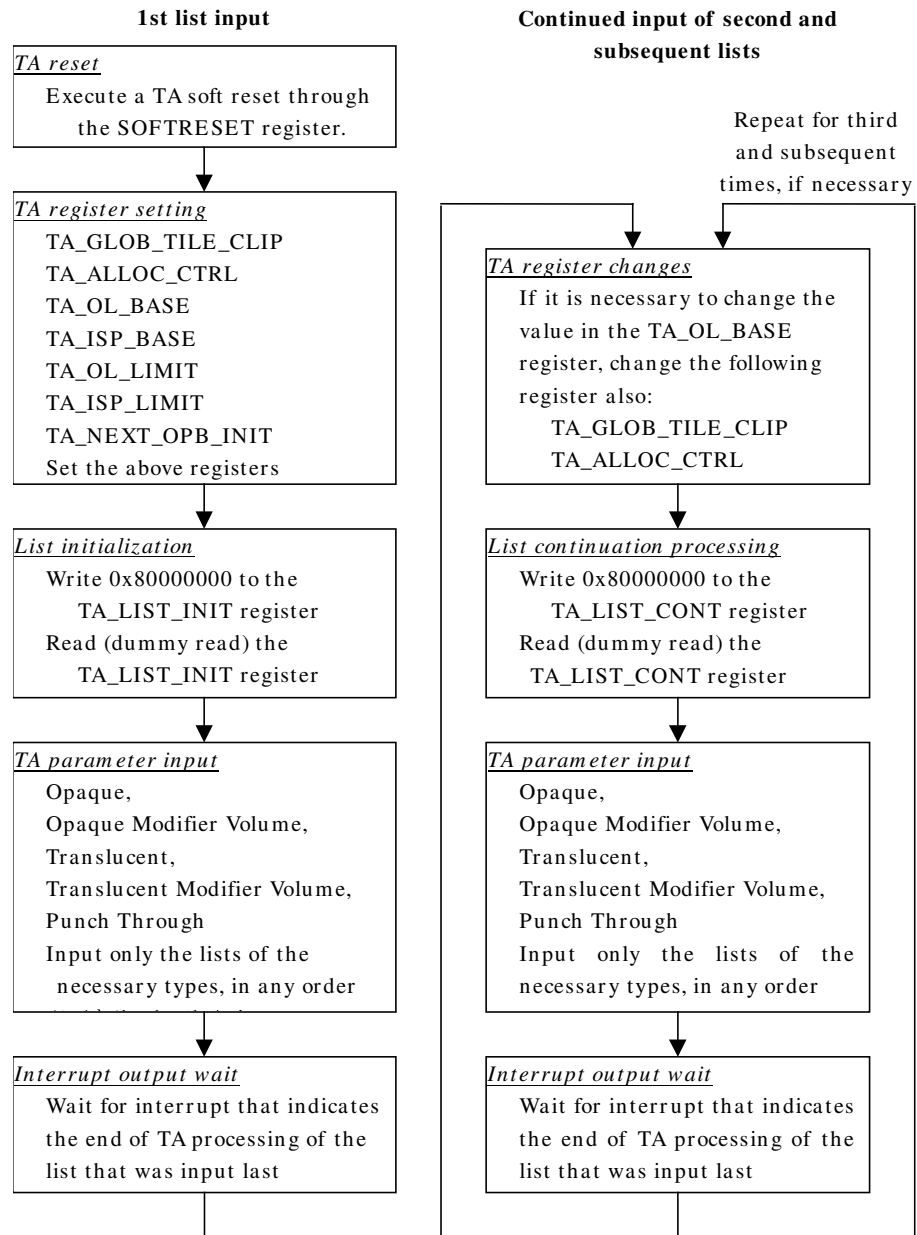


Fig. 3-59

The total OPB size for all of the lists that will be input to the TA must be taken into consideration when determining the value that is to be set in the TA\_NEXT\_OPB\_INIT register before list initialization. In addition, before performing the continuation processing for the second and subsequent lists, it is necessary to change the value in the TA\_OL\_BASE register to the OPB starting address for that list. If necessary, also change the value in the TA\_GLOB\_TILE\_CLIP register and in the TA\_ALLOC\_CTRL register.

The dummy read during TA initialization and list continuation processing is inserted in order to prevent the TA parameter input operation from being performed before the write to the TA registers due to differences in the data paths within the circuitry. Therefore, the dummy read does not have to be performed specifically on the register indicated above; any TA register is fine.

### § 3.7.1.2 TA Register Settings for List Input

When list initialization is performed via the TA\_LIST\_INIT register, the TA sets up the area from the address that is specified in the TA\_OL\_BASE register to the address that is specified in the TA\_NEXT\_OPB\_INIT register as the OPB initial area in texture memory. The OPB initial area size that is needed for one TA input list is the product of the total OPB size for all lists specified by the TA\_ALLOC\_CTRL register before the input of that list, multiplied by the number of Tiles in the Global Tile Clip area that is specified by the TA\_GLOB\_TILE\_CLIP register. The amount of memory that should be reserved in texture memory as the OPB initial area is the sum of the OPB initial area size for the one list added together for each of the polygon lists that are input to the TA.

$$\begin{aligned} (\text{OPB initial area size for one list}) = & \{(\text{Opaque list OPB size}) \\ & + (\text{Opaque Modifier Volume list OPB size}) \\ & + (\text{Translucent List OPB size}) \\ & + (\text{Translucent Modifier Volume list OPB size}) \\ & + (\text{Punch Through list OPB size}) \\ & \times (\text{Number of Tiles in Global Tile Clip area}) \times 4 \text{ bytes} \\ & \times 4 \text{ byte} \end{aligned}$$

$$\begin{aligned} (\text{OPB initial area size for list initialization}) = & (\text{OPB initial area size for first list input}) \\ & + (\text{OPB initial area size for second list input}) \\ & + (\text{OPB initial area size for third list input}) \\ & + \dots \dots \dots + \dots \dots \dots + \dots \dots \dots \end{aligned}$$

The value in the TA\_NEXT\_OPB\_INIT register, which should be set prior to list initialization, is the sum starting address value of the Object List that is stored in texture memory and the OPB initial area size.

$$\begin{aligned} (\text{TA\_NEXT\_OPB\_INIT register value}) = & (\text{TA\_OL\_BASE register value at list initialization}) \\ & + (\text{OPB initial area size at list initialization}) \end{aligned}$$

In addition, it is necessary to change the value in the TA\_OL\_BASE register before the list continuation processing that is performed through the TA\_LIST\_CONT register; the value is the sum of the Object List starting address and the total of the previously input OPB initial area sizes.

$$\begin{aligned} (\text{TA\_OL\_BASE register value prior to list continuation processing}) \\ = & (\text{Value in the TA\_OL\_BASE register at list initialization}) \\ & + (\text{total of the previously input OPB initial area sizes}) \end{aligned}$$

When the values in the TA\_GLOB\_TILE\_CLIP register and the TA\_ALLOC\_CTRL register are changed for each list, be certain to change them prior to the list continuation processing that is performed through the TA\_LIST\_CONT register.

An example of the processing for inputting TA parameters twice described below.

**Example: When list input to the TA is done twice**

**pass 1: First input**

1) *TA register settings*

TA\_GLOB\_TILE\_CLIP = 0x00010001  
TA\_ALLOC\_CTRL = 0x00021212  
TA\_OL\_BASE = 0x00200000  
TA\_ISP\_BASE = 0x00280000  
TA\_OL\_LIMIT = 0x0027FFE0  
TA\_ISP\_BASE = 0x00300000  
TA\_NEXT\_OPB\_INIT = 0x00200600

2) *List initialization*

TA\_LIST\_INIT = 0x80000000

3) *TA parameter input*

OP → PT → TR → OMV → TMV

4) *Interrupt output wait*

Wait for interrupt at end of TMV processing

**pass 2: Second input**

1) *TA register setting changes*

TA\_ALLOC\_CTRL = 0x00020002  
TA\_OL\_BASE = 0x00200400

2) *List continuation processing*

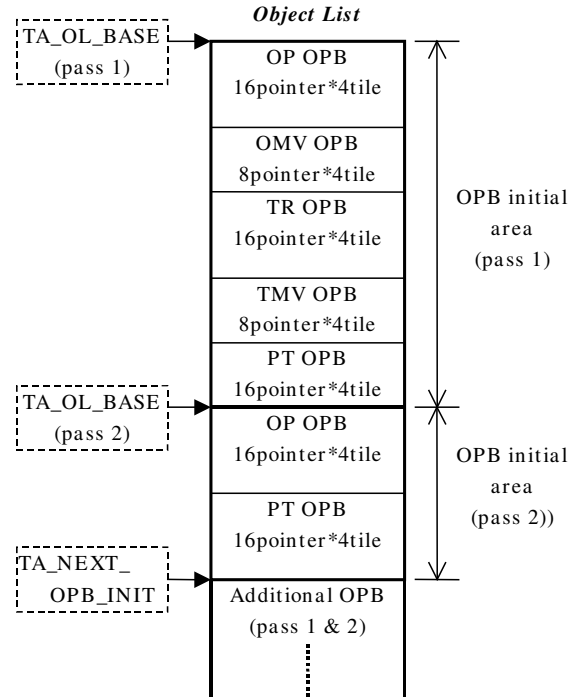
TA\_LIST\_CONT = 0x80000000

3) *TA parameter input*

OP → PT

4) *Interrupt output wait*

Wait for interrupt at end of PT processing



OP = Opaque, TR = Translucent, PT = Punch Through,  
OMV = Opaque Modifier Volume,  
TMV = Translucent Modifier Volume

Fig. 3-60

In this example, the settings for each register are determined as follows:

- (1) Determine the starting address and limit address for storing the Object List and the ISP/TSP Parameters.

TA\_OL\_BASE = 0x00200000, TA\_OL\_LIMIT = 0x0027FFE0,  
TA\_ISP\_BASE = 0x00280000, TA\_ISP\_LIMIT = 0x00300000

- (2) Determine the drawing enabled areas for the polygon lists that will be input first and second.

In both cases: 2 Tiles × 2 Tiles → TA\_GLOB\_TILE\_CLIP = 0x00010001

- (3) Determine the list types and OPB sizes for the polygon lists that will be input first and second.

First list: Input all five types; OPB sizes: OP = 16, OMV = 8, TR = 16, TMV = 8, and PT = 16

→ first TA\_ALLOC\_CTRL = 0x00021212

Second list: Input OP and PT; OPB sizes: OP = 16, PT = 16

→ second TA\_ALLOC\_CTRL = 0x00020002

- (4) Based on the drawing enabled area and the input list OPB size, calculate the total OPB initial area for the two lists.

First list: (16 + 8 + 16 + 8 + 16) × 4 Tiles × 4 bytes = 1024 bytes = 0x400

→ second TA\_OL\_BASE = 0x00200400

Second list: (16 + 16) × 4 Tiles × 4 bytes = 512 bytes

Total: 1024 + 512 = 1536 bytes = 0x600 → TA\_NEXT\_OPB\_INIT = 0x00200600

### § 3.7.1.3 Region Array Data Storage

The TA creates Object Lists and ISP/TSP Parameters for the same number of groups as the

number of times that a list was input, basing the parameters on a polygon list that was input in several passes using the multipass function. The same quantity of Region Array data in this instance is as the number of lists that were input for one Tile is required.

**Relationship between the TA input polygon lists and the CORE display list**

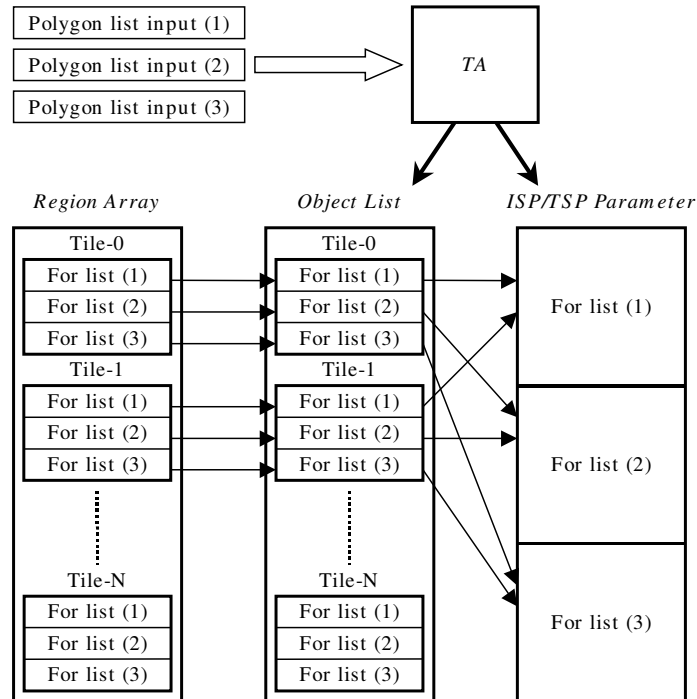


Fig. 3-61

When drawing using a CORE display list that was created from a polygon list that was input in several pieces to the TA, drawing must continue in the same Tile. Therefore, it is necessary to store Region Array data for the same Tile in consecutive areas in texture memory. The Z Clear bit and Flush Accumulate bit within the Region Array data must be controlled according to whether the data is for the first drawing or last drawing to the same Tile.

**Region array when only one list is input**

For drawing in tile 0 (Z Clear = 0, Flush Accumulate = 0)
For drawing in tile 1 (Z Clear = 0, Flush Accumulate = 0)
For drawing in tile 2 (Z Clear = 0, Flush Accumulate = 0)
⋮

**Region array in multi-pass processing**

First for drawing in tile 0 (Z Clear = 0, Flush Accumulate = 1)
Second for drawing in tile 0 (Z Clear = 1, Flush Accumulate = 1)
Third for drawing in tile 0 (Z Clear = 1, Flush Accumulate = 1)
⋮
Last for drawing in tile 0 (Z Clear = 1, Flush Accumulate = 0)
First for drawing in tile 1 (Z Clear = 0, Flush Accumulate = 1)
⋮

Fig. 3-62



#### § 3.7.1.4 Object List Starting Address for Each List

The TA stores the first OPB for each Tile from the polygon lists that were input in texture memory according to consistent rules. If a list was input in several pieces through multipass processing, a new OPB is stored each time.

The Object List starting addresses (List Pointers) for the five types of lists that are specified in the Region Array data for each Tile corresponding to each list input can be determined through the following calculations:

(Opaque List Pointer for Nth list input)  
= OL\_base + OP\_size × T\_num × 0x4

(Opaque Modifier Volume List Pointer for Nth list input)  
= OL\_base + (OP\_size × GC\_Tile + OM\_size × T\_num) × 0x4

(Translucent List Pointer for Nth list input)  
= OL\_base + [(OP\_size + OM\_size) × GC\_Tile + TR\_size × T\_num] × 0x4

(Translucent Modifier Volume List Pointer for Nth list input)  
= OL\_base + [(OP\_size + OM\_size + TR\_size) × GC\_Tile + TM\_size × T\_num] × 0x4

(Punch through List Pointer for Nth list input)  
= OL\_base + [(OP\_size + OM\_size + TR\_size + TM\_size) × GC\_Tile + PT\_size × T\_num] × 0x4

OL\_base: Object list starting address for Nth list input (TA\_OL\_BASE register value)

GC\_Tile: Total number of Tiles in Global Tile Clip area for Nth list input

OP\_size: Opaque list OPB size for Nth list input

OM\_size: Opaque Modifier Volume list OPB size for Nth list input

TR\_size: Translucent List OPB size for Nth list input

TM\_size: Translucent Modifier Volume list OPB size for Nth list input

PT\_size: Punch Through list OPB size for Nth list input

T\_num: Number of Tiles in Global Tile Clip area prior to the Tile for which the address is being derived

*Example: GC\_tile and T\_num values*

0	1	2	3	4			
5	6	7	8	9			
10	11	12	13	14			
15	16	17	18	19			

The gray portion indicates tiles in the global tile clip area

In this case, GC\_tile = 20, and the number indicated in each tile is T\_num.

Fig. 3-63

Storage of OPBs for each Tile by TA is done in the horizontal direction, starting from the upper left corner of the screen and continuing until the Tile on the right edge of the screen, followed by the Tiles one row down, starting from the left end again, and so on.

An example of Region Array data for an Object List that was created when TA parameters were input three times is shown below. (Refer to section 3.7.7.)

**Example: List input to the TA three times**

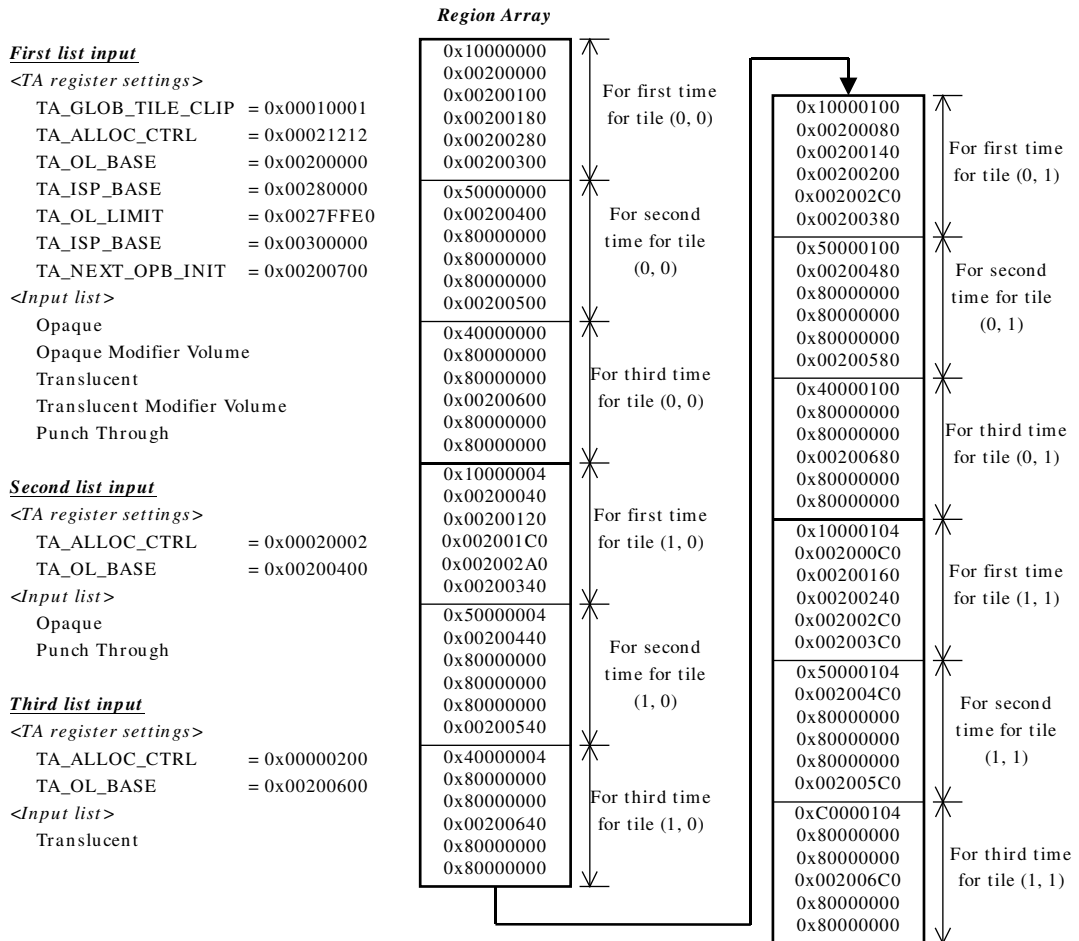


Fig. 3-64

For example, the calculations for the List Pointers that specify Region Array for the first time for Tile (1, 0) are as follows:

(Opaque List Pointer)

$$= 0x00200000 + 16 \times 2 \times 0x4 = 0x00200080$$

(Opaque Modifier Volume List Pointer)

$$= 0x00200000 + (16 \times 4 + 8 \times 2) \times 0x4 = 0x00200140$$

(Translucent List Pointer)

$$= 0x00200000 + \{ (16 + 8) \times 4 + 16 \times 2 \} \times 0x4 = 0x00200200$$

(Translucent Modifier Volume List Pointer)

$$= 0x00200000 + \{ (16 + 8 + 16) \times 4 + 8 \times 2 \} \times 0x4 = 0x002002C0$$

(Punch Through List Pointer)

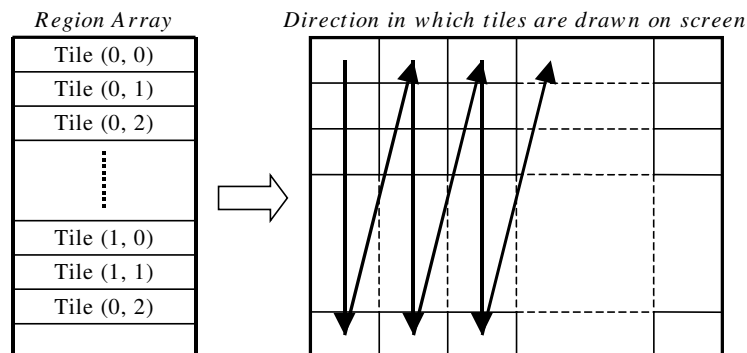
$$= 0x00200000 + \{ (16 + 8 + 16 + 8) \times 4 + 16 \times 2 \} \times 0x4 = 0x00200380$$

### § 3.7.2 Tile Arrangement

The Region Array is the first data that the CORE reads when drawing Tiles; the Region Array data indicates the positions of the Tiles in the screen. Because the CORE draws the Tiles in the order indicated in the Region Array data that is stored in texture memory, the user can freely specify the direction in which drawing proceeds within a screen. This order in which the Region Array data is stored in texture memory is called the "Tile arrangement."

Because the CPU creates the Region Array directly and stores it in texture memory, the Tile arrangement can be set freely, but the two methods that are used normally are "vertical arrangement" and "horizontal arrangement."

#### ◆ Vertical arrangement



#### ◆ Horizontal arrangement

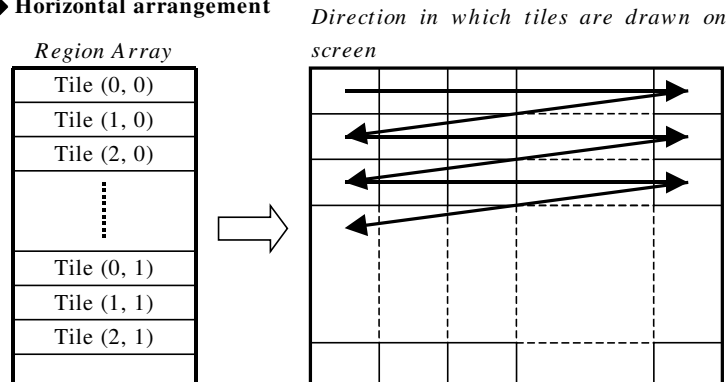


Fig. 3-65

The starting address of the Object List data is also specified in the Region Array data. Because the TA generates the Object List automatically, the address must be set accordingly. (Refer to section 3.7.3.4.) Because storage of Object List data by the TA in texture memory is done in the horizontal direction, address calculation requires special care when stacking Tiles vertically.

The drawing performance of the CORE can vary slightly, depending on the Tile arrangement. For reasons concerning the CORE's internal parameter cache hit rate, arranging the Tiles in the Y direction offers slightly better drawing performance than the X direction.

Note also that the Tile arrangement may be restricted, depending on the functions that are being used. If Y-direction filtering is to be performed, it will not be performed correctly if the Tiles are not arranged in the Y direction. (Refer to section 3.4.10.) When using the strip buffer, the Tiles must be arranged in the X direction. (Refer to section 3.4.13.)

### § 3.7.3 Tile Accelerator

The TA performs the following processing in order to generate the CORE display list (Object List and ISP/TSP Parameters).

- Partitioning infinite strip polygon data
- Dividing polygons into Tiles
- Clipping Tiles
- Generating the Object List
- Generating the ISP/TSP Parameters

#### § 3.7.3.1 Strip Partitioning

Polygon data (Triangle polygons) that is input to the TA is compatible with infinite strips, but the CORE only supports strips with a maximum number of six triangles. Therefore, the TA partitions infinite strip polygon data into strips of 1 to 6 triangles, and then stores the data in texture memory. The number of triangles (the strip number) in the partitioned strips can be specified within the polygon data that is input. In addition, the "end of strip" bit must be set in the last vertex data in the strip. If the last partitioned strip has fewer triangles than the number specified, then when the vertex data at the end of the strip is input, the TA generates the polygon data with that strip number.

The TA does not support strips of Spites (Quad polygons) or Modifier Volumes (Triangle polygons).

**Example: When strip number = 4**

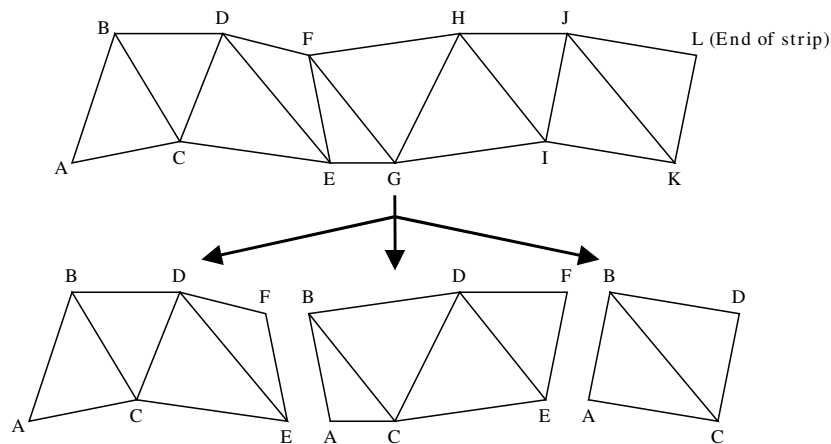


Fig. 3-66

### § 3.7.3.2 Tile Division

The TA supports a drawing screen of up to 1280 pixels (H) × 480 pixels (V). Because the Tile size is fixed at 32 pixels × 32 pixels, the number of Tiles on the screen is 40 Tiles (H) × 15 Tiles (V) (for a total of 600 Tiles).

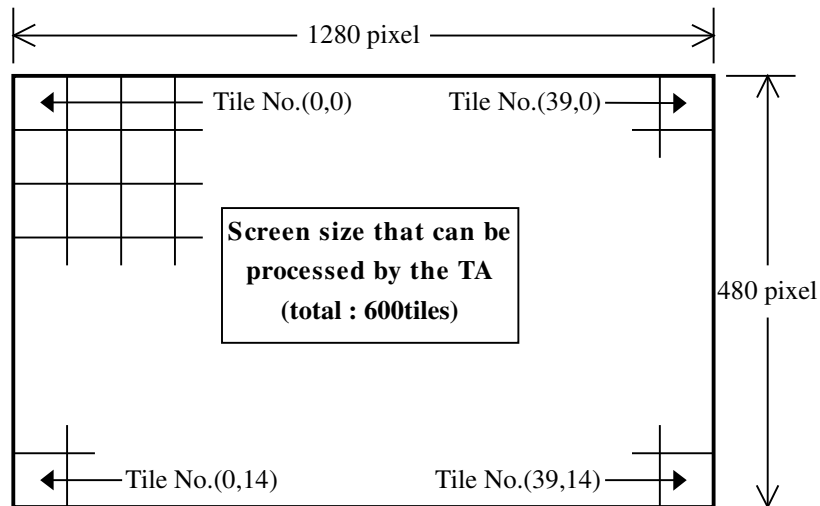
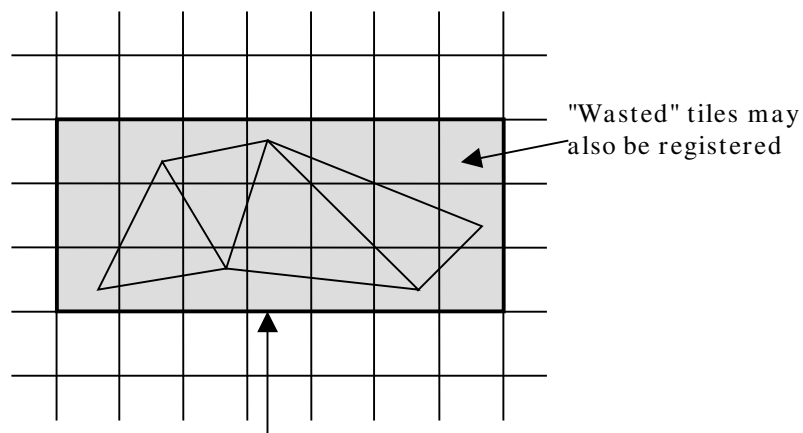


Fig. 3-67

The CORE requires an Object List that shows the starting address of the polygon data (the ISP/TSP Parameters) that is included in each Tile. In order to generate this Object List, the TA divides the polygons that are input into Tiles. This processing converts the floating-point X and Y vertex coordinates that were input into integer values by truncating the decimal portion, then determines the rectangle area (which consists of all of the individual Tiles that enclose the entire polygon) on the basis of the minimum and maximum X and Y coordinates. All of the Tiles within the area are deemed to contain part of the polygon.

After being input to the TA and then partitioned into strips, the polygon data is registered in the Object List for the Tiles inside the bounding box. Therefore, the polygon is registered even in the Object List for Tiles which do not actually contain part of the polygon, so the amount of Object List data may be larger than what might be expected. Note especially that in the case of a long, thin polygon that is displayed on an angle will result in most Tiles being such "wasted" Tiles.



Area that encompasses the entire polygon (bounding box)

Fig. 3-68

### § 3.7.3.3 Tile Clipping

When dividing a polygon into Tiles, it is possible to specify the clipping area in units of Tiles. Each polygon is then registered in Object Lists only for Tiles within the valid drawing area. There are two types of clipping areas that can be specified: a Global Tile Clipping area (that is valid for all polygons) and a User Tile Clipping area (that can be specified for individual polygons). For each type, the rectangular area is specified through the numbers of the Tiles that occupy the upper left and lower right corners. The Global Tile Clipping area values are specified through the TA\_GLOB\_TILE\_CLIP register, and the User Tile Clipping area values are specified through the User Tile Clipping parameter (a Control Parameter). The inside of the Global Tile Clipping area is always the valid area, while for the User Tile Clipping area it is possible to select either "off," "inside valid," or "outside valid." The valid drawing area is determined by ANDing these two areas together (i.e., by taking the logical product).

The Global Tile Clipping and User Tile Clipping areas are both used for Modifier Volume polygons.

#### Global Tile Clip & User Tile Clip

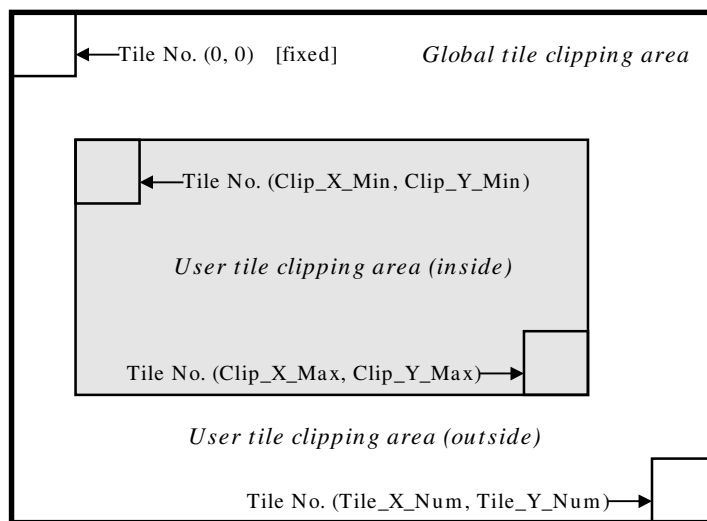
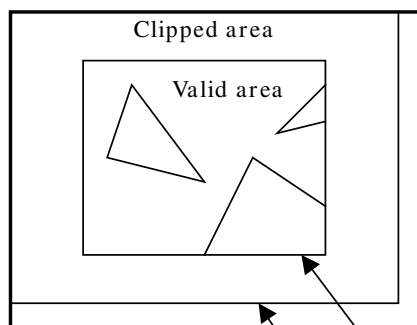


Fig. 3-69

#### Parameter Control Word

**User\_Clip=10**

User Tile Clipping inside enable



#### Parameter Control Word

**User\_Clip=11**

User Tile Clipping outside enable

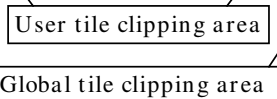
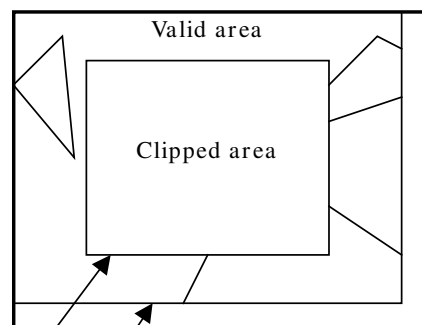


Fig. 3-70

### § 3.7.3.4 Object List Generation

Polygons that are input to the TA are registered in an Object List that corresponds to the Tiles

that are located in the bounding box as a result of Tile division. The TA has a built-in 600-Tile buffer, called the "object List Pointer buffer," that is used to retain individual Tile data that is necessary in order to generate the Object List. The CPU can read this information by using the TA\_OL\_POINTERS register.

The Object List consists of a data block that ranges in size from 8, 16,  $32 \times 32$  bits, called the "Object Pointer Block (OPB);" the size of the OPB can be specified for each type of list through the TA\_ALLOC\_CTRL register. The OPB that the TA stores in texture memory corresponds to the Tiles in the Global Tile Clipping area; no Object List is stored for Tiles outside of the area. Note that no parameters are input to the TA for lists of a type for which "no list" was specified for the OPB size.

Once the list that is currently being input is ended by inputting the "end of list" Control Parameter, the TA automatically stores the "end of list" data (Refer to "Object Pointer Block Link Data" in section 3.7.8) in the Object Pointer Block for each Tile.

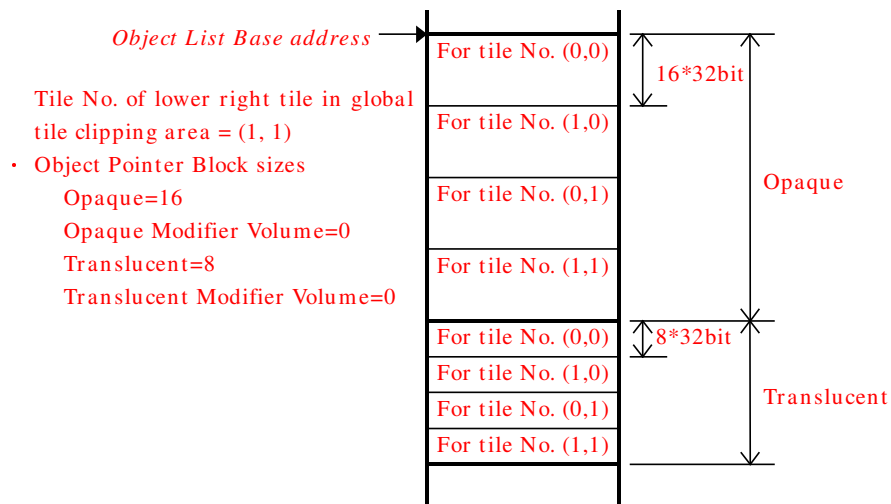
### § 3.7.3.4.1 List Initialization Processing and List Continuation Processing

~~The description in this section is separate for HOLLY1 and HOLLY2, because the processing is different.~~

~~In HOLLY1, if list initialization is performed through the TA\_LIST\_INIT register, the TA allocates an Object List data area in texture memory, starting from the address that is specified in the TA\_OL\_BASE register. The amount of memory that is allocated is twice the number of Tiles in the Global Tile Clipping area specified in the TA\_GLOB\_TILE\_CLIP register for the OPB size that was specified in the TA\_ALLOC\_CTRL register for each list type. In addition, the order of the Tiles stored in memory is (1) from left to right and (2) from top to bottom. The order of the lists is (1) opaque, (2) opaque Modifier Volume, (3) translucent, and (4) translucent Modifier Volume.~~

~~The amount of memory allocated for the Object List upon initialization =  
 (OPB size for Opaque Lists)  
 (OPB size for opaque Modifier Volume lists)  
 (OPB size for translucent lists)  
 (OPB size for translucent Modifier Volume lists)  
 × (number of Tiles in the Global Tile Clipping area)  
 × 32 bits~~

#### Example: Object List upon initialization



~~Fig. 3-74~~

~~In this way, the TA stores the initial Object Pointer Block for each Tile in texture memory according to fixed rules. Therefore, the starting addresses of the Object Lists for the four lists that must be set in the Region Array can be derived according to the following calculations:~~

~~(Object list starting address for Opaque List)  
 = OL\_base + OP\_size × TP\_num × 4h~~

~~(Object list starting address for opaque Modifier Volume list)~~

$$\text{OL\_base} + \text{OP\_size} \times \text{GC\_Tile} + \text{OM\_size} \times \text{T\_num} \times 4h$$

~~(Object list starting address for translucent list)~~

$$\text{OL\_base} + \{ (\text{OP\_size} + \text{OM\_size}) \times \text{GC\_Tile} + \text{TP\_size} \times \text{T\_num} \} \times 4h$$

~~(Object list starting address for translucent Modifier Volume list)~~

$$\text{OL\_base} + \{ (\text{OP\_size} + \text{OM\_size} + \text{TP\_size}) \times \text{GC\_Tile} + \text{TP\_size} \times \text{T\_num} \} \times 4h$$

~~OL\_base: Object list base address~~

~~OL\_base: Object list base address~~

~~GC\_Tile: Total number of Tiles in the Global Tile Clipping area~~

~~OP\_size: OPB size for Opaque List~~

~~OM\_size: OPB size for opaque Modifier Volume list~~

~~TP\_size: OPB size for translucent list~~

~~TM\_size: OPB size for translucent Modifier Volume list~~

~~T\_num: Number of Tiles in the Global Tile Clipping area prior to the Tile in question~~

Example: GC tile and T\_num

0	1	2	3	4			
5	6	7	8	9			
10	11	12	13	14			
15	16	17	18	19			

In this case, GC\_tile = 20, and the number indicated in each tile is T\_num.

The gray area indicates those tiles that are within the Global Tile Clip area.

~~Fig. 3-72~~



~~In HOLLY2, list continuation processing has been added to the HOLLY1 specifications;~~ If list initialization is performed through the TA\_LIST\_INIT register, the TA initializes its internal status, loads the value in the TA\_NEXT\_OPB\_INIT register into the TA\_NEXT\_OPB register, and then allocates space in texture memory as the OPB initial area, from the address that is specified in the TA\_OL\_BASE register to the address that is specified in the TA\_NEXT\_OPB\_INIT register.

If list continuation processing is performed through the TA\_LIST\_CONT register, the TA initializes its internal status in the same manner as before, but leaves the TA\_NEXT\_OPB register unchanged. As a result, the additional OPB for the list that is continuing to be input is stored after the OPB that was input last time.

The sequence of the Tile OPBs that are stored in texture memory is the same as when Tiles are arranged horizontally. The order of the lists is: (1) Opaque (2) Opaque Modifier Volume (3) Translucent (4) Translucent Modifier Volume (5) Punch Through.

**Positions where each OPB is stored after initialization**

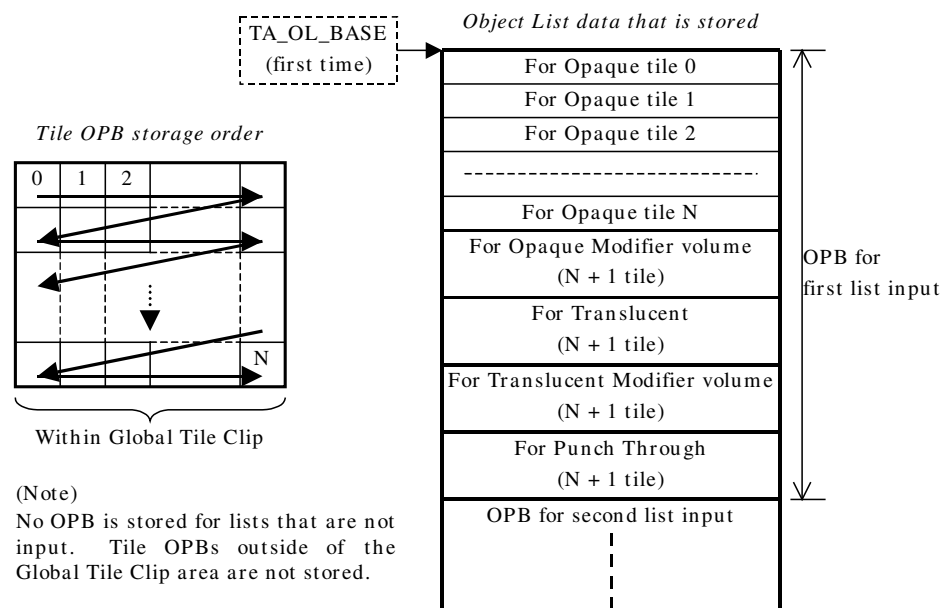


Fig. 3-73

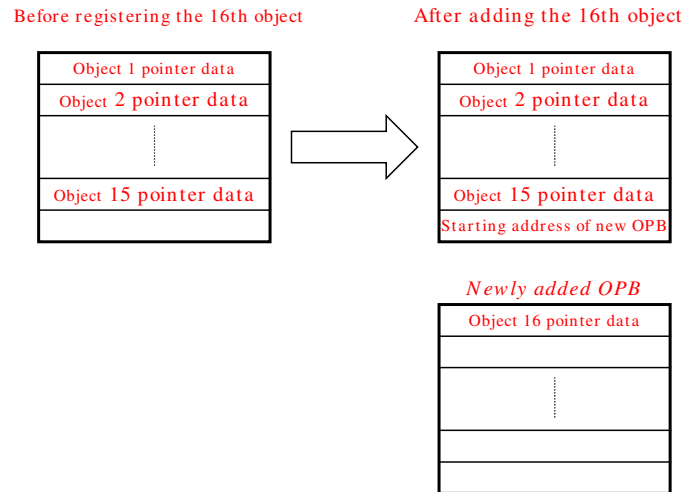
### § 3.7.3.4.2 Adding an OPB

The number of objects that can be registered in one Object Pointer Block is (OPB size - 1).

~~In HOLLY1, if the number of objects that are included in that Tile exceeds that value, the TA adds a new Object Pointer Block in texture memory. The TA automatically stores the starting address for the newly added OPB in the final data of the OPB. (Refer to "Object Pointer Block Link Data" in section 3.7.8.)~~

Following figure is deleted.

#### Example: Adding a new OPB (when the OPB size is 16)



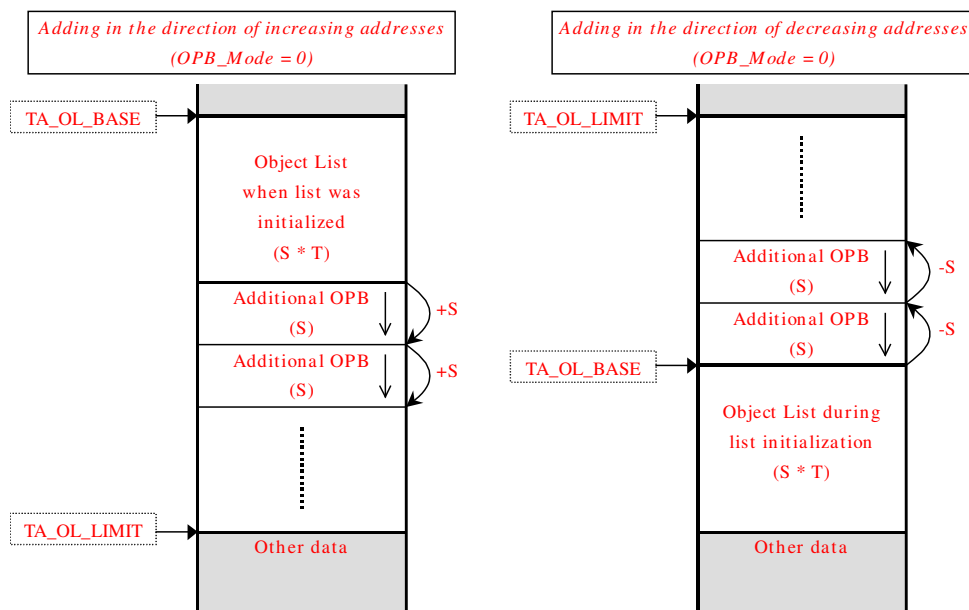
~~Fig. 3-74~~

~~When adding a new Object Pointer Block, the address direction can be specified through the TA\_ALLOC\_CTRL register.~~

#### Example: Address direction for the additional OPB

S: Object pointer block size of the list that is being processed

T: Total number of tiles in the global tile clipping area



~~Fig. 3-75~~

~~In HOLLY2, If the number of objects that are included in a Tile exceed that value, the TA~~

allocates an additional OPB area sufficient for the OPB size of that list, starting from the address indicated by the TA\_NEXT\_OPB register, and stores a pointer for the excess object in that address. At the same time, the value in the TA\_NEXT\_OPB register, which is the starting address of the additional OPB, is automatically stored in the last address of the OPB that was stored previously. (Refer to "Object Pointer Block Link Data," section 3.7.8.) In addition, the value in the TA\_NEXT\_OPB register is updated to the starting address of the next additional OPB.

When a list is input on a continuation basis, the additional OPB is added after the address where the previous OPB was stored.

**Example: Adding a new OPB (when the OPB size is 16)**

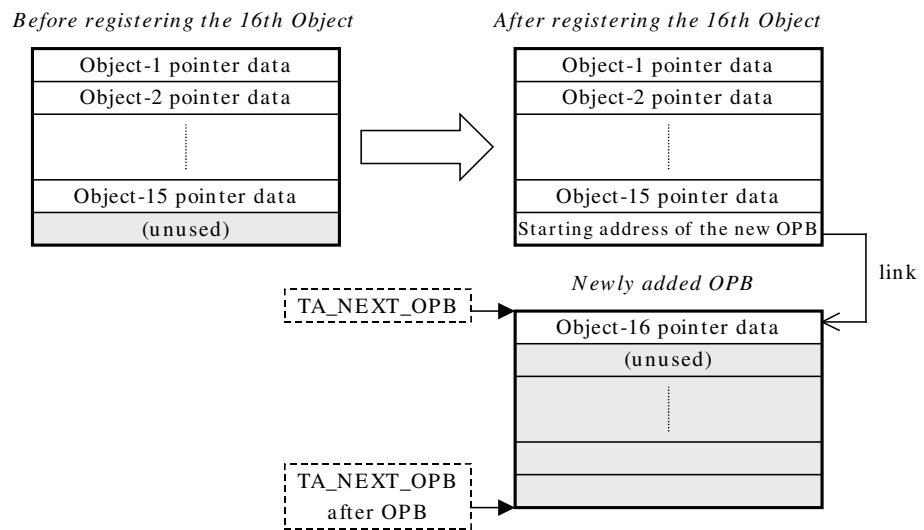


Fig. 3-76

The direction of the addresses when adding a new OPB can be specified in the TA\_ALLOC\_CTRL register.]

**Example: Additional OPB address direction**

S: OPB size in list being processed  
T: Total number of tiles in global tile clip area  
N: List input count

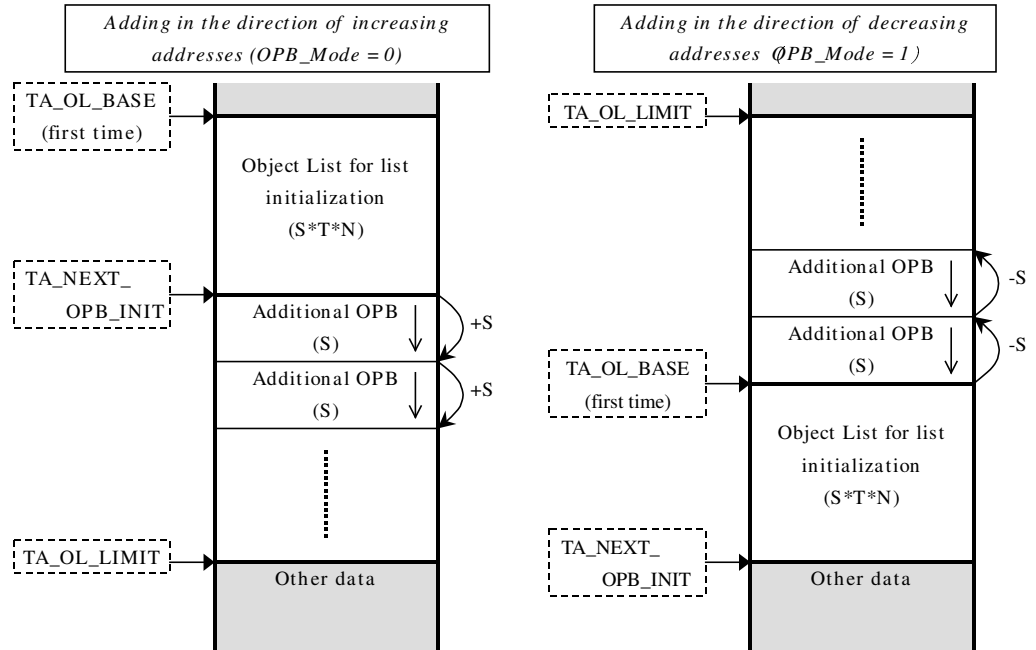


Fig. 3-77

### § 3.7.3.4.3 Processing When a Limit Address Is Exceeded

If the Object List data storage address has exceeded the Object List limit address specified in the TA\_OL\_LIMIT register, that Object List data is not stored in texture memory. In this event, the TA stores the "End of List" Object List data at the limit address, and links to this "End of List" the OPBs for all of the Tiles for which additional OPBs could not be stored. Therefore, the address that is specified in the TA\_OL\_LIMIT register cannot be used for other data, etc.

**Example of processing when the limit address has been exceeded by other than an additional OPB for Tile A (\*Additional specification in HOLLY2)**

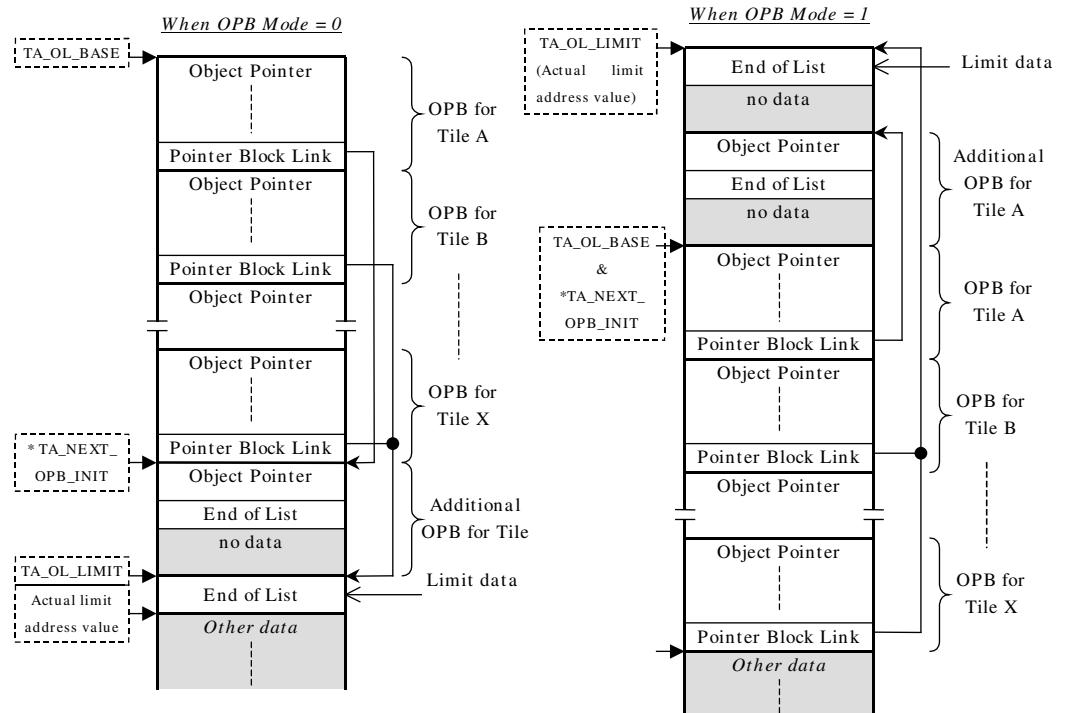


Fig. 3-78

### § 3.7.3.5 ISP/TSP Parameter Generation

Polygon lists that are input from the CPU are rearranged by the TA into ISP/TSP Parameter format, and are stored in texture memory in order, starting from the address that is specified in the TA\_ISP\_BASE register. ~~In HOLLY2,~~ When a list is input on a continuation basis, the ISP/TSP Parameters are stored after the address where the parameters were stored the previous time. However, data for polygons that do not exist at all within the effective drawing area jointly defined by the Global Tile Clipping area and the User Tile Clipping areas is discarded and is not stored in texture memory.

Furthermore, polygon data is not stored in an address that exceeds the ISP/TSP Parameter limit address that was specified in the TA\_ISP\_LIMIT register. Therefore, a display list in which the limit address was exceeded cannot be used for drawing. Users must take into consideration the size of the ISP/TSP Parameter data that will be stored in texture memory, based on number of polygons that are to be input to the TA, when specifying the limit address.

Three formats are supported for the Shading Color data within the polygon data that is input to the TA: "Packed Color," "Floating Color," and "intensity." However, the CORE supports "Packed Color" only. Therefore, the TA converts the shading data that is input into 32-bit Packed Color format before storing the data in texture memory.

#### <Shading data conversion>

##### Floating Color → Packed Color

The TA converts each element of ARGB data into a fixed decimal value between 0.0 and 1.0, multiplies the value by 255, and packs the result in a 32-bit value.

##### Intensity → Packed Color

Regarding alpha values, the TA converts the specified Face Color Alpha value into a fixed decimal value between 0.0 and 1.0, multiplies the value by 255, and derives an 8-bit value. Regarding RGB values, the TA converts the specified Face Color R/G/B value into a fixed decimal value between 0.0 and 1.0, multiplies the value by 255, converts the intensity value into a fixed decimal value between 0.0 and 1.0, multiplies the converted R/G/B value and the converted intensity value together, multiplies that result by 255, and derives an 8-bit value for each of R, G, and B. Finally, the TA packs each 8-bit value into a 32-bit value.

### § 3.7.4 Explanation of TA Parameters

There are three types of polygon data that are input to the TA: Control Parameters, Global Parameters, and Vertex Parameters. One data element consists of 32 or 64 bytes of each. The first four bytes of each type of parameter are called the Parameter Control Word, which is used for determining the parameter settings and type.

The Control Parameters are used for special processing, such as ending the Object List. The Global Parameters consist of various settings that apply to the polygons that are expressed by the Vertex Parameters, which are input after the Global Parameters. In the case of a Triangle polygon, a minimum of at least three Vertex Parameters is required. The Vertex Parameters contain vertex data for polygons that also use the settings in the Global Parameters that were input previously.

The Global Parameters and Vertex Parameters must be grouped together for input by list type (opaque, translucent, etc.) Furthermore, although there are no restrictions on the input order for the four types of lists (opaque, etc.), parameters for polygons of one particular type may only be input once.

#### § 3.7.4.1 Control Parameter

The Control Parameters are used for special processing, such as ending the Object List.

Parameter type	Processing
End Of List	Ends the list for the type (opaque, translucent, etc.) that is currently being input. This parameter must be input last in the list for each type.
User Tile Clip	Specifies the User Tile Clipping values. The specified values remain valid until they are updated. The clipping values specify the Tile number for the Tiles in the upper left and lower right corners of the rectangular area. Only the lower six bits are valid for Tile numbers in the X direction. Only the upper four bits are valid for Tile numbers in the Y direction. In addition, do not specify a value greater than 39 (0x27) for the Tile number in the X direction, or greater than 14 (0xE) for the Tile number in the Y direction. User Clip X Min: Upper left Tile number in the X direction (0 to 39) User Clip Y Min: Upper left Tile number in the Y direction (0 to 14) User Clip X Max: Lower right Tile number in the X direction (0 to 39) User Clip Y Max: Lower right Tile number in the Y direction (0 to 14)
Object List Set	This is used to register polygons of a type that is not supported by the TA, or to register just an Object List. The specified Object Pointer data is stored in the Object List for the Tiles in the specified bounding box. In the Object Pointer value, specify the data that is to be stored in the Object List. In the bounding box values, specify the upper left and lower right Tile numbers that define the rectangular area that includes the object. Only the lower six bits are valid for Tile numbers in the X direction. Only the upper four bits are valid for Tile numbers in the Y direction. In addition, do not specify a value greater than 39 (0x27) for the Tile number in the X direction, or greater than 14 (0xE) for the Tile number in the Y direction. Bounding Box X Min: Upper left Tile number in the X direction (0 to 39) Bounding Box Y Min: Upper left Tile number in the Y direction (0 to 14) Bounding Box X Max: Lower right Tile number in the X direction (0 to 39) Bounding Box Y Max: Lower right Tile number in the Y direction (0 to 14)

Once the End of List parameter has been input and all of the data for polygons of the type

currently being input have been stored in texture memory, one of four types of interrupt signals (corresponding to the polygon type) is output. As a result of this interrupt request, the CPU knows that the TA has completed processing for that polygon type.

When performing clipping processing that uses a User Tile Clipping area, the clipping values must already be specified beforehand through the User Tile Clip parameters.

**<Using Object List Set>**

Use the Object List Set parameters in the following cases:

- (1) When you only want to register an Object List, without storing the ISP/TSP Parameters again for polygons that will be absolutely unchanged on the screen
- (2) When you only want to register a type of polygon that the TA does not support (such as a Gouraud shaded Quad polygon)

Input the Object List Set parameters after the list has been initialized by the TA\_LIST\_INIT register, or after the End Of List parameter. First, store the ISP/TSP Parameters that were generated by the CPU directly in the texture memory, and then set the last address value in the TA\_ISP\_BASE register. For the data in the parameters, set the values for the bounding box for the polygon to be registered under the ISP/TSP Parameters that were already stored, and set the Object Pointer value that specifies the starting address, etc., for the ISP/TSP Parameters. Once the Object List Set parameters have been completely input, input the remaining polygon data as normal parameters.



### § 3.7.4.2 Global Parameter

The Global Parameters consist of three types of control data that are stored in texture memory as the ISP/TSP Parameters, and parameters that specify the data configuration of the Vertex Parameters that are input subsequently.

Parameter type	Processing
Polygon	<p>This is used when the list type is either Opaque or Translucent. There are five types of parameters with different data configurations. The Vertex Parameters that are input subsequently are used to generate Triangle polygon data for the strip number that is specified in the Parameter Control Word.</p> <p>These parameters specify the Face Color when the Shading Color type in the Vertex Parameters is "Intensity" format. The Face Color is the color data (a 32-bit floating-point decimal value) that is multiplied by the vertex intensity value. There are two types of Face Colors: one for the Base Color and one for the Offset Color.</p> <p style="margin-left: 40px;">Face Color:                      for Base Color</p> <p style="margin-left: 40px;">Face Offset Color:              for Offset Color</p> <p>Input polygons for which the settings will change inside and outside of the volume in "with Two Volumes" format. In "with Two Volumes" format, two TSP Instruction Words, two Texture Control Words, and two Face Colors are set.</p> <p>Also specify Sort-DMA data, if necessary.</p>
Sprite	<p>This is used when the list type is either Opaque or Translucent. The Vertex Parameters that are input subsequently are used to generate flat shaded independent Quad polygon data.</p> <p>Specify the Base Color and Offset Color for Flat Shading. (Both use 32-bit packed ARGB data.)</p> <p>Also specify Sort DMA data, if necessary.</p>
Modifier Volume	<p>This is used when the list type is either Opaque Modifier Volume or Translucent                      Modifier                      Volume.</p> <p>The Vertex Parameters that are input subsequently are used to generate independent Triangle polygon data. The Triangle polygon data in the Vertex Parameters that are input after the Global Parameter that was specified as the end of the volume in the Parameter Control Word is registered in all of the Tile Object Lists that encompass the entire volume.</p>

When transferring translucent polygon data by using the Sort-DMA function (refer to 2.6.5.3), it is necessary to set the Sort-DMA data. In addition, when using the Sort-DMA function, the "Polygon Type 1" Global Parameter must not be used.

Data	Description
Data Size for Sort-DMA	This specifies in 32-byte units the data size (including the Control Parameters, the Global Parameters, and the Vertex Parameters) of the objects that includes the Global Parameters in question. Only the lower 8 bits are valid; if "0" is specified, it is treated as "256." Note that "1" through "3" may not be specified.
Next Address for Sort-DMA	This specifies the offset address value for the object parameters that are to be transferred next. The actual address is the sum of the base address value specified in the SB_SDBAAW (0x005F 6814) register plus this value. Only the lower 27 bits are valid. The SB_SDLAS register (0x005F 681C) is used to specify whether this value is specified in 1-byte or 32-byte units. The following values have special meanings: Link End Code (0x0000 0001): Link table read Link All End Code (0x0000 0002): Sort-DMA end

### § 3.7.4.3 Vertex Parameter

The Vertex Parameters specify a variety of data for the vertices. The Vertex Parameters must always be input after the Global Parameters. The data configuration of the Vertex Parameters is determined by the type of Global Parameters (Polygon/Sprite/Volume Modifier) that were input previously and the Parameter Control Word.

Parameter type	Processing
Polygon	This is used when the Global Parameters that were input previously were of the Polygon type. There are 15 types of parameters with different data configurations, and it is necessary to input a minimum of three. "End of Strip" must be specified at the end of the object. If the parameters are to be input in "with Two Volumes" format, set two each of the UV, Base/Offset Color, and Base/Offset Intensity parameters for the inside and outside of the volume.
Sprite	This is used when the Global Parameters that were input previously were of the Sprite type. There are two types of parameters with different data configurations. Specify data for four vertices for one polygon to generate Flat-Shaded independent Quad polygon data.
Modifier Volume	This is used when the Global Parameters that were input previously were of the Modifier Volume type. Specify data for three vertices for one polygon to generate independent Triangle polygon data for the Modifier Volume.

Data	Data
X, Y, Z	Vertex coordinates (IEEE single-precision floating point values) Specify the screen coordinates for X and Y, and a reciprocal (1/z or 1/w) for Z.
U, V	Texture coordinates (16-bit or 32-bit floating point values) For 32-bit UV, specify IEEE single-precision floating point values. For 16-bit UV, extract the upper 16 bits of the 32-bit floating point values for U and V respectively, and specify a 32-bit value consisting of U as the upper 16 bits and V as the lower 16 bits.
Base/Offset Color	Shading Color data (32-bit integers) for Packed Color format Store these values as is in the ISP/TSP Parameters.
Base/Offset Color Alpha/R/G/B	Shading Color data (32-bit floating-point values) for Floating Color format Convert each data element into an 8-bit integer (0 to 255), group them

	into 32-bit values, and store them in the ISP/TSP Parameters.
Base/Offset Intensity	Shading Color data (32-bit floating-point values) for Intensity format Convert the Face Color alpha values specified in the Global Parameters into 8-bit integers (0 to 255). Multiply the RGB values by the corresponding Face Color R/G/B value, and convert the result into an 8-bit integer (0 to 255). Combine each 8-bit value thus obtained into a 32-bit value and store it in the ISP/TSP Parameters.

In the case of the Polygon type, the last Vertex Parameter for an object must have "End of Strip" specified. If Vertex Parameters with the "End of Strip" specification were not input, but parameters other than the Vertex Parameters were input, the polygon data in question is ignored and an interrupt signal is output.

When using Bump Mapping, input the Bump Map parameters instead of the Offset Color. The Shading Color type must be set to something other than Intensity format. The Bump Map parameters are valid for the third and subsequent vertices from the start of the strip.

In the case of a flat-shaded polygon, the Shading Color data (the Base Color, Offset Color, and Bump Map parameters) become valid starting with the third vertex after the start of the strip.

#### § 3.7.4.4 Parameter Control Word

This data is used to determine the data configuration and type of each parameter. The Parameter Control Word is added to the first four bytes.

bit 31-24	23-16	15-0
Para Control	Group Control	Obj Control

##### § 3.7.4.4.1 Para Control

This is the control data for all of the parameters. The End Of Strip bit (only) is valid only in the Vertex Parameters.

bit 31-29	28	27-26	25-24
Para Type	End Of Strip	Reserved	List Type

~~The control data for HOLLY2 is shown below:~~

bit 31-29	28	27	26-24
Para Type	End Of Strip	Reserved	List Type

##### **Para Type**

Specifies the parameter type.

Parameter type	Parameter	Hex Code
Control Parameter	End Of List	0
	User Tile Clip	1
	Object List Set	2
	Reserved	3
Global Parameter	Polygon or Modifier Volume	4
	Sprite	5
	Reserved	6
Vertex Parameter		7

##### **End Of Strip**

Valid only in the Vertex Parameters. A parameter in which this bit is "1" ends a strip. The Spite and Modifier Volume Vertex Parameter must be set to "1".

##### **List Type**

Specifies the Object List type. This value is valid in the following four cases:

- (a) The first Global Parameter that was input after list initialization through the TA\_LIST\_INIT register or after list continuation processing through the TA\_LIST\_CONT register.
- (b) The first Global Parameter that was input after an End Of List parameter was input
- (c) The first Object List Set parameter that was input after list initialization through the TA\_LIST\_INIT register or after list continuation processing through the TA\_LIST\_CONT register.
- (d) The first Object List Set parameter that was input after an End Of List parameter was input

List type	Code	Parameters that can be used
Opaque	0	Polygon or Sprite
Opaque Modifier Volume	1	Modifier Volume
Translucent	2	PolygonまたはSprite
Translucent Modifier Volume	3	Modifier Volume
Punch Through (HOLLY2)	4	Polygon or Sprite
Reserved (HOLLY2)	5~7	Prohibited

#### § 3.7.4.4.2 Group Control

This is the control data for an object group. This is valid only in Global Parameters.

bit 23	22-20	19-18	17-16
Group_En	Reserved	Strip_Len	User_Clip

##### **Group\_En**

Set "1" in order to update the Strip\_Len and User\_Clip settings. If "0" is set, the existing settings are used.

##### **Strip\_Len**

Specifies the length of the strip that is to be partitioned. This is valid only when Group\_En is "1".

Code	strip分割数
0	1 strip
1	2 strip
2	4 strip
3	6 strip

##### **User\_Clip**

Specifies how the User Tile Clipping area is to be used. This is valid only when Group\_En is "1".

Code	User Tile Clipping
0	Disable
1	Reserved
2	Inside enable
3	Outside enable

#### § 3.7.4.4.3 Obj Control

This data sets an object. This is valid only in Global Parameters.

bit 15-8	7	6	5-4	3	2	1	0
----------	---	---	-----	---	---	---	---

Reserved	Shadow	Volume	Col_Type	Texture	Offset	Gouraud	16bit_UV
----------	--------	--------	----------	---------	--------	---------	----------

### **Shadow**

The value of this bit is used in "Shadow bit (bit 24)" of the Object List. This bit must be set to "1" for parameters in "with Two Volumes" format. In Intensity Volume Mode, set this bit to "1" in order to perform shadow processing on a polygon.

### **Volume**

This specifies whether the parameters are in "with Two Volumes" format, or whether or not the polygon is the last Triangle polygon in the volume. In the case of the Modifier Volume type, the Volume Instruction (bits 31 to 29) in the ISP/TSP Instruction Word must be set correctly, along with this bit. In the case of the Sprite type, set this bit to "0."

Parameter type	Bit Value	Explanation
Polygon	0	For a format other than "with Two Volumes"
	1	For "with Two Volumes" format
Modifier Volume	0	For a Triangle polygon that is not the last in the volume
	1	For a Triangle polygon that is the last in the volume

### Shadow Bit and Volume Bit Combinations

Shadow	Volume	Explanation	
		Polygon	Modifier Volume
0	0	Normal polygons, or polygons for which shadow processing is not performed (in Intensity Volume mode)	Triangle polygons that are not the last in the volume
0	1	Reserved	Triangle polygon that is the last in the volume
1	0	Polygons for which shadow processing is performed (in Intensity Volume mode)	Reserved
1	1	Polygons in "with Two Volumes" format	Reserved

### **Col\_Type**

Specifies the format for the Shading Color data that is to be input. For Intensity format, if the Face Color that was used for the previous object is to be used for the current object, the amount of data that has to be transferred can be reduced by specifying "Intensity Mode 2," since the same Face Color data does not have to be input again.

Code	Color data format	Description
0	Packed Color	8-bit values for each of A, R, G, and B
1	Floating Color	32-bit floating-point values for each of A, R, G, and B
2	Intensity Mode 1	The Face Color is specified by the immediately preceding Global Parameters.
3	Intensity Mode 2	The previous Face Color value that was specified by Global Parameters in Intensity Mode 1 is used for the Face Color. Note that a polygon for which this mode is used must only be input after a Mode 1 polygon has been input at least once. It is not necessary for the Mode 1 polygon to have immediately preceded this polygon.

### **Texture**

Set this bit to "1" when using a texture. The value of this bit is used in the Texture bit in the TSP Instruction Word in the ISP/TSP Parameters.

### **Offset**

Set this bit to "1" when using an Offset Color. The value of this bit is used in the Offset bit in the TSP Instruction Word in the ISP/TSP Parameters.

Set this bit to "1" for a Bump Mapped polygon.

### **Gouraud**

Set this bit to "1" when using Gouraud Shading. When this bit is "0," Flat Shading is set and the Shading Color data for the third and subsequent vertices becomes valid. The value of this bit is used in the Gouraud Shading bit in the TSP Instruction Word in the ISP/TSP Parameters.

Set "0" in the case of a Spite.

### **16bit\_UV**

Set this bit to "1" when using 16-bit values for the Texture UV coordinate values. If this bit is "0," 32-bit values are used. The value of this bit is used in the 16-bit UV bit in the TSP Instruction Word in the ISP/TSP Parameters.

Set "1" in the case of a Spite.

Four bits in the ISP/TSP Instruction Word are overwritten with the corresponding bit values from the Parameter Control Word.

Parameter Control Word			ISP/TSP Instruction Word	
Bit 3	Texture	→	Bit 25	Texture
Bit 2	Offset	→	Bit 24	Offset
Bit 1	Gouraud	→	Bit 23	Gouraud shading
Bit 0	16bit_UV	→	Bit 22	16 Bit UV

### § 3.7.5 Parameter Format

The three types of parameters (Control Parameters, Global Parameters, and Vertex Parameters) are actually input to the TA in the form of 64-bit data. The data configuration used is shown below.

bit 63-32	bit 31-0
0x04	0x00
0x0C	0x08
0x14	0x10
0x1C	0x18
0x24	0x20
0x2C	0x28
0x34	0x30
0x3C	0x38

#### § 3.7.5.1 Control Parameter Format

End Of List	
0x00	Parameter Control Word(0x0000 0000)
0x04	(ignored)
0x08	(ignored)
0x0C	(ignored)
0x10	(ignored)
0x14	(ignored)
0x18	(ignored)
0x1C	(ignored)

User Tile Clip	
0x00	Parameter Control Word(0x2000 0000)
0x04	(ignored)
0x08	(ignored)
0x0C	(ignored)
0x10	User Clip_X_Min
0x14	User Clip_Y_Min
0x18	User Clip_X_Max
0x1C	User Clip_Y_Max

←	invalid	bit 5-0
←	invalid	bit 3-0
←	invalid	bit 5-0
←	invalid	bit 3-0

Object List Set	
0x00	Parameter Control Word(0x4000 0000)
0x04	Object Pointer
0x08	(ignored)
0x0C	(ignored)
0x10	Bounding Box X_Min
0x14	Bounding Box Y_Min
0x18	Bounding Box X_Max
0x1C	Bounding Box Y_Max

←	invalid	bit 5-0
←	invalid	bit 3-0
←	invalid	bit 5-0
←	invalid	bit 3-0

### § 3.7.5.2 Global Parameter Format

<b>Polygon Type 0 (Packed/Floating Color)</b>	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word
0x0C	Texture Control Word
0x10	(ignored)
0x14	(ignored)
0x18	<i>Data Size for Sort DMA</i>
0x1C	<i>Next Address for Sort DMA</i>

<b>Polygon Type 1 (Intensity, no Offset Color)</b>	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word
0x0C	Texture Control Word
0x10	Face Color Alpha
0x14	Face Color R
0x18	Face Color G
0x1C	Face Color B

<b>Polygon Type 2 (Intensity, use Offset Color)</b>	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word
0x0C	Texture Control Word
0x10	(ignored)
0x14	(ignored)
0x18	<i>Data Size for Sort DMA</i>
0x1C	<i>Next Address for Sort DMA</i>
0x20	Face Color Alpha
0x24	Face Color R
0x28	Face Color G
0x2C	Face Color B
0x30	Face Offset Color Alpha
0x34	Face Offset Color R
0x38	Face Offset Color G
0x3C	Face Offset Color B

<b>Polygon Type 3 (Packed Color, with Two Volumes)</b>	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word 0
0x0C	Texture Control Word 0
0x10	TSP Instruction Word 1
0x14	Texture Control Word 1
0x18	<i>Data Size for Sort DMA</i>
0x1C	<i>Next Address for Sort DMA</i>

<b>Polygon Type 4 (Intensity, with Two Volumes)</b>	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word 0
0x0C	Texture Control Word 0
0x10	TSP Instruction Word 1
0x14	Texture Control Word 1
0x18	<i>Data Size for Sort DMA</i>
0x1C	<i>Next Address for Sort DMA</i>
0x20	Face Color Alpha 0
0x24	Face Color R 0
0x28	Face Color G 0
0x2C	Face Color B 0
0x30	Face Color Alpha 1
0x34	Face Color R 1
0x38	Face Color G 1
0x3C	Face Color B 1



Sprite (Packed Color)	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	TSP Instruction Word
0x0C	Texture Control Word
0x10	Base Color
0x14	Offset color
0x18	<i>Data Size for Sort DMA</i>
0x1C	<i>Next Address for Sort DMA</i>

Modifier Volume	
0x00	Parameter Control Word
0x04	ISP/TSP Instruction Word
0x08	(ignored)
0x0C	(ignored)
0x10	(ignored)
0x14	(ignored)
0x18	(ignored)
0x1C	(ignored)

<Notes>

- If textures are not used, the Texture Control Word is ignored.
- In the case of Polygon Type 4 (Intensity, with Two Volumes), the Face Color is used in both the Base Color and the Offset Color.
- The seventh (0x18) and eighth (0x1C) data items in all Global Parameter configurations, except for Polygon Type 1 and Modifier Volume, are Sort-DMA parameters. It is necessary to set these parameters if data is to be transferred using Sort-DMA.

### § 3.7.5.3 Vertex Parameter Format

~~In HOLLY2, there are changes to the parameters for polygon types 0 and 2. Refer to the end of this section for details.~~

Polygon Type 0 (Non-Textured, Packed Color)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	(ignored)
0x14	(ignored)
0x18	Base Color
0x1C	(ignored)

Polygon Type 1 (Non-Textured, Floating Color)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	Base Color Alpha
0x14	Base Color R
0x18	Base Color G
0x1C	Base Color B

Polygon Type 2 (Non-Textured, Intensity)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	(ignored)
0x14	(ignored)
0x18	Base Intensity
0x1C	(ignored)

Polygon Type 3 (Packed Color)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U
0x14	V
0x18	Base Color
0x1C	Offset Color

Polygon Type 4 (Packed Color, 16bit UV)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U / V
0x14	(ignored)
0x18	Base Color
0x1C	Offset Color

Polygon Type 5 (Floating Color)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U
0x14	V
0x18	(ignored)
0x1C	(ignored)
0x20	Base Color Alpha
0x24	Base Color R
0x28	Base Color G
0x2C	Base Color B
0x30	Offset Color Alpha
0x34	Offset Color R
0x38	Offset Color G
0x3C	Offset Color B

Polygon Type 6 (Floating Color, 16bit UV)	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U / V
0x14	(ignored)
0x18	(ignored)
0x1C	(ignored)
0x20	Base Color Alpha
0x24	Base Color R
0x28	Base Color G
0x2C	Base Color B
0x30	Offset Color Alpha
0x34	Offset Color R
0x38	Offset Color G
0x3C	Offset Color B

Polygon Type 7 (Intensity)	
-------------------------------	--

Polygon Type 8 (Intensity, 16bit UV)	
---	--

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U
0x14	V
0x18	Base Intensity
0x1C	Offset Intensity

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U / V
0x14	(ignored)
0x18	Base Intensity
0x1C	Offset Intensity

**Polygon Type 9  
(Non-Textured, Packed Color,  
with Two Volumes)**

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	Base Color 0
0x14	Base Color 1
0x18	(ignored)
0x1C	(ignored)

**Polygon Type 10  
(Non-Textured, Intensity,  
with Two Volumes)**

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	Base Intensity 0
0x14	Base Intensity 1
0x18	(ignored)
0x1C	(ignored)

**Polygon Type 11  
(Textured, Packed Color,  
with Two Volumes)**

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U0
0x14	V0
0x18	Base Color 0
0x1C	Offset Color 0
0x20	U1
0x24	V1
0x28	Base Color 1
0x2C	Offset Color 1
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

**Polygon Type 12  
(Textured, Packed Color, 16bit  
UV,  
with Two Volumes)**

0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U0 / V0
0x14	(ignored)
0x18	Base Color 0
0x1C	Offset Color 0
0x20	U1 / V1
0x24	(ignored)
0x28	Base Color 1
0x2C	Offset Color 1
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

<b>Polygon Type 13 (Textured, Intensity, with Two Volumes)</b>	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U <sub>0</sub>
0x14	V <sub>0</sub>
0x18	Base Intensity 0
0x1C	Offset Intensity 0
0x20	U <sub>1</sub>
0x24	V <sub>1</sub>
0x28	Base Intensity 1
0x2C	Offset Intensity 1
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

<b>Polygon Type 14 (Textured, Intensity, 16bit UV, with Two Volumes)</b>	
0x00	Parameter Control Word
0x04	X
0x08	Y
0x0C	Z
0x10	U <sub>0</sub> / V <sub>0</sub>
0x14	(ignored)
0x18	Base Intensity 0
0x1C	Offset Intensity 0
0x20	U <sub>1</sub> / V <sub>1</sub>
0x24	(ignored)
0x28	Base Intensity 1
0x2C	Offset Intensity 1
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

<b>Sprite Type 0 (for Line)</b>	
0x00	Parameter Control Word
0x04	AX
0x08	AY
0x0C	AZ
0x10	BX
0x14	BY
0x18	BZ
0x1C	CX
0x20	CY
0x24	CZ
0x28	DX
0x2C	DY
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

<b>Sprite Type 1 (for Sprite)</b>	
0x00	Parameter Control Word
0x04	AX
0x08	AY
0x0C	AZ
0x10	BX
0x14	BY
0x18	BZ
0x1C	CX
0x20	CY
0x24	CZ
0x28	DX
0x2C	DY
0x30	(ignored)
0x34	AU / AV
0x38	BU / BV
0x3C	CU / CV

<b>Modifier Volume</b>	
0x00	Parameter Control Word
0x04	AX
0x08	AY
0x0C	AZ
0x10	BX
0x14	BY
0x18	BZ
0x1C	CX
0x20	CY
0x24	CZ
0x28	(ignored)
0x2C	(ignored)
0x30	(ignored)
0x34	(ignored)
0x38	(ignored)
0x3C	(ignored)

<Notes>

- The data configuration that is used is determined by the Parameter Control Word in the Global Parameters; unnecessary data is automatically ignored.
- When using Bump Mapping, the Offset Color data is used as the Bump Map parameters (K1K2K3Q). The Bump Map parameters are valid for the third and subsequent vertices from the start of the strip.

The polygon type 0 and 2 parameter formats that were changed in HOLLY2 are shown below.  
\* In both polygon type 0 and type 2, 0x10 and 0x18 are reversed, compared with HOLLY1.

<b>Polygon Type 0 (Non-Textured, Packed-Color)</b>	
<del>0x0</del> <del>0</del>	<del>Parameter Control Word</del>
<del>0x0</del> <del>4</del>	<del>X</del>
<del>0x0</del> <del>8</del>	<del>Y</del>
<del>0x0</del> <del>€</del>	<del>Z</del>
<del>0x1</del> <del>0</del>	<del>*(ignored)</del>
<del>0x1</del> <del>4</del>	<del>(ignored)</del>
<del>0x1</del> <del>8</del>	<del>*Base Color</del>
<del>0x1</del> <del>€</del>	<del>(ignored)</del>

<b>Polygon Type 2 (Non-Textured, Intensity)</b>	
<del>0x0</del> <del>0</del>	<del>Parameter Control Word</del>
<del>0x0</del> <del>4</del>	<del>X</del>
<del>0x0</del> <del>8</del>	<del>Y</del>
<del>0x0</del> <del>€</del>	<del>Z</del>
<del>0x1</del> <del>0</del>	<del>*(ignored)</del>
<del>0x1</del> <del>4</del>	<del>(ignored)</del>
<del>0x1</del> <del>8</del>	<del>*Base Intensity</del>
<del>0x1</del> <del>€</del>	<del>(ignored)</del>

## § 3.7.6 Overview of TA Parameters

### § 3.7.6.1 Notes When Using the TA

The following points must be noted when using the Tile Accelerator to generate display lists for the CORE.

#### <Register-related>

- Set the TA\_GLOB\_TILE\_CLIP register, the TA\_ALLOC\_CTRL register and the TA\_NEXT\_OPB\_INIT register ~~in HOLLY2~~ before initializing lists through the TA\_LIST\_INIT register.
- ~~In HOLLY2,~~ be certain to change the TA\_OL\_BASE register to the correct address before performing list continuation processing through the TA\_LIST\_CONT register.
- If the address in texture memory where the Object List is stored exceeds the address specified in the TA\_OL\_LIMIT register, respectively, the TA outputs an interrupt. Although the display list is generated correctly in this case, the resulting image will not appear as expected.
- If the address in texture memory where the ISP/TSP Parameters are stored exceeds the address specified in the TA\_ISP\_LIMIT register, the TA outputs an interrupt. The display list (ISP/TSP Parameters) is not generated correctly in this case, and therefore should not be used for drawing. It is necessary in this case to reconsider the memory allocations and to start over from list initialization. Users must take into consideration the size of the memory needed for the ISP/TSP Parameters, based on number of polygons that are to be input to the TA, when allocating memory.

#### <Parameter input-related>

- The five types of polygon data (Opaque, Opaque Modifier Volume, Translucent, Translucent Modifier Volume, and Punch Through) ~~(five types in the case of HOLLY2)~~ must all be grouped and input together. The End Of List parameter must also be input at the end of each list.
- Although there are no restrictions concerning the order in which the five types of lists are input ~~(five types, in the case of HOLLY2)~~, only one list of each type may be input. ~~In HOLLY2,~~ when inputting a list in several pieces, list continuation processing through the TA\_LIST\_CONT register must be used.
- If "No List" is specified for the Object Pointer Block size in the TA\_ALLOC\_CTRL register for a certain type of list, parameters for that type of list may not be input.
- After initializing the lists through the TA\_LIST\_INIT register, input the User Tile Clipping area parameters first so that the User Tile Clipping values are set.
- Input the Object List Set parameter either after initializing the lists through the TA\_LIST\_INIT register, or after an End Of List parameter.
- When inputting data for a Triangle polygon, input at least three Vertex Parameters.
- When inputting data for a strip of Triangle polygons, End Of Strip must be specified in the last Vertex Parameter of the strip.
- If there is no need to change the Global Parameters, a Vertex Parameter for the next polygon may be input immediately after inputting a Vertex Parameter for which "End of Strip" was specified.
- When inputting polygon data using Intensity Mode 2, it is essential for polygon data in Intensity Mode 1 to already have been input at least once. It is not necessary, however, for the Mode 1 polygon to have immediately preceded the Mode 2 polygon.
- When inputting data for a Modifier Volume, input Global Parameters that indicate that the data being input is for a Modifier Volume, before inputting the Vertex Parameters for the last polygon of the Volume.

#### <Miscellaneous>

- The TA generates Object Lists and the ISP/TSP Parameters; the Region Array is created by the CPU and then stored directly in texture memory.
- When inputting data for an independent Triangle polygon or Quad polygon, efficient display lists will be generated if data for polygons that are likely to be included in the same Tiles are input together.
- In order to prevent memory space that will not be used from being allocated for the Object Pointer Block for a list of a type that will not be displayed on the screen, it is recommended that "No List" be set in the TA\_ALLOC\_CTRL register.

### § 3.7.6.2 Parameter Combinations

The data configurations and combinations of the Global Parameters and the Vertex Parameters are determined according to the Parameter Control Word setting in the Global Parameters.

List_Type	Parameter Control Word						Global Parameter	Vertex Parameter
	bit 6	5-4	3	2	1	0		
	Volume	Col_Type	Texture	Offset	Gouraud	16bit_UV		
Opaque /Translucent	0	0	0	(0)	x	invalid	Polygon Type 0	Polygon Type 0
	0	1	0	(0)	x	invalid	Polygon Type 0	Polygon Type 1
	0	2	0	(0)	x	invalid	Polygon Type 1	Polygon Type 2
	0	3	0	(0)	x	invalid	Polygon Type 0	Polygon Type 2
	1	0	0	(0)	x	invalid	Polygon Type 3	Polygon Type 9
	1	2	0	(0)	x	invalid	Polygon Type 4	Polygon Type 10
	1	3	0	(0)	x	invalid	Polygon Type 3	Polygon Type 10
	0	0	1	x	x	0	Polygon Type 0	Polygon Type 3
	0	0	1	x	x	1	Polygon Type 0	Polygon Type 4
	0	1	1	x	x	0	Polygon Type 0	Polygon Type 5
	0	1	1	x	x	1	Polygon Type 0	Polygon Type 6
	0	2	1	0	x	0	Polygon Type 1	Polygon Type 7
	0	2	1	1	x	0	Polygon Type 2	Polygon Type 7
	0	2	1	0	x	1	Polygon Type 1	Polygon Type 8
	0	2	1	1	x	1	Polygon Type 2	Polygon Type 8
	0	3	1	x	x	0	Polygon Type 0	Polygon Type 7
	0	3	1	x	x	1	Polygon Type 0	Polygon Type 8
	1	0	1	x	x	0	Polygon Type 3	Polygon Type 11
	1	0	1	x	x	1	Polygon Type 3	Polygon Type 12
	1	2	1	x	x	0	Polygon Type 4	Polygon Type 13
	1	2	1	x	x	1	Polygon Type 4	Polygon Type 14
	1	3	1	x	x	0	Polygon Type 3	Polygon Type 13
	1	3	1	x	x	1	Polygon Type 3	Polygon Type 14
	(0)	(0)	0	(0)	(0)	invalid	Sprite	Sprite Type 0
	(0)	(0)	1	x	(0)	(1)	Sprite	Sprite Type 1
Opaque /Translucent Modifier Volume	All invalid						Modifier Volume	Modifier Volume

#### <Note>

- "x" in the table indicates "Don't care."
- A value in parentheses indicates that the setting in question is fixed at that value, and that the Parameter Control Word setting will be ignored.
- When in "Non-Textured" mode (i.e., the Texture bit is "0"), the set value for the Offset bit is ignored; the value is fixed at "0."

### § 3.7.6.3 Parameter Input Example

Continued on next page



Continued from previous page		
Translucent Polygon	Control Parameter (Object List Set)	Object list setting for object-3 (0x4000 0000)
	Control Parameter (Object List Set)	Object list setting for object-4 (0x4000 0000)
	Control Parameter (Object List Set)	Object list setting for object-5 (0x4000 0000)
	Global Parameter 6 (Polygon Type 0)	Global Parameters for object-6 (0x828C 0010) 6strip, Clip disable, Floating Color, Non-Textured, no Offset, Flat
	Vertex Parameter 0 (Polygon Type 6)	Vertex data 0 (0xE000 0000)
	Vertex Parameter 1 (Polygon Type 6)	Vertex data 1 (0xE000 0000)
	Vertex Parameter 2 (Polygon Type 6)	Vertex data 2 (0xE000 0000)
	Vertex Parameter 3 (Polygon Type 6)	Vertex data 3 (0xE000 0000)
	Vertex Parameter 4 (Polygon Type 6)	Vertex data 4 (0xFFFF FFFF) End Of Strip
	Control Parameter (User Tile Clip)	User Tile clipping value set (0x2000 0000)
	Global Parameter 7 (Sprite)	Global Parameters for object-7 (0xA283 000D) Clip outside enable, Packed Color, Textured, use Offset, Flat, 16bit UV
	Vertex Parameter (Sprite Type 1)	Quad polygon vertex data 0 (0xF000 0000)
	Vertex Parameter (Sprite Type 1)	Quad polygon vertex data 1 (0xF000 0000)
	Vertex Parameter (Sprite Type 1)	Quad polygon vertex data 2 (0xF000 0000)
	Global Parameter 8 (Sprite)	Global Parameters for object-8 (0xA200 000D) Clip outside enable, Packed Color, Textured, use Offset, Flat, 16bit UV
	Vertex Parameter (Sprite Type 1)	Quad polygon vertex data 0 (0xF000 0000)
	Global Parameter 9 (Sprite)	Global Parameters for object-9 (0xA200 000D) Clip outside enable, Packed Color, Textured, use Offset, Flat, 16bit UV
	Vertex Parameter (Sprite Type 1)	Quad polygon vertex data 0 (0xF000 0000)
	Global Parameter 10 (Sprite)	Global Parameters for object-10 (0xA280 0000) Clip disable, Packed Color, Non-Textured, no Offset, Flat
	Vertex Parameter (Sprite Type 0)	Quad polygon vertex data 0 (0xF000 0000)
	Control Parameter (End Of List)	End of translucent polygon list (0x0000 0000)
Opaque Modifier Volume	Control Parameter (User Tile Clip)	User Tile clipping value set (0x2000 0000)
	Global Parameter 0 (Modifier Volume)	Global Parameters for Modifier Volume 0 (0x8182 0000) Clip inside enable, normal triangle in the volume
	Vertex Parameter 0 (Modifier Volume)	Triangle polygon vertex data 0 (0xF000 0000)
	Vertex Parameter 1 (Modifier Volume)	Triangle polygon vertex data 1 (0xF000 0000)
	Vertex Parameter 2 (Modifier Volume)	Triangle polygon vertex data 2 (0xF000 0000)
	Vertex Parameter 3 (Modifier Volume)	Triangle polygon vertex data 3 (0xF000 0000)
	Vertex Parameter 4 (Modifier Volume)	Triangle polygon vertex data 4 (0xF000 0000)
	Global Parameter 0 (Modifier Volume)	Global Parameters for Modifier Volume 0 (0x8100 0040) Clip inside enable, last triangle in the volume
	Vertex Parameter 5 (Modifier Volume)	Triangle polygon vertex data 5 (0xF000 0000)
	Control Parameter (End Of List)	End of Opaque Modifier Volume list (0x0000 0000)

### § 3.7.7 Region Array Data Configuration

The Region Array stores Pointer data that points to the starting addresses of the Object Lists when the **five** types (~~five types, in HOLLY2~~) of lists are being drawn in individual Tiles. The CPU creates the Region Array and stores it directly in texture memory.

~~In HOLLY1, The data for one Tile consists of 5 x 32-bit pieces of data, as shown below: the header word and four pointers.~~

~~In addition, the starting address that is used when the CORE reads the Region Array is specified by the REGION\_BASE register.~~

bit 31	30-14	13-8	7-2	1-0
<del>Last Region</del>	<del>Reserved</del>	<del>Tile Y position (x 32)</del>	<del>Tile X position (x 32)</del>	<del>Reserved</del>
<del>Opaque List Pointer</del>				
<del>Opaque Modifier Volume List Pointer</del>				
<del>Translucent List Pointer</del>				
<del>Translucent Modifier Volume List Pointer</del>				

#### ~~Last Region~~

~~Specifies the end of the Tiles to be drawn. This bit must be set to "1" for the last Tile.~~

#### ~~Tile Y & X~~

~~Specifies the position of the upper left corner coordinates of the Tile on the drawing screen. The actual values are the specified values multiplied by 32.~~

#### ~~List Pointer~~

~~The bit configuration of the four types of List Pointers is as shown below.~~

bit 31	30-24	23-2	1-0
<del>Empty PTR</del>	<del>Reserved</del>	<del>Pointer to Object List (32bit resolution)</del>	<del>00</del>

#### ~~Empty PTR~~

~~Set a "1" when the type of list in question does not exist. Set "1" for a polygon list that was input to the TA with "No List" specified in the TA\_ALLOC\_CTRL register. When this bit is "1," the other bits have no meaning. Therefore, setting "1" is sufficient.~~

#### ~~Pointer to Object List~~

~~Specifies the absolute address of the first Object List for that type of list (the starting address of the Object List corresponding to each Tile). Specify this value on a 32-bit boundary.~~

~~In HOLLY2~~, there are two types of data configurations for one Tile: type 1 and type 2. The type selection can be made through the FPU\_PARAM\_CFG register. ~~Just as in HOLLY1~~, the starting address for when the CORE loads the Region Array is specified by the REGION\_BASE register.

[Type 1]

bit 31	30	29	28	27-14	13-8	7-2	1-0
Last Region	Z Clear	0	Flush Accumulate	Reserved	Tile Y position ( x 32)	Tile X position ( x 32)	Reserved
Opaque List Pointer							
Opaque Modifier Volume List Pointer							
Translucent List Pointer							
Translucent Modifier Volume List Pointer							

[Type 2]

bit 31	30	29	28	27-14	13-8	7-2	1-0
Last Region	Z Clear	Pre Sort	Flush Accumulate	Reserved	Tile Y position ( x 32)	Tile X position ( x 32)	Reserved
Opaque List Pointer							
Opaque Modifier Volume List Pointer							
Translucent List Pointer							
Translucent Modifier Volume List Pointer							
Punch Through List Pointer							

**Last Region**

Specifies whether the region is the final drawing data for the screen. When drawing several times in the same Tile (multipass processing), specify "1" for this bit only in the very last drawing data for the Tile that is being drawn last on the screen.

Setting	Description of processing
0	Normal drawing data
1	End of screen drawing data

**Z Clear**

Specifies whether to clear the internal Z buffer or not before performing drawing processing for this Tile. When drawing several times in the same Tile (multipass processing), specify "0" for this bit only in the first drawing data to a given Tile.

Setting	Description of processing
0	Clear Z buffer
1	Do not clear Z buffer

**Pre Sort**

Specifies the Translucent polygon sort mode for drawing processing for this Tile. This bit is valid only in the type 2 data configuration.

Setting	Description of processing
0	Auto-sort mode
1	Pre-sort mode

**Flush Accumulate**

Specifies whether to copy the drawing results to the frame buffer or not after completing drawing processing for this Tile. When drawing several times in the same Tile (multipass processing), specify "1" for this bit only in the very last drawing data for the Tile.

Setting	Description of processing
0	Copy to the frame buffer
1	Do not copy to the frame buffer

**Tile Y & X**

Specifies the position of the upper left corner coordinates of the Tile on the drawing screen  
~~the same as HOLLY1.~~ Actual coordinates are 32 times the specified value.

### **List Pointer**

~~In HOLLY1 there are four types of lists, but in HOLLY2, with the addition of Punch Through, there are five types. The bit configuration and contents of the List Pointer are the same as in HOLLY1.~~

The bit structure of the five types of List Pointers is as follows:

Bit 31	30 to 24	23 to 2	1 and 0
Empty PTR	Reserved	Pointer to Object List (32-bit resolution)	00

### **Empty PTR**

Set a "1" when the type of list in question does not exists. Set "1" for a polygon list that was input to the TA with "No List" specified in the TA\_ALLOC\_CTRL register. When this bit is "1," the other bits have no meaning. Therefore setting "1" is sufficient.

### **Pointer to Object List**

Specifies the absolute address of the first Object List for that type of list (the starting address of the Object List corresponding to each Tile). Specify this value on a 32-bit boundary.

~~In both HOLLY1 and HOLLY2,~~ the Region Array is not generated by the TA; instead, it has to be created by the CPU and stored directly in texture memory.(refer to 3.7.1) Normally, when the TA is used to create the display list for the CORE, the Region Array data is created in the following manner:

- Only data for Tiles within the Global Tile Clipping area are stored.
- "1" is set in the Empty PTR bit for List Pointers of types that are not used on the screen. "1" should also be set for the List Pointers of Tiles that definitely do not contain a polygon in that list.
- The address values that were calculated on the basis of the Object Lists generated by the TA are set as the data for the Pointer to Object List data in the List Pointers.

### § 3.7.8 Object List Data Configuration

Object lists contain pointers (for the starting addresses of the ISP/TSP Parameters) for the objects that are included in each Tile. Object lists consist of a grouping of Object Pointer Blocks of a size specified by the TA\_ALLOC\_CTRL register. The TA automatically creates Object Lists from the polygon data that was input, and stores them in texture memory. An Object List consists of 32-bits per object, and includes the following four types:

When the list is generated by the TA, this data is generated automatically on the basis of the parameters that were input to the TA.

**Triangle Strip:** Used for Triangle polygons in a strip.

Bit 31	30-25						24	23-21	20-0
O	Mask						Shadow	Skip	Triangle Strip Start (32bit word address)
	T0	T1	T2	T3	T4	T5			

**Triangle Array:** Used for an independent Triangle polygon.

Bit 31-29	28-25	24	23-21	20-0
100	Number of Triangles	Shadow	Skip	Triangle Array Start (32bit word address)

**Quad Array:** Used for an independent Quad polygon.

Bit 31-29	28-25	24	23-21	20-0
101	Number of Quads	Shadow	Skip	Quad Array Start (32bit word address)

**Object Pointer Block Link:** Used when linking to an Object Pointer Block, and at the end of a list.

bit 31-29	28	27-24	23-2	1-0
111	End of List	Reserved	Next Pointer Block (32bit word address)	00

#### **Mask**

Set only those bits that correspond to those stripped Triangle polygons that are included in the Tile in question. When the list is generated by the TA, this determination is made by the TA for each polygon, and this bit is set automatically.

#### **Shadow**

This bit is set to "1" for objects for which the TSP parameters are switched, depending on whether the object is inside or outside of a Modifier Volume. In Intensity Shadow Mode, this bit specifies whether shadow processing is performed or not. When the list is generated by the TA, the value of the Shadow bit in the Parameter Control Word is automatically set.

#### **Skip**

Specifies the data size (× 32 bits) for one vertex in the ISP/TSP Parameters. Normally, the actual data size is "Skip + 3," but if Parameter Selection Volume Mode is enabled and the Shadow bit described above is set to "1," the actual data size is "Skip × 2 + 3." When the list is generated by the TA, the value shown in the table below is set automatically, based on the value of the Parameter Control Word in the TA parameters.

Parameter Control Word			Skip value
Texture	Offset	16bit_UV	
0	invalid	invalid	001
1	0	0	011
1	0	1	010
1	1	0	100
1	1	1	011

#### **Number of Triangles/Quads**

When the ISP/TSP Parameters for polygon data that is included within the same Tile is stored contiguously in texture memory, this field specifies the number that are contiguous. This value is "0" in the case of only one (noncontiguous) polygon, and "1" in

the case of two contiguous polygons. Normally, this field is only used with independent Triangle polygons or Quad polygons. When the list is generated by the TA, this value is set automatically.

***End of List***

Set this bit to "1" at the end of the Object List data for the Tile in question. When the list is generated by the TA, this value is set automatically according to the End Of List parameter.

***Triangle Strip Start***

***Triangle/Quad Array Start***

Specifies the starting address of the object data (the ISP/TSP Parameters) at a 32-bit boundary. The actual address in texture memory is derived by adding this value to the value in the PARAM\_BASE register. When the list is generated by the TA, this value is set automatically.

***Next Pointer Block***

Specifies the starting address of the next Object Pointer Block at a 32-bit boundary. When the list is generated by the TA, this value is set automatically.

### § 3.7.9 ISP/TSP Parameter Data Configuration

The ISP/TSP Parameters consist of the ISP/TSP Instruction Word, the TSP Instruction Word, the Texture Control Word, and, for strips with a strip number of 1 to 6, vertex coordinates, Texture UV coordinates, and Shading Colors (Base, Offset).

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex X
Vertex Y
Vertex Z
Texture U
Texture V
Base Color
Offset Color

For triangle strips, the gray area is repeated up to 7 times.

For a polygon with two volumes, the Texture UV and the Shading Color are both needed for each vertex.

#### **Vertex X, Y, Z**

The vertex coordinates are IEEE single-precision floating point values. Set the screen coordinates for X and Y, and either 1/z or 1/w for Z. The Z value for the fourth vertex of a Quad polygon does not need to be specified because it is generated in the CORE. When the list is generated by the TA, the Vertex Parameter values are set automatically.

#### **Texture U, V**

If a texture is used on a polygon, the UV coordinates of the texture have to be specified. These coordinates can be specified in two formats. If the "16-bit UV" bit in the ISP/TSP Instruction Word is "0," set two IEEE single-precision floating point values. If the "16-bit UV" bit is "1," extract the upper 16 bits of each of the 32-bit floating point values U and V, and set them as one data item, with U as the upper 16 bits and V as the lower 16 bits. When the list is generated by the TA, the Vertex Parameter values are set automatically.

#### **Base Color, Offset Color**

Shading Color data includes Base Color, Offset color, and Bump Map parameters. If a polygon does not use a texture, or if the Offset bit (in the ISP/TSP Instruction Word) is "0," the Offset Color is not needed. In addition, the Bump Map parameters (K1K2K3Q) are stored instead of the Offset Color when a Bump Map texture is used, and the data is valid for the third and subsequent vertices.

In the case of a Flat-Shaded polygon, the data is valid for the third and subsequent vertices. The alpha value of the Base/Offset Color is valid only when the "Use Alpha" bit (in the TSP Instruction Word) is "1." An alpha value of "0x00" indicates that the polygon is completely transparent, while an alpha value of "0xFF" indicates that the polygon is completely opaque. In addition, the alpha value of the Offset Color is used as the Fog coefficient when Fog Control (in the TSP Instruction Word) is set to Per Vertex mode. The values for the fourth vertex in a Quad polygon do not need to be specified because they are generated within the CORE. When the list is generated by the TA, the Vertex Parameter values are set automatically.

#### **Base/Offset Color (Packed Color)**

bit 31-24	23-16	15-8	7-0
Alpha	Red	Green	Blue

#### **Bump Map Parameter**

bit 31-24	23-16	15-8	7-0
K1	K2	K3	Q

Examples of ISP/TSP Parameters for various types of polygons are shown below.

**Single Triangle Polygon**

(Textured, Gouraud, 32bit UV,  
use Offset)

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex A :Vertex X
Vertex A :Vertex Y
Vertex A :Vertex Z
Vertex A :Texture U
Vertex A :Texture V
Vertex A :Base Color
Vertex A :Offset Color
Vertex B :Vertex X
Vertex B :Vertex Y
Vertex B :Vertex Z
Vertex B :Texture U
Vertex B :Texture V
Vertex B :Base Color
Vertex B :Offset Color
Vertex C :Vertex X
Vertex C :Vertex Y
Vertex C :Vertex Z
Vertex C :Texture U
Vertex C :Texture V
Vertex C :Base Color
Vertex C :Offset Color

**Single Quad Polygon**

(Textured, Flat, 16bit UV,  
no Offset)

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex A :Vertex X
Vertex A :Vertex Y
Vertex A :Vertex Z
Vertex A :Texture U/V
Vertex A :Base Color (invalid)
Vertex B :Vertex X
Vertex B :Vertex Y
Vertex B :Vertex Z
Vertex B :Texture U/V
Vertex B :Base Color (invalid)
Vertex C :Vertex X
Vertex C :Vertex Y
Vertex C :Vertex Z
Vertex C :Texture U/V
Vertex C :Base Color
Vertex D :Vertex X
Vertex D :Vertex Y
Vertex D :Vertex Z (invalid)
Vertex D :Texture U/V (invalid)
Vertex D :Base Color (invalid)

**2 Stripped Triangle Polygon**

(Textured, Gouraud, 16bit UV,  
use Offset, with Two Volumes)

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex A :Vertex X
Vertex A :Vertex Y
Vertex A :Vertex Z
Vertex A :Texture U0/V0
Vertex A :Base Color 0
Vertex A :Offset Color 0
Vertex A :Texture U1/V1
Vertex A :Base Color 1
Vertex A :Offset Color 1
Vertex B :Vertex X
Vertex B :Vertex Y
Vertex B :Vertex Z
Vertex B :Texture U0/V0
Vertex B :Base Color 0
Vertex B :Offset Color 0
Vertex B :Texture U1/V1
Vertex B :Base Color 1
Vertex B :Offset Color 1
Vertex C :Vertex X
Vertex C :Vertex Y
Vertex C :Vertex Z
Vertex C :Texture U0/V0
Vertex C :Base Color 0
Vertex C :Offset Color 0
Vertex C :Texture U1/V1
Vertex C :Base Color 1
Vertex C :Offset Color 1
Vertex D :Vertex X
Vertex D :Vertex Y
Vertex D :Vertex Z
Vertex D :Texture U0/V0
Vertex D :Base Color 0
Vertex D :Offset Color 0
Vertex D :Texture U1/V1
Vertex D :Base Color 1
Vertex D :Offset Color 1

**2 Stripped Triangle Polygon  
(Non-Textured, Gouraud)**

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word (invalid)
Vertex A :Vertex X
Vertex A :Vertex Y
Vertex A :Vertex Z
Vertex A :Base Color
Vertex B :Vertex X
Vertex B :Vertex Y
Vertex B :Vertex Z
Vertex B :Base Color
Vertex C :Vertex X
Vertex C :Vertex Y
Vertex C :Vertex Z
Vertex C :Base Color
Vertex D :Vertex X
Vertex D :Vertex Y
Vertex D :Vertex Z
Vertex D :Base Color

**Single Triangle Polygon  
(Bump Mapped, Flat, 16bit UV)**

ISP/TSP Instruction Word
TSP Instruction Word
Texture Control Word
Vertex A :Vertex X
Vertex A :Vertex Y
Vertex A :Vertex Z
Vertex A :Texture U/V
Vertex A :Base Color (invalid)
Vertex A :K1K2K3Q (invalid)
Vertex B :Vertex X
Vertex B :Vertex Y
Vertex B :Vertex Z
Vertex B :Texture U/V
Vertex B :Base Color (invalid)
Vertex B :K1K2K3Q (invalid)
Vertex C :Vertex X
Vertex C :Vertex Y
Vertex C :Vertex Z
Vertex C :Texture U/V
Vertex C :Base Color
Vertex C :K1K2K3Q

Fig. 3-79



### § 3.7.9.1 ISP/TSP Instruction Word

When the list is generated by the TA, the ISP/TSP Instruction Word in the Global Parameters is used for the ISP/TSP Instruction Word that is stored in texture memory. However, bits 3 through 0 (Texture/Offset/Gouraud/16bit\_UV) of the Parameter Control Word in the Global Parameters are used for bits 25 through 22 of the ISP/TSP Instruction Word (Texture/Offset/Gouraud shading/16bit\_UV).

#### Opaque or Translucent

bit 31-29	28-27	26	25	24	23	22	21	20	19-0
Depth Compare mode	Culling Mode	Z Write Disable	Texture	Offset	Gouraud shading	16Bit UV	Cache Bypass	Dcalc Ctrl	Reserved

#### Opaque Modifier Volume or Translucent Modifier Volume

bit 31-29	28-27	26-0
Volume Instruction	Culling Mode	Reserved

#### Depth Compare Mode

This bit is used in combination with the Z Write Disable bit, and supports compare processing, which is required for OpenGL and D3D versus Z buffer updates. It is important to note that, because the value of either 1/z or 1/w is referenced for the Z value, the closer that the polygon is, the larger that the Z value will be.

This setting is ignored for Translucent polygons in Auto-sort mode; the comparison must be made on a "Greater or Equal" basis. This setting is also ignored for Punch Through polygons in HOLLY; the comparison must be made on a "Less or Equal" basis.

Setting	Depth Function
0	Never
1	Less
2	Equal
3	Less Or Equal
4	Greater
5	Not Equal
6	Greater Or Equal
7	Always

#### Culling Mode

This specifies the back-face culling mode. The "No Culling" specification means that culling is not performed. The value that is specified in the FPU\_CULL\_VAL register is used in the remaining three specifications.

Setting	Culling Mode	Processing
0	No culling	no culling
1	Cull if Small	Cull if $( det  < fpu\_cull\_val)$
2	Cull if Negative	Cull if $( det  < 0)$ or $( det  < fpu\_cull\_val)$
3	Cull if Positive	Cull if $( det  > 0)$ or $( det  < fpu\_cull\_val)$

This specification eliminates extremely small Triangle and Quad polygons, in addition to normal back-face culling. The FPU\_CULL\_VAL register requires at least a 4-bit mantissa (plus an 8-bit index), and the value must be positive. The "det" value is a Triangle Adjoint Matrix, and is equal to the screen area for the Triangle.

The adjoint matrix is derived in the manner described below.

Transform a plane equation  $ax + by + c = d$  into  $Ax + By + 1 = C$ , and find the values A, B, and C that simultaneously satisfy the three vertices  $(x_0, y_0, z_0)$ ,  $(x_1, y_1, z_1)$ , and  $(x_2, y_2, z_2)$  that were given. First, from the three vertices we can derive the following matrix:

$$(A, B, C) \begin{pmatrix} x0 & x1 & x2 \\ y0 & y1 & y2 \\ 1 & 1 & 1 \end{pmatrix} = (z0, z1, z2)$$

Solving this inverse matrix yields the values A, B, and C that express the plane that passes through the three vertices. That result is:

$$(A, B, C) = (z0, z1, z2) \frac{1}{\Delta} \text{Adj}$$

Therefore:

$$\text{Adj} = \begin{pmatrix} y1 - y2 & x2 - x1 & x1y2 - x2y1 \\ y2 - y0 & x0 - x2 & x2y0 - x0y2 \\ y0 - y1 & x1 - x0 & x0y1 - x1y0 \end{pmatrix}$$

$$\Delta = x0(y1 - y2) + x1(y2 - y0) + x2(y0 - y1)$$

This  $\Delta$  becomes "det." (In the case of a homogenous coordinate system (the screen coordinates), the above values are all multiplied by  $1/W$ .)

### **Z Write Disable**

If the Z Write Disable bit is set to "1," the Z value comparison is executed normally and the image is drawn, but the Z value is not updated, even if the polygon is visible, for example. This is used for OpenGL depth masking.

### **Texture**

This specifies whether a texture is to be used on a polygon or not. Set "1" if a texture is to be used. When the list is generated by the TA, the Texture bit in the Parameter Control Word is automatically set here.

### **Offset**

This specifies whether an Offset Color is to be used or not. When this bit is set to "1," the offset value is added to the shading calculation; if this bit is "0," an offset value of "0" is added to the calculation. In the case of Gouraud shading, the offset value is interpolated between vertices. When the list is generated by the TA, the Offset bit in the Parameter Control Word is automatically set here.

In the case of a Bump Mapped polygon, this setting must be set to use Offset Color. Set the Bump Map parameters (K1K2K3Q) instead of the Offset Color value.

### **Gouraud shading**

This specifies the type of shading. If this bit is set to "1," Gouraud shading is used, in which each of the vertex colors is interpolated according to the perspective. If this bit is set to "0," Flat Shading is used with the color from the third vertex. Note that the amount of data stored in memory is the same in either case. When the list is generated by the TA, the Gouraud bit in the Parameter Control Word is automatically set here.

Note that the amount of data for the ISP/TSP Parameters that is stored in texture memory is the same, whether Flat Shading is specified or Gouraud shading is specified. When Flat Shading is specified, the Base Color and Offset Color from the third vertex are valid.

In the case of a Bump Mapped polygon, Flat Shading must be specified.

### **16Bit UV**

This specifies the number of bits that are used for the Texture UV values. If the Texture bit is "1" and this bit is "1," a pair of UV values is set as a single 32-bit data item by discarding the lower 16 bits of the 32-bit floating point values and combining the remaining bits. When the list is generated by the TA, the 16bit\_UV bit in the Parameter Control Word is automatically set here.

When the UV value is 16 bits, the bit configuration is as shown below.

bit 31-16	15-0
-----------	------

16bit U	16bit V
---------	---------

### **Cache Bypass**

This bit specifies whether or not to use the TSP parameter cache; if the cache is not to be used, set this bit to "1." Because the TSP parameters for polygons that are not included in any Tiles except for just one are only used once, there is no need to store those parameters in the cache. Therefore, this bit is used to prevent performance from suffering due to such TSP parameters being needlessly stored in the cache. When the list is generated by the TA, this bit is set automatically.

### **Dcalc Ctrl**

This bit specifies the precision of the calculations that are performed when calculating the D value that is used as an indicator of when to change textures in MIPMAP processing.

When this bit is set to "0," the calculations are as shown below. "a," "b," "c," "d," "e," "f," "p," "q," and "r" are texture mapping coefficients, and "X" and "Y" are screen coordinates.

$$dudx = \frac{ar - pc}{(pX + qY + r)^2}$$

$$dvdx = \frac{dr - pf}{(pX + qY + r)^2}$$

$$dudy = \frac{br - qc}{(pX + qY + r)^2}$$

$$dvdy = \frac{er - qf}{(pX + qY + r)^2}$$

When this bit is set to "1," the calculations are as shown below. "X'" and "Y'" are the screen coordinates for the first vertex.

$$dudx = \frac{a(pX' + qY' + r) - p(aX' + bY' + c)}{(pX + qY + r)^2}$$

$$dvdx = \frac{d(pX' + qY' + r) - p(dX' + eY' + f)}{(pX + qY + r)^2}$$

$$dudy = \frac{b(pX' + qY' + r) - q(aX' + bY' + c)}{(pX + qY + r)^2}$$

$$dvdy = \frac{e(pX' + qY' + r) - q(dX' + eY' + f)}{(pX + qY + r)^2}$$

Setting this bit to "1" results in more precise calculations for small polygons, but the calculation time (drawing time) is longer.

### **Volume Instruction**

The CORE supports inclusion and exclusion volumes that are formed by Triangle polygons, and inclusion and exclusion volumes that are formed by Quad polygons. It is necessary, for each object, to specify the type and the last of the polygons that form a volume. However, the TA only supports volumes that are formed by Triangle polygons.

Because the Z comparison must be consistent for polygons that form a volume, the bit that is used to specify the Depth Compare Mode in data for a normal object is used to specify the Volume Instruction in data for a Modifier Volume object.

Setting	Volume Instruction
---------	--------------------

0	'Normal' Polygon
1	Inside Last Polygon
2	Outside Last Polygon
3-7	Reserved

"Normal Polygon" indicates a polygon other than the last polygon that forms the volume. "Inside Last Polygon" indicates the last polygon that forms an inclusion volume. "Outside Last Polygon" indicates the last polygon that forms an exclusion volume. When the list is generated by the TA, this bit must be set properly by the CPU.

The Culling Mode bit is the same as for data for a normal object, but if any but the very smallest polygons are culled, the display image may not appear as expected.

### § 3.7.9.2 TSP Instruction Word

The TSP Instruction Word that was input to the TA in the Global Parameters is used as is for the TSP Instruction Word that is stored in texture memory.

bit 31-29	28-26	25	24	23-22	21	20	19	18-17
SRC Alpha Instr	DST Alpha Instr	SRC Select	DST Select	Fog Control	Color Clamp	Use Alpha	Ignore Tex. Alpha	Flip UV

bit 16-15	14-13	12	11-8	7-6	5-3	2-0
Clamp UV	Filter Mode	Super-Sample Texture	MIP-Map 'D' adjust	Texture/Shading 'Instruction'	Texture U Size	Texture V Size

#### ***SRC/DST Alpha Instruction***

Accumulation buffer control is performed through two 3-bit values (SRC Alpha Instr and DST Alpha Instr) and two 1-bit values (SRC Select and DST Select). The first two 3-bit values specify the  $\alpha$  blending function for the source and the destination, respectively, and the two 1-bit values specify the source and the destination. Used in combination, these bits can support all OpenGL and D3D texture blending functions.

The SRC Select bit and the DST Select bit specify whether or not to use the secondary accumulation buffer that can be used for polygons that have multiple textures, such as Bump Mapped polygons. This type of special effect can be implemented by using the secondary accumulation buffer while repeating opaque operations in a different texture/shading mode.

With the blending function, the RGBA values of the SRC and the DST are combined and then written back to the DST. A mathematical representation of the write-back function is shown below.

$$\text{DST} := \text{SRC} \times \text{BlendFunction}(\text{SRC Alpha Instruction}) + \text{DST} \times \text{BlendFunction}(\text{DST Alpha Instruction})$$

In this equation, "BlendFunction(Instruction)" returns the RGBA coefficient that was calculated from the SRC and DST color data in accordance with the 3-bit instructions that were specified in the SRC Alpha Instr and the DST Alpha Instr. The instruction codes are listed below.

Setting	Instruction	Coefficient
0	Zero	(0, 0, 0, 0)
1	One	(1, 1, 1, 1)
2	'Other' Color	(O <sub>R</sub> , O <sub>G</sub> , O <sub>B</sub> , O <sub>A</sub> )
3	Inverse 'Other' Color	(1-O <sub>R</sub> , 1-O <sub>G</sub> , 1-O <sub>B</sub> , 1-O <sub>A</sub> )
4	SRC Alpha	(S <sub>A</sub> , S <sub>A</sub> , S <sub>A</sub> , S <sub>A</sub> )
5	Inverse SRC Alpha	(1-S <sub>A</sub> , 1-S <sub>A</sub> , 1-S <sub>A</sub> , 1-S <sub>A</sub> )
6	DST Alpha	(D <sub>A</sub> , D <sub>A</sub> , D <sub>A</sub> , D <sub>A</sub> )
7	Inverse DST Alpha	(1-D <sub>A</sub> , 1-D <sub>A</sub> , 1-D <sub>A</sub> , 1-D <sub>A</sub> )

"Other Color" and "Inverse Other color" mean that the DST color is to be used when specified for an SRC instruction, and that the SRC color data is to be used when specified for a DST instruction.

After the coefficients have been determined and have been multiplied with SRC/DST, respectively, the addition operation is selected. At this point, check for overflows and clamp the derived value as appropriate.

In the case of an Opaque polygon, "1" must be specified in the SRC instruction and "0" must be specified in the DST instruction.

In the case of a ~~HOLLY2~~ Punch Through polygon, "4" (SRC Alpha) must be specified in the SRC instruction and "5" (Inverse SRC Alpha) must be specified in the DST instruction.

### ***SRC/DST Select***

These two bits select the source/destination data for the blending function.

When the SRC Select bit is "1," the contents of the secondary accumulation buffer are used as the source data instead of the color data that resulted from the shading/texturing calculations for the current polygon. The value in the Secondary Accumulation Buffer is also used for the pixel alpha value. This function is used to blend the result (that was stored in the secondary accumulation buffer) of operations performed on multiple textures with the current polygon color data and then return the new result to the primary accumulation buffer.

When the DST Select bit is "1," the contents of the secondary accumulation buffer are used as the destination data instead of the Shading/Texturing color data that is stored in the primary accumulation buffer. This function is used to blend the current polygon color data with the contents of the secondary accumulation buffer and then return the results to the secondary accumulation buffer. This specification applies to all DST data resulting from the blending calculations described earlier.

### ***Fog Control***

There are separate Fog Color registers for Look Up Table mode and Per Vertex mode (the FOG\_COL\_RAM register and the FOG\_COL\_VERT register).

Setting	Fog Mode	Explanation
00	Look Up Table	Generates the Fog $\alpha$ value through linear interpolation of the table data corresponding to the depth value.
01	Per Vertex	Uses the Offset Color $\alpha$ value for the Fog $\alpha$ value. This value is interpolated if the Gouraud bit is set to "1," and is constant if Flat Shading is specified. If the Offset bit is not set to "1," "No Fog" mode is in effect.
10	No Fog	Fog processing is not performed.
11	Look Up Table Mode 2	Substitutes the polygon color for the Fog Color, and the polygon $\alpha$ value for the Fog $\alpha$ value.

### ***Color Clamp***

Color clamp processing is performed before Fog processing. There are two registers, one for underflows and one for overflows (the FOG\_CLAMP\_MIN register and the FOG\_CLAMP\_MAX register).

### ***Use Alpha***

When this bit is "1," the  $\alpha$  value in the vertex's Shading Color data is valid; if Gouraud Shading is in effect, then values between vertices are interpolated. If this bit is "0," the polygon is regarded to be completely opaque ( $\alpha$  value = 1.0).

### ***Ignore Texture Alpha***

If a texture has an  $\alpha$  value, this bit can be used to ignore that  $\alpha$  value. When this bit is "1," the  $\alpha$  value of the texture is regarded to be completely opaque ( $\alpha$  value = 1.0). This bit is valid only in regards to the  $\alpha$  value of textures.

### **Flip UV**

These bits specify whether or not to use the flip function for each individual texture size, in either the U direction or the V direction. Bit 18 is used for the U direction, and bit 17 is used for the V direction; if the bit in question is "1," the texture flips in that direction.

#### **Example: When flipped in the U direction**

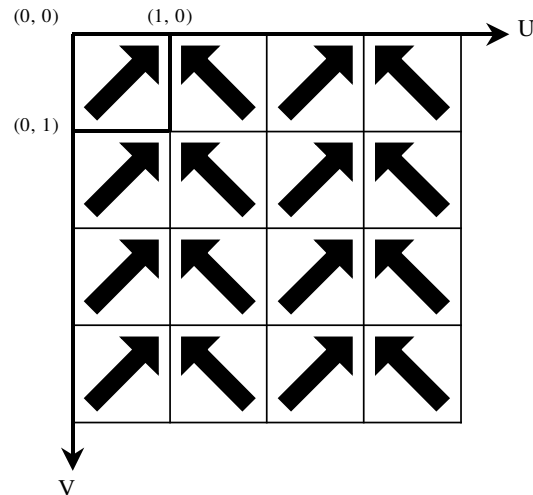


Fig. 3-80

### **Clamp UV**

These bits specify whether or not to use the clamp function in either the U direction or the V direction. The clamp function is used if the bit is "1." If the clamp function is enabled, the flip function is disabled.

### **Filter Mode**

These bits specify the mode of the texture filter function.

Setting	Filter mode
00	Point Sampled
01	Bilinear Filter
10	Tri-linear Pass A
11	Tri-linear Pass B

The tri-linear filter consists of two processes; one object results from Pass A and Pass B. In order to implement the Tri-linear filter, the CPU needs to register the object twice (once as an object with the "Pass A" specification, and once as an object with the "Pass B" specification). The Tri-linear filter is valid only when MIPMAP is specified for a texture, but in other Filter Modes, textures for which MIPMAP is not specified can also be used. In the case of a texture for which MIPMAP is specified, the CORE automatically selects the appropriate MIPMAP level.

If an attempt is made to apply tri-linear filter processing to an object for which the "MIP Mapped" bit is not set to "1," the object is processed in "Point Sampled" mode.

### **Super-Sample texture**

If this bit is set to "1," the quality of the texture filter function is enhanced through super-sampling. However, drawing time is extended considerably. If MIPMAP is being used, the CORE automatically selects the MIPMAP level with the next highest resolution.

### **MIP-map D Adjust**

The D value used for MIPMAP is calculated internally by the CORE, but fine adjustments are necessary when forcibly enlarging or reducing the image in order to find where aliasing and blurring occur. The D value calculated by the core is multiplied by the specified adjustment value. This value is a 4-bit unsigned fixed decimal value, with a 2-bit decimal portion. Example values are shown in the table below.

Example setting	Actual value
00.00	<i>Illegal</i>
00.01	0.25
01.00	1.0
11.11	3.75

<Note> Because a value of "0.0" is invalid for D, it must not be specified.

The D value is determined according to the following equation.

$$D^2 = \text{Max} \left[ \frac{du^2}{dx} + \frac{dv^2}{dx}, \frac{du^2}{dy} + \frac{dv^2}{dy} \right]$$

For example, if the D value is "1," a full-resolution texture map is used. If the D value is "2," a half-resolution texture map is used.

### **Texture/Shading Instruction**

This determines the method for combining the Shading Color values (Base Colors  $\alpha$ , and Offset Colors) interpolated between vertices with texture  $\alpha$  values and texture color values. However, this setting is invalid for Non-Textured polygons.

Setting	Mode	Explanation
0	Decal	$PIX_{RGB} = TEX_{RGB} + OFFSET_{RGB}$ $PIX_A = TEX_A$
1	Modulate	The texture color value is multiplied by the Shading Color value. The texture $\alpha$ value is substituted for the Shading $\alpha$ value. $PIX_{RGB} = COL_{RGB} \times TEX_{RGB} + OFFSET_{RGB}$ $PIX_A = TEX_A$
2	Decal Alpha	The texture color value is blended with the Shading Color value according to the texture $\alpha$ value. $PIX_{RGB} = (TEX_{RGB} \times TEX_A) + (COL_{RGB} \times (1 - TEX_A)) + OFFSET_{RGB}$ $PIX_A = COL_A$
3	Modulate Alpha	The texture color value is multiplied by the Shading Color value. The texture $\alpha$ value is multiplied by the Shading $\alpha$ value. $PIX_{RGB} = COL_{RGB} \times TEX_{RGB} + OFFSET_{RGB}$ $PIX_A = COL_A \times TEX_A$

~~In HOLLYe~~ In the case of a Punch Through polygon, only those pixels for which the shading alpha value (PIX[A]) that results from this instruction is 1.0 (0xFF) are drawn.



### ***U Size***

This specifies the U size of the texture.

Setting	Size (texels)
0	8
1	16
2	32
3	64
4	128
5	256
6	512
7	1024

In the case of a stride texture (where Scan Order = 1 and Stride Select = 1 in the Texture Control Word), the texture U size is specified by the stride value (bits 4 through 0) in the TEXT\_CONTROL register. However, this U size value is the value that is used for calculating the texture coordinates. The value that is specified here must be greater than the U size of the stride texture.

<Example: Polygon that uses a 320 x 240 stride texture>

- TEXT\_CONTROL register: stride = 0xA
- TSP Instruction Word: U Size = 0x6, V Size = 0x5(512 × 256)
- 2 stripped polygon UV coordinates: [Vertex 1] U = 0.0, V = 240/256  
[Vertex 2] U = 0.0, V = 0.0  
[Vertex 3] U = 320/512, V = 240/256  
[Vertex 4] U = 320/512, V = 0.0

### ***V Size***

This specifies the V size of the texture. However, this value is ignored and the V size becomes the same as the U size if the Scan Order bit is "0" and the MIP mapped bit is "1." The correspondence between the settings and the actual size is the same as shown for U Size above.

### § 3.7.9.3 Texture Control Word

The Texture Control Word in the Global Parameters that were input to the TA are used as is for the Texture Control Word that is stored in texture memory.

#### RGB/YUV Texture or Bump Map

bit 31	30	29-27	26	25	24-21	20-0
MIP Mapped	VQ Compressed	Pixel Format	Scan Order	Stride Select	Reserved	Texture Address (64bit word address)

#### Palette Texture

bit 31	30	29-27	26-21	20-0
MIP Mapped	VQ Compressed	Pixel Format	Palette Selector	Texture Address (64bit word address)

#### **MIP Mapped**

If the texture is being MIP-mapped, set this bit to "1." This bit is valid only when the Scan Order bit is "0."

#### **VQ Compressed**

If the texture is a VQ texture, set this bit to "1." A VQ texture is a texture that has been compressed using a code book with 256 codes that correspond to  $2 \times 2$  pixels.

In the case of a palette texture, the texture is compressed using a code book that corresponds to  $2 \times 4$  pixels (8BPP) or  $4 \times 4$  pixels (4BPP).

#### **Pixel Format**

This specifies the pixel format of the texture.

Setting	Pixel Format	Description
0	1555	$\alpha$ value: 1 bit; RGB values: 5 bits each
1	565	R value: 5 bits; G value: 6 bits; B value: 5 bits
2	4444	$\alpha$ value: 4 bits; RGB values: 4 bits each
3	YUV422	32 bits per 2 pixels; YUYV values: 8 bits each
4	Bump Map	16 bits/pixel; S value: 8 bits; R value: 8 bits
5	4 BPP Palette	Palette texture with 4 bits/pixel
6	8 BPP Palette	Palette texture with 8 bits/pixel
7	Reserved	Regarded as 1555

If the Filter Mode is a mode other than Point Sampling, twice as much processing time per pixel is required for a YUV422 texture.

#### **Scan Order**

Set this bit to "1" when the texture is Non-Twiddled format. When this bit is set to "0," the texture is Twiddled format. When this bit is "1," the MIP Mapped bit is ignored. Using Non-Twiddled format textures results in lower processing performance than when Twiddled format is used.

#### **Stride Select**

When this bit is "1," the U size of the texture is specified by the TEXT\_CONTROL register (i.e., the U Size bit is ignored). The U size is then the value in the register multiplied by 32. This bit is valid only when the Scan Order bit is "1."

### ***Texture Address***

This address value, given in units of 64-bits, is the starting address for the texture data. In the case of a VQ texture, this specifies the starting address of the Code Book.

### ***Palette Selector***

This specifies the palette number. The actual palette address is the upper portion of the texture value (4 bits or 8 bits) with this value appended. In 8BPP Palette mode, however, only the upper two bits are valid.

#### **4BPP Palette**

bit 9-4	3-0
Palette Selector (bit 26-21)	Texture data (4 bits)

#### **8BPP Palette**

bit 9-8	7-0
Palette Selector (bit 26-25)	Texture data (8 bits)

The color format for a palette format texture is specified by the PAL\_RAM\_CTRL register, and can be selected from among four types: 1555, 565, 4444, and 8888. When the color format is 8888 mode, texture filtering performance is reduced by half.

## § 3.8 Details on Miscellaneous Functions

### § 3.8.1 YUV-data Converter

The Tile Accelerator has a YUV-data converter that converts YUV420- or YUV422-format data into YUV422-format texture data in macro block units ( $16 \text{ pixels} \times 16 \text{ pixels}$ ), and then stores the data in texture memory. DMA transfers from system memory to the TA are performed one macro block of YUV data at a time. The transfer order of each type of data must be as shown below.

#### For YUV420 format

- (1) Transfer U data for  $16 \times 16$  pixels (64 bytes).
- (2) Transfer V data for  $16 \times 16$  pixels (64 bytes).
- (3) Transfer Y data for  $16 \times 16$  pixels (256 bytes).

The U and V data are each stored in the internal buffer one time; once half of the Y data has been loaded into the internal buffer, transfer to texture memory begins. Once texture data for  $16 \times 8$  pixels has been output, the other half of the Y data is loaded into the internal buffer and then the texture data for the remaining  $16 \times 8$  pixels is transferred to texture memory.

#### For YUV422 format

- (1) Transfer U data for  $16 \times 8$  pixels (64 bytes).
- (2) Transfer V data for  $16 \times 8$  pixels (64 bytes).
- (3) Transfer Y data for  $16 \times 8$  pixels (128 bytes).
- (4) Transfer remaining U data for  $16 \times 8$  pixels (64 bytes).
- (5) Transfer remaining V data for  $16 \times 8$  pixels (64 bytes).
- (6) Transfer remaining Y data for  $16 \times 8$  pixels (128 bytes).

The U and V data are each stored in the internal buffer one time; once the Y data has been loaded into the internal buffer, transfer to texture memory begins. Once texture data for  $16 \times 8$  pixels has been output, the other half of the U, V, and Y data is loaded into the internal buffer and then the texture data for the remaining  $16 \times 8$  pixels is transferred to texture memory.

If the TA\_YUV\_TEX\_BASE register is written to, the YUV-data Converter initializes the address where data is stored in texture memory to the value stored in the TA\_YUV\_TEX\_BASE register, and then begins operating, assuming the first data that is input to be U data.

Once the number of macro blocks of texture data specified by YUV\_U\_Size and YUV\_V\_Size in the TA\_YUV\_TEX\_CTRL register have been stored in texture memory, the YUV Data Converter outputs an interrupt and then automatically resets the storage address to the value stored in the TA\_YUV\_TEX\_BASE register. Note that the texture data is transferred to texture memory in Non-Twiddled format.

The order in which YUV data is stored in system memory is shown below. The data is transferred consecutively through DMA transfer.

**< For YUV420 format >**

- (1) U-data(16 × 16pixel) (2) V-data(16 × 16pixel)  
(3) Yo-data(8 × 8pixel) (4) Y1-data(8 × 8pixel) (5) Y2-data(8 × 8pixel) (6) Y3-data(8 × 8pixel)

**< For YUV422 format >**

- (1) Uo-data(16 × 8pixel) (2) Vo-data(16 × 8pixel) (3) Yo-data(8 × 8pixel) (4) Y1-data(8 × 8pixel)  
(5) U1-data(16 × 8pixel) (6) V1-data(16 × 8pixel) (7) Y2-data(8 × 8pixel) (8) Y3-data(8 × 8pixel)

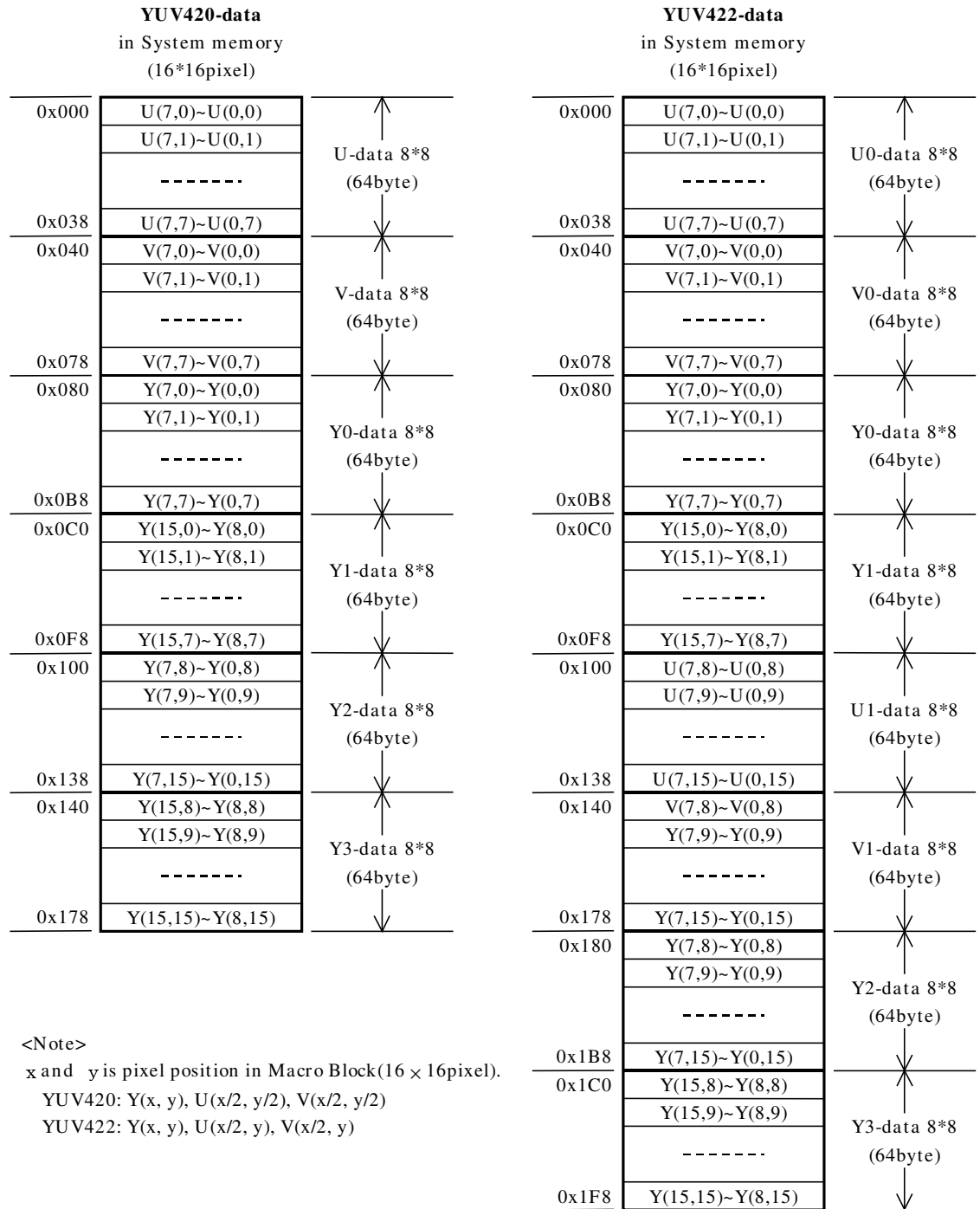


Fig. 3-81

DMA transfer of macro blocks from system memory is performed in the order shown below,

starting from the upper left corner of the screen and continuing for the amount specified by YUV\_U\_Size and YUV\_V\_Size in the TA\_YUV\_TEX\_CTRL register.

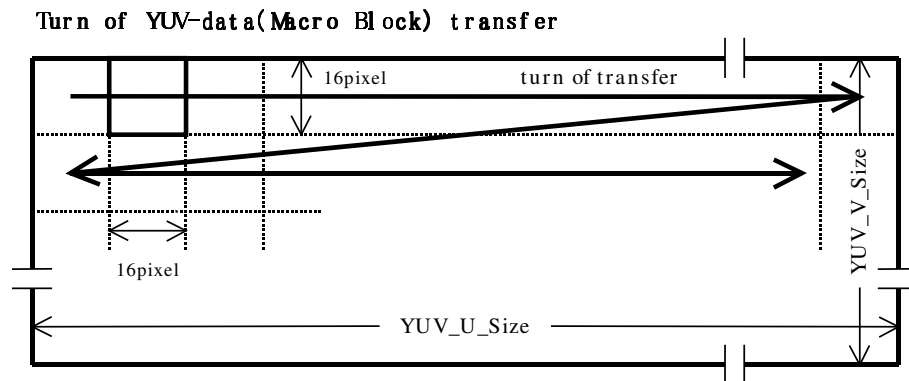


Fig. 3-82

The YUV data that is input is stored in  $16 \times 16$ -pixel units in texture memory by the method specified in the TA\_YUV\_TEX\_CTRL register.

**< YUV\_Tex = 0:>**

The YUV data that is input is stored in texture memory as one texture with a size of  $[(YUV\_U\_Size + 1) \times 16]$  (H)  $\times$   $[(YUV\_V\_Size + 1) \times 16]$  (V). This format has a weakness in that storage time is longer because the storage addresses in texture memory will not be continuous every 16 pixels (32 bytes) in the horizontal direction.

**< When YUV\_Tex = 1:>**

$[(YUV\_U\_Size + 1) \times (YUV\_V\_Size + 1)]$  textures of the macro size ( $16 \times 16$  pixels) are stored in texture memory. Storage time is shorter, because the storage addresses in texture memory are continuous. However, each texture must be used with a different polygon and arranged on screen.

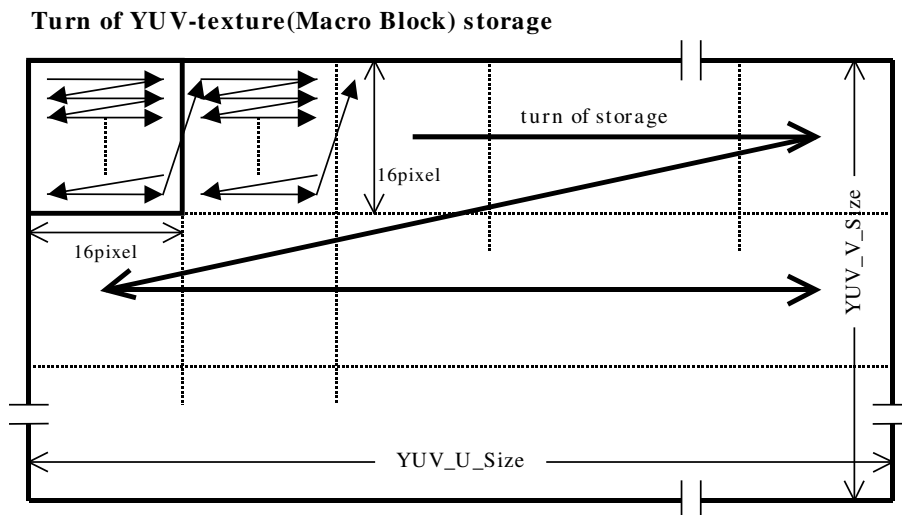


Fig. 3-83

## **§ 4 Peripheral Interface**

This system is equipped with two bus interfaces, the G1 Bus and the G2 Bus, for interfacing with peripheral devices such as the audio chip AICA, the GD-ROM (the name for the CD-ROM drive in this system), and a modem. The devices are grouped into asynchronous and synchronous devices, and each device is assigned to one of these interfaces. The interfaces are described below.

## § 4.1 G1 Bus

The G1 Bus is used to read country codes and access asynchronous devices that are connected to the G1 Bus, such as the GD-ROM, system ROM, and FLASH memory. This interface is designed to handle large volumes of data from the GD-ROM, for system bootup, and to access environment settings, etc. Each of the devices connected to this bus is accessed by a different method, as described below.

The general setting registers for the G1 Bus are located in the SB block; the only type of access that is possible from the SH4 is 4-byte access in the non-cache area. There are separate device setting registers for the GD-ROM.

The only interrupts from external devices connected to the G1 Bus are those from the GD-ROM device; these are level interrupts that are input from the device. The interrupt source can be determined by reading the GD-ROM device registers; reading the drive's status register clears the interrupt. (For details on interrupt sources, refer to section 8.5.2.)

### § 4.1.1 GD-ROM

The CD-ROM that is installed in the Dreamcast System is called the "GD-ROM," and is used to supply large amounts of music data, game data, and program source code to the host system from Sega's proprietary "GD-ROM" discs. The music data (CD-DA) from the GD-ROM is output as digital audio data to the AICA audio chip on a 48Fs (Fs = 44.1KHz) cycle. The GD-ROM also supplies a 33.8688MHz clock signal (the audio clock source) to the AICA. The GD-ROM is located on the system's G1 Bus, and in addition to the digital audio music data, the GD-ROM supplies program source code and data to the system through the G1 Bus. The Dreamcast System's main program can only be started up from CD media. The GD-ROM also supports a wide variety of proprietary media.

The main specifications of the GD-ROM drive are listed below:

- Access time (1/3 stroke): 250ms or less
- Normal area: 4x; high-density area: 6 to 12x (CAV: 2000rpm)
- Buffer memory: 128K
- CD-DA shock-proof function built in
- Ball chucking
- Multiple security functions
- Can read the following types of discs:
  - GD-ROM
  - CD-DA, CD-ROM
  - Photo CD, video CD
  - CD Extra, CD + G, CD + EG
- The following disc types are rejected:
  - CD-I, CD-I Ready (playback possible)

Regarding the disc specifications, the basic physical formats have separate audio and data tracks, and a single disc includes both a "single-density (program) area," which consists of normal density tracks, and a "high-density (program) area," which consists of high density tracks

The physical format of the "single-density (program) area" conforms with the "RED BOOK" and "YELLOW BOOK" CD-ROM standards, and the physical format conforms with ISO9660. This format can be played back by a normal CD-ROM drive.

The physical format of the "high-density (program) area" conforms with a proprietary Sega standard, and the physical format conforms with ISO9660. This format can only be played back by a CD-ROM drive that conforms with the Sega standard.

#### § 4.1.1.1 Register Map

Regarding register access for the GD-ROM device, read/write accesses to data registers are made in 16-bit (2-byte) sizes, while read/write accesses to other registers are made in 8-bit (1-



byte) sizes. The GD-ROM device register map is shown below.

Address	Function ( Read / Write )
0x005F 7000	Reserved
0x005F 7018	Alternate Status / Device Control
:	Reserved
0x005F 7080	Data / Data
0x005F 7084	Error / Features
0x005F 7088	Interrupt Reason / Sector Count
0x005F 708C	Sector Number / Sector Number
0x005F 7090	Byte Control Low / Byte Control Low
0x005F 7094	Byte Control High / Byte Control High
0x005F 7098	Drive Select / Drive Select
0x005F 709C	Status / Command

Table 4-1

#### § 4.1.1.2 Access Methods

Although the GD-ROM access timing is based on the ATA standard (the electrical interface conforms with ATA-3), the GD-ROM supports only the timing modes listed below. (The GD-ROM does not support "Single Word-DMA" from the ATA standard.)

- (1) PIO Modes 0 to 4  
Accesses to the GD-ROM by the CPU are only possible as byte or word accesses at 4-byte boundaries in the non-cacheable area. In this case, external accesses are single byte or word accesses.
- (2) Multi Word DMA Modes 0 to 2  
Read accesses from the GD-ROM by means of DMA permit transfer of any number of bytes at 1-byte boundaries. However, because internal operation is based on 32-byte burst access, if a number of bytes that is not evenly divisible by 32 are transferred, the excess transfer capacity is filled with zeroes. In this case, external accesses are performed as (number of transfer bytes/2) word accesses. If the GD-ROM is accessed in the middle of a DMA transfer, the bus is released for the access (\*interrupting the transfer) as long as the G1 Bus signals G1DREQ and G1DACKN are not active.

For details on access methods, refer to the GD-ROM protocol SPI specifications.

#### § 4.1.1.3 Initial Settings

#### § 4.1.1.4 Access Procedure

### § 4.1.2 System ROM

The system ROM stores the system data and boot routines for the Dreamcast System, and is accessed by the CPU. The following data is stored in the system ROM:

- (1) IPL codes: Boot processing and system configuration
- (2) Multiplayer interface
- (3) Dreamcast-OS core

The system ROM is mapped in SH4 area 0, and the duration of the address setup and hold times and the read and write pulses in the bus cycle can be specified through register settings. Access is possible as 1-, 2-, 4-, or 32-byte access.

The following table shows the contents of the ROM specifications that are used in the Dreamcast system.

Type	Mask ROM
ROM size (capacity)	16Mbit
Bus width	8/16bit
Access time	100ns~240ns

Table 4-2 ROM Specifications

#### § 4.1.2.1 Access Methods

Access to ROM and the FLASH memory (described later) is always performed from the CPU, and is possible as 1-, 2-, 4-, or 32-byte access. Note that 32-byte access can only be made to the cache area.

#### § 4.1.2.2 System Initial Settings

#### § 4.1.2.3 Access Procedure

### **§ 4.1.3 FLASH Memory**

FLASH memory is used for backing up data for hermits, IDs, and communications. The size of FLASH memory is 128K (8-bit bus) and has an 8K protected area. Data can be written/erased a minimum of 100,000 times.

#### **§ 4.1.3.1 System Initial Settings**

#### **§ 4.1.3.2 Access Procedure**

#### § 4.1.4 System Code

The system code is set by pull-up/pull-down resistors connected to the A/D lines (HOLLY pins: G1MRA[18:11]) on the G1 Bus on the board. The CPU can read the system code that is set on the board by reading SB\_G1SYSM (0x005F74B0) in the G1 Interface Block Control Registers. For details on the register, refer to section 8.4.1.3, "G1 Interface."

The system code includes four bits, G1MRA [18:15], which identifies the unit as a product or a development unit. The system code settings are described below. (Note that in the table "0" indicates that the line is pulled down, and "1" indicates that the line is pulled up. Any settings that are not shown in the table below are reserved.)

G1MRA				System
18	17	16	15	
0	0	0	0	Mass production unit
1	1	0	0	SET4-8M
1	0	0	0	SET4-32M
1	0	0	1	Dev.Box-16M
1	0	1	0	Dev.Box-32M
1	1	0	1	Graphics box
Other codes				Reserved

Table 4-3

G1MRA[14:11] is the country code. The settings are described below.

(Note that, in the table, "0" signifies "pull down" and "1" signifies "pull up;" any settings not shown in the table are "Reserved.")

G1MRA				Destination region
14	13	12	11	
0	0	0	1	Japan, South Korea, Asia NTSC
0	1	0	0	North America, Brazil, Argentina
1	1	0	0	Europe

Table 4-4

Note that the language that is shown on the initial setting screen is Japanese for both Asia NTSC and South Korea, and English for the Brazil PAL/M region and the Argentina PAL/N region.

##### § 4.1.4.1 Initial Setting

None.

##### § 4.1.4.2 Access Procedure

This register can only be accessed by 32-bit access, because it is a HOLLY internal register.

## § 4.2 G2 Interface

The G2 interface is used to access the AICA audio chip, a modem, and any expansion devices that are connected to the G2 Bus, which is synchronized with a 25MHz clock.

### § 4.2.1 Interface

The G2 bus is an expansion bus (a bus for the connection of expansion devices) that is used for connection with the audio IC AICA or a modem.

The control block for this bus does not have a target function (a function that would permit access by a device connected to the G2 bus), only a master function (a function that permits access to a device that is connected to the G2 bus). In other words, devices that are connected to the G2 bus only function as targets.

Therefore, a device that is connected to the G2 bus cannot be accessed directly from buses other than the G2 bus, such as the CPU bus. In addition, a device that is connected to the G2 bus cannot transfer data to another device that is connected to the G2 bus. Data transfers between devices that are connected to the G2 bus are accomplished through repeated read/write operations by the CPU, etc.

Data transfers involving devices that are connected to the G2 bus are conducted with a 16-bit data bus. The bus operation clock is 25MHz, so accesses to expansion devices are made using 1/16 or less of the CPU bandwidth. (The CPU's internal clock is 200MHz, and the register width is 32 bits.)

The G2 bus control block includes a CPU write FIFO buffer that operates either as 8 levels x 4 bytes or 1 level x 32 bytes. Writes to the G2 bus registers and to devices that are connected to the G2 bus are performed through the write FIFO buffer. Reads can only be initiated once the write FIFO is empty. Therefore, if an attempt is made to perform a read from the G2 bus after having performed a write to a slow device that is connected to the G2 bus, all functions that link the CPU to the G2 bus (CPU bus <-> SB <-> G2 bus) will lock up until the read is completed. In order to prevent the buses from locking up, it is necessary to check the state of the write FIFO buffer when accessing a slow device.

The G2 bus control block includes a DMA transfer function ("G2-DMA," hereafter) that is used to transfer data via the G2 bus without depending on the CPU. G2-DMA is supported for four channels that operate independently, and are used for data transfers between system memory and devices that are connected to the G2 bus. Data is transferred in units of 32 bytes. The G2 bus also includes a control input line that is used to initiate G2-DMA transfers. G2-DMA operates without regard to the status of the write FIFO buffer. when G2-DMA and the CPU are both accessing the G2 bus, their priority ranking alternates. In addition, the priority among the G2-DMA channels is switched on a round-robin basis.

The G2 bus includes three interrupt inputs: the AICA and the modem each control one, and the third is used by external expansion devices connected to the G2 bus. When multiple external expansion devices are connected to the G2 bus, the one interrupt input is shared by all of the devices.

If an interrupt is generated because the CPU accessed a device that is connected to the G2 bus, but the area being accessed has no corresponding device that is connected, or if a CPU timeout interrupt is generated during a G2-DMA transfer, the interrupt is generated even if G2-DMA has completed its data transfer.

Furthermore, an interrupt is generated and G2-DMA is not initiated: if an invalid value is set in the G2-DMA register; if an invalid value is set in the SB\_ADSTAG register or the SB\_ADSTAR register (when cho:AICA in either case) and G2-DMA is enabled; or an address that is outside of the range set by the SB\_G2APRO register is set in the SB\_ADSTAR register (when cho:AICA) and G2-DMA is enabled.

For details on setting up and using interrupts, refer to sections 2.7, 8.4.1.1, and 8.5.

For details on AICA and modem devices for connection to the G2 bus, and for explanations of restrictions concerning the creation of new devices for connection to the G2 bus, refer to the corresponding sections.

The term "CPU" refers to a controller that is not the G2 control block and is not a device that is connected to the G2 bus.

#### <<Addresses Used for the G2 Bus>>

The addresses listed below (Table 4-5) are allocated to the G2 bus. Table 4-6 indicates valid addresses. (Note that the addresses that are indicated are physical addresses, and are different from the logical addresses that the CPU uses.)

Address range	Size	Contents
0x005F7800 - 0x005F7BFF	1KB	G2 bus control register area
0x00600000 - 0x0067FFFF	512KB	G2 bus access area (primarily for modem)
0x00700000 - 0x01FFFFFF	25MB	G2 bus access area (primarily for AICA)
0x02700000 - 0x03FFFFFF	25MB	G2 bus access area (AICA image)
0x14000000 - 0x17FFFFFF	64MB	G2 bus access area (unused; for expansion)

Table 4-5

Address range	Size	Access	Contents
0x005F7800 0x005F78FF	- 256B	4	G2 bus control registers
0x00600000 0x006007FF	- 2KB	1/2/4	Asynchronous cycle area (for modem)... Note 2
0x00620000 0x0062FFFF	- 64KB	1/2/4 /32	Synchronous cycle 16-bit address area (unused; for expansion)
0x00700000 0x00FFFFFF	- 9MB	1/2/4 /32	Synchronous cycle 32-bit address area (for AICA)...Note 3
0x01000000 0x01FFFFFF	- 16MB	1/2/4 /32	Synchronous cycle 32-bit address area (unused; for expansion)
0x02700000 0x02FFFFFF	- 9MB	1/2/4 /32	Synchronous cycle 32-bit address area (AICA image)
0x03000000 0x03FFFFFF	- 16MB	1/2/4 /32	Synchronous cycle 32-bit address area (unused; for expansion)
0x14000000 0x17FFFFFF	- 64MB	1/2/4 /32	Synchronous cycle 32-bit address area (unused; for expansion)

Note 1: "Access" indicates the byte size that can be accessed, as follows:

- 32: 32-byte continuous access permitted
- 4: 4-byte (long word) access permitted
- 2: 2-byte (word) access permitted
- 1: 1-byte access permitted
- : Access not permitted

Note 2: In the asynchronous cycle area, only 1-/2-/4-byte access is permitted, at +0 addresses. Byte access at +1, +2, and +3 addresses, 2-byte (word) access at +2 addresses and 32-byte continuous access is prohibited.

Note 3: Access to AICA areas is as shown in the above table, but refer to the section corresponding to the actual AICA chip.

Table 4-6

Access to the following addresses is prohibited:

Address range	Size	Contents
0x005F7900 - 0x005F7BFF	768B	Test area
0x00600800 - 0x0061FFFF	126KB	Test area
0x00630000 - 0x0067FFFF	340KB	Test area

Table 4-7

Accesses to addresses other than those listed in the above tables are not G2 bus accesses.

<<Restrictions Concerning the G2 Bus>>

- Access to the G2 bus control registers must be made as 4-byte long word access.
- In the asynchronous cycle area, access to addresses for which address bits A1 and A0 are 0b00 (0x00600000, 0x00600004, etc.) is performed as 1-/2-/4-byte accesses, and the lower 8-bits of data are valid. 1-/2-byte accesses to addresses for which address bits A1 and A0 are not 0b00 are prohibited.
- Synchronous cycle 16-bit address areas can be accessed either through 1-byte, 2-byte (word), 4-byte (long-word) or 32-byte continuous access. However, these areas are unused in the standard configuration.
- Synchronous cycle 32-bit address areas can be accessed either through 1-byte, 2-byte (word), 4-byte (long-word) or 32-byte continuous access. However, these areas are unused in the standard configuration, except for AICA. note that there are restrictions on usage for AICA; refer to the corresponding sections.
- The availability of DMA on the G2 bus is indicated below.

The following DMA transfers are usable:

- 1) AICA-DMA: System memory → AICA (Mode that appears empty at CPU initiation)
- 2) EXT-DMA0: System memory → External expansion device (Mode that appears empty at CPU initiation)  
External expansion device → System memory (Mode that appears empty at CPU initiation)
- 3) EXT-DMA1: System memory → External expansion device (Mode that appears empty at CPU initiation)  
External expansion device → System memory (Mode that appears empty at CPU initiation)
- 4) GD-DMA: GD-ROM → external expansion device

Use of the following DMA transfers is prohibited:

- 1) AICA-DMA: System memory → AICA (Any mode other than the mode that appears empty at CPU initiation)  
AICA → system memory
- 2) EXT-DMA0: (Any mode other than the mode that appears empty at CPU initiation)
- 3) EXT-DMA1: (Any mode other than the mode that appears empty at CPU initiation)
- 4) GD-DMA: External expansion device → GD-ROM  
AICA → GD-ROM

- Software access restrictions are listed below:
  - If the SH4 accesses the G2 bus at the same time that GD-DMA <<GD -> AICA>> or <<GD → EXT>> is being executed, a 16[micro]s or longer cycle may be generated on the Root bus (the internal bus) causing Maple to overflow. Therefore, access to the G2 bus should wait until GD-DMA ends and after the buffer has been confirmed as being empty. Details are provided below:
    - GD-DMA <<GD-ROM → AICA>> and an SH4 G2 bus access cannot both occur at the same time. If the SH4 accesses the G2 bus while there is burst write data for AICA in the G2 interface buffer and the AICA buffer, the SH4 needs to wait only for the duration of the AICA 32-byte transfer. (This is because, if there is burst data in the G2 buffer, the next data is not accepted from the SH4 interface until the buffer becomes empty.)
    - GD-DMA <<GD-ROM -> EXT>> and an SH4 AICA read access cannot both occur at the same time. If the SH4 performs an AICA read while there is write data for an external expansion device in the G2 interface buffer, the SH4 needs to wait only for the duration of the write to the external expansion device and the AICA read.



#### § 4.2.2 AICA

The AICA chip controls the sound system.

The main specifications for the AICA sound chip are listed below.

- Sampling frequency: 44.1KHz
- Dedicated sound processor (ARM7DI) on chip; provides seven interrupts with priority levels through register flags
- Parallel processing DSP that is specialized for voice processing
  - 128 steps; includes ring buffer function
- PCM sound source on chip
  - PCM data format: 8-, 16-bit linear/4-bit ADPCM (ADPCM is a proprietary format established by YAMAHA)
- A maximum of 64 voices
  - Independent LFO (Low Frequency Oscillators) function and EG (Envelope Generator) function for all channels
  - LPF with a cutoff frequency that can be varied over time for all channels
  - Pitch change possible on all channels
  - Forward loop function
- Supports one external digital audio input
- Provides an SDRAM interface as external memory, permits common access by the AICA's internal sound processor, the sound source, the DSP, and the system (SH4)
- Digital mixer on chip permits digital mixing of signals from the DSP, a PCM sound source and an external digital audio input
- Supports a Real Time Clock (RTC) by means of a secondary battery

A summary of the chip specifications and configuration is provided below.

The AICA is an audio chip with its own internal 64-channel PCM/ADPCM sound source, supports sound effects produced by its 128-step/sample (1 sample = 44.1KHz) DSP and dedicated sound processor, generates sound data, and processes waveform data from the host system. The sound data that is generated by the AICA can be mixed as digital audio output with one external digital audio input. At the output stage, the signal can be output as 64Fs digital audio to an audio DAC/AMP that is external to the chip. In the Dreamcast system, 48Fs digital audio from the GD-ROM is input to the AICA as external audio. The digital audio output (64Fs) that is generated by the AICA passes through the audio DAC and AMP, and is output as stereo sound through the RCA and expansion VGA connectors, along with the video signals from the graphics system.

The AICA chip configuration consists of 2MB of wave memory (SDRAM) for wave data from the internal sound source and the host system, etc.; an RTC (Real Time Clock), and a MIDI interface for development purposes. A 3V lithium battery and a 32.768KHz crystal are added externally for backup of the RTC. The AICA also provides access to the video mode settings (switches) for the DVE (video DAC/encoder).

Regarding the clock system, the interface block and the audio block use different clock frequencies. The audio block uses a 22.5792MHz clock that is generated by passing the 33.8688MHz clock signal that is supplied from the GD-ROM through a PLL in the audio block. The host system interface block uses a 25MHz clock that is supplied from the G2 Bus.

The chip configuration and connections with peripheral devices are illustrated below.

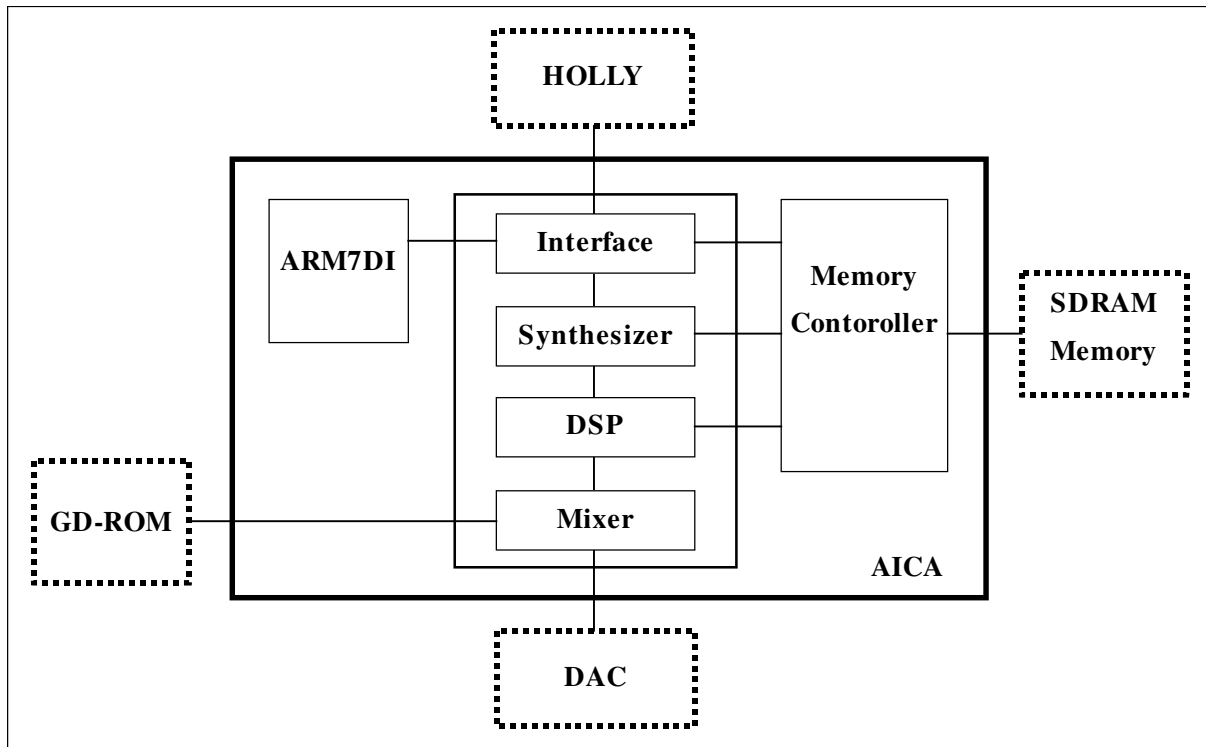


Fig. 4-1 Internal Block Diagram of the AICA

#### § 4.2.2.1 Memory/Register Map

The following table describes the memory/register area for the AICA for physical memory accesses by the SH4 on the system side, and the area that is accessed by the ARM, which is the AICA's internal sound CPU.

A register map and an explanation regarding the channel, common and DSP data is provided in section 8.4.5.

Note that the allowed access size for accesses of the AICA area by the SH4 is 4 bytes only.

Area	G2 (SH4) addresses	AICA internal (ARM) addresses
Channel data	0x0070 0000~0x0070 27FF	0x0080 0000~0x0080 27FF
Common data	0x0070 2800~0x0070 2FFF	0x0080 2800~0x0080 2FFF
DSP data	0x0070 3000~0x0070 7FFF	0x0080 3000~0x0080 7FFF
RTCdata	0x0071 0000~0x0071 000B	-
Memory area (SDRAM)	0x0080 0000~0x00FF FFFF	0x0000 0000~0x007F FFFF

Table 4-8 AICA Memory/Register Map

- The usable size of the memory area (SDRAM) that is shown in the table varies because the amount of memory that is installed depends on whether the system is to be used for development purposes or not.

(G2) 2MB:	0x0080	0000-0x009F	FFFF;
Development version 8MB:		0x0080 0000-0x00FF FFFF	
(AICA) 2MB:	0x0000	0000-0x001F	FFFF;
Development version 8MB:		0x0000 0000-0x007F FFFF	

- The G2 address information listed in the table indicates physical memory addresses. In the

case of an access by the SH4, the actual access address depends on the area where the SH4 is conducting its cache access. (Refer to the cache access table in section 2.1.)

#### § 4.2.2.2 Initial Settings

#### § 4.2.2.3 Access Procedure

A restriction in the AICA specifications requires slots to be left open so that AICA accesses by the SH4 are completed within 16[μ]sec.

The following restrictions apply to AICA accesses by the SH4.

- Reads of the AICA by the SH4 sometimes take 16[μ]sec., during which time the CPU bus, the internal bus (Root Bus) and the G2 bus come to a complete stop. Therefore, reducing AICA reads is a necessity for making the system faster.
- Similarly, writes to the AICA are fast if the AICA buffer is empty, but sometimes more than 16[μ]sec. are required in order to send data in the buffer to wave memory so that the buffer is empty. In order to increase system speed, it is necessary to limit consecutive writes to the AICA to eight times; DMA can be used to reduce the number of writes.

Restrictions concerning DMA are provided below.

- If the AICA is to be read by the SH4 while AICA-DMA, EXT-DMA0, or EXT-DMA1 is being executed, interrupt the AICA-DMA, EXT-DMA0, or EXT-DMA1 operation and confirm that the buffer is empty before reading the AICA. After the AICA read is completed, resume the DMA that was interrupted. Alternatively, wait until DMA is completed and then confirm that the buffer is empty before reading the AICA.
- \* From the time when the buffer has been confirmed as being empty until the AICA read is completed, it is possible to access system memory, the TA FIFO buffer and the interrupt control register, but other accesses from the SH4 are prohibited.

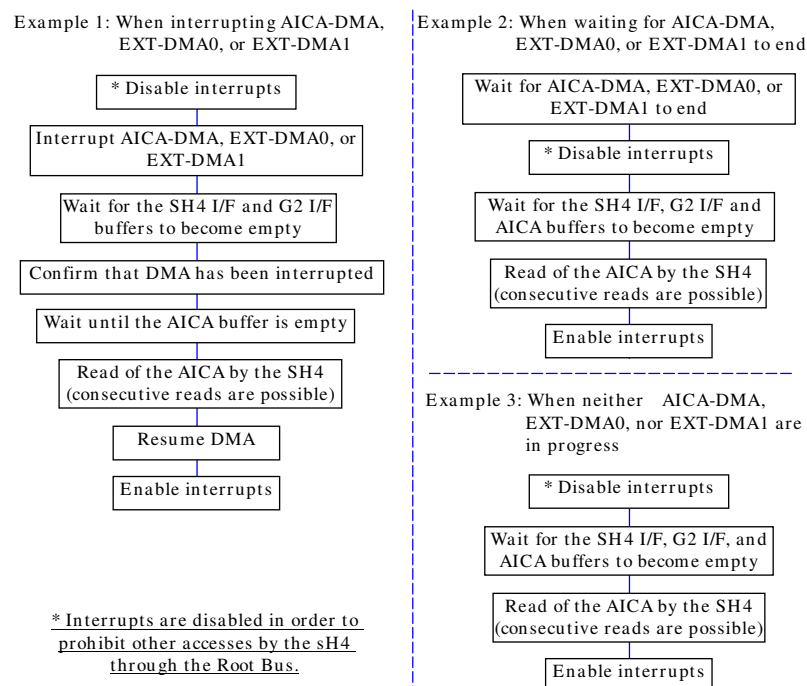


Fig. 4-2

- To perform a write to the AICA by the SH4 while AICA-DMA is in progress, interrupt AICA-DMA and confirm that the buffer is empty before writing to the AICA. After the write to the AICA is completed, resume the AICA-DMA. Otherwise, wait for the AICA-DMA to end, confirm that the buffer is empty and then write to the AICA. In addition, writes to the AICA should be performed no more than eight consecutive times. If more than eight writes to the AICA are to be performed, confirm that the buffer is empty again after eight or fewer writes, and then perform eight or fewer writes to the AICA again.
- \* SH4 accesses other than those to the AICA are possible.

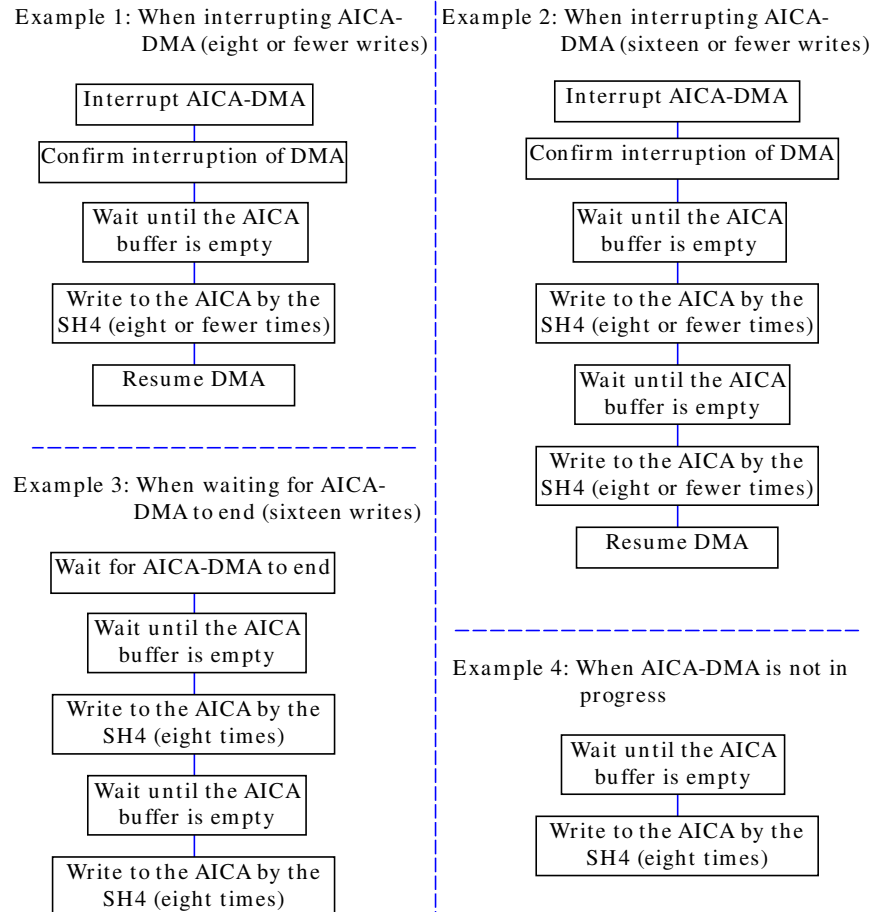


Fig. 4-3

#### § 4.2.2.4 Wave Memory

The AICA has an interface for externally connected SDRAM that is shared and accessed by its internal sound processor, sound source, DSP, and the host system. Table 4-2 shows the specifications for the memory that is used. The following table shows the specifications and settings for the memory that is used.

Memory size	2MB (can be expanded up to 8MB)
Technology	16Mbit SDRAM (2banks × 512Kwords × 16bits)
Number of memory maps used	1
Chip bus width	16 bits
Operating frequency	67.7376MHz
Operation settings	<ul style="list-style-type: none"><li>• Burst Read and Single Write</li><li>• Wrap Type = Sequential</li><li>• CAS Latency = 2</li><li>• Burst Length = Full Page</li></ul>

Table 4-9 AICA's External Memory Specifications

### § 4.2.3 RTC(Real Time Clock)

The Real Time Clock (RTC), a timer that increments its count once per second, is built into the AICA audio chip, and is capable of operating off a backup battery at 2.0 to 3.5V even when the main power is off. A 3V lithium battery and a 32.768KHz crystal (as the clock source) are each connected externally to the AICA for the RTC.

Only the SH4 can access the RTC.

#### § 4.2.3.1 Access Method

The RTC is accessed through the following three registers, starting from 0x00710000, described below.

As shown below, the RTC registers form a 32-bit counter RTC [31:0], which can count seconds for approximately 136 years, and a write enable bit (EN...0x00710008-bit 0, write only) for those registers. (Because these registers are accessed in the same manner as AICA, the access size is 4 bytes only, and only the lower 16 bits are valid.)

RTC[31:0] is normally write protected, but can be written when a "1" is written to the EN bit. Furthermore, when RTC[15:0] is written, the counter below one second is cleared. When RTC[31:16] is written, write protection is enabled again.

If the data is read while the count is being increased, the correct value might not be output. Therefore, it is necessary to confirm the value by reading several times, for example.

#### RTC Resister

Address : 0x0071 0000

bit15-0
RTC[31:16]

Address : 0x0071 0004

bit15-0
RTC[15:0]

Address : 0x0071 0008

bit15-1	0
Reserved	EN

#### § 4.2.4 MODEM

In order to add communications functions to the Dreamcast System, an external plug-in type modem is supported. The modem consists of a modem chip, telephone line interface, and an ASIC that includes ID circuitry.

The modem uses Rockwell's RCVDL56DPGL/SP chip set, and the operating frequency of the modem block is 56.448MHz. After a reset is released, the modem requires an interval of at least 400ms in order to initialize its registers.

The major functions and features of the modem are listed below:

- Full duplex V.34 (33.6Kbps) data modem
- Supports MNP2-5, V.42, and V.42bis for error correction and data compression (Error correction and data compression are performed by system software.)
- Passive modem; does not include a controlling microprocessor

##### § 4.2.4.1 Address Map

The address map is shown below. (Refer to "RCVDL56DPFL/SP, RCV56DPFL/SP, RCV336DPFL/SP Modem Data Pump Designer's Guide" for details.)

ADDRESS	CONTENTS	ADDRESS	CONTENTS
0x0060 0000	Modem ID0	0x0060 0440	Modem Register No.10
0x0060 0004	Modem ID1	0x0060 0444	11
:		0x0060 0448	12
0x0060 0400	Modem Register No.00	0x0060 044C	13
0x0060 0404	01	0x0060 0450	14
0x0060 0408	02	0x0060 0454	15
0x0060 040C	03	0x0060 0458	16
0x0060 0410	04	0x0060 045C	17
0x0060 0414	05	0x0060 0460	18
0x0060 0418	06	0x0060 0464	19
0x0060 041C	07	0x0060 0468	1A
0x0060 0420	08	0x0060 046C	1B
0x0060 0424	09	0x0060 0470	1C
0x0060 0428	0A	0x0060 0474	1D
0x0060 042C	0B	0x0060 0478	1E
0x0060 0430	0C	0x0060 047C	1F
0x0060 0434	0D	0x0060 048C	HRES
0x0060 0438	0E		
0x0060 043C	0F		

Table 4-10

##### § 4.2.4.2 Access Method

The modem area is mapped in the area from 0x0060 0000 to 0x0060 07FF on the G2 bus (the effective addresses that are actually mapped are listed in the table above). Each register can be accessed only by means of one-byte reads and writes.

#### § 4.2.4.2.1 ID

The ID register is used to get the ID of the device that is connected in the modem slot. As described earlier, the access size is 1 byte only, and this register is a read-only register. The register contents are described below.

##### MODEM ID0

Address : 0x0060 0000

bit 7-0
Country Code

Country Code (default = 0x00)

Value	Country
0x00	Reserved
0x01	Japan
0x02	USA
0x02-0xFF	Reserved

##### MODEM ID1

Address : 0x0060 0004

bit 7-4	3-0
Maker Code	Device Type

Maker Code (default = 0x1)

Value	Maker
0x0	SEGA
0x1	Rockwell
0x2-0xF	Reserved

Device Type (default = 0x0)

Value	Device
0x0	33.6Kbps
0x1-0xF	Reserved

#### § 4.2.4.2.2 Reset

A hardware reset of the modem chip is performed through the HRES register (0x00600480). The modem set requires a minimum reset interval of 3[micro]sec, and a minimum of 400msec after releasing the reset.

##### HRES

Address : 0x0060 0480

bit 7-0
Reset

Reset (default=0x0)

Setting	Status
0x0	Reset
0x1	Reset release



## § 4.2.5 Expansion Devices

Expansion devices are connected through the main unit's expansion connector (the G2 bus), and are accessed through the G2 interface.

Expansion devices respond on the G2 bus synchronous cycle. Because the modem uses an asynchronous cycle, the expansion device must not respond.

<<Area Assignments>>

Expansion devices are normally assigned to 1K areas (refer to the table below) in what is normally a 16K space that starts from the 16-bit address 0x0000. The address assignment is based on the state of the expansion device's SELx pins.

area	address range	SH4 address
0	0x0000 - 0x03FF	0x00620000 - 0x006203FF
1	0x0400 - 0x07FF	0x00620400 - 0x006207FF
2	0x0800 - 0x0BFF	0x00620800 - 0x00620BFF
3	0x0C00 - 0x0FFF	0x00620C00 - 0x00620FFF
4	0x1000 - 0x13FF	0x00621000 - 0x006213FF
5	0x1400 - 0x17FF	0x00621400 - 0x006217FF
6	0x1800 - 0x1BFF	0x00621800 - 0x00621BFF
7	0x1C00 - 0x1FFF	0x00621C00 - 0x00621FFF
8	0x2000 - 0x23FF	0x00622000 - 0x006223FF
9	0x2400 - 0x27FF	0x00622400 - 0x006227FF
10	0x2800 - 0x2BFF	0x00622800 - 0x00622BFF
11	0x2C00 - 0x2FFF	0x00622C00 - 0x00622FFF
12	0x3000 - 0x33FF	0x00623000 - 0x006233FF
13	0x3400 - 0x37FF	0x00623400 - 0x006237FF
14	0x3800 - 0x3BFF	0x00623800 - 0x00623BFF
15	0x3C00 - 0x3FFF	0x00623C00 - 0x00623FFF

Table 4-11

Expansion devices can use the address spaces shown below.

ADDRESS	SIZE	AREA
0x00620000~0x0062FFFF	64KByte	Synchronous cycle 16-bit address area
0x01000000~0x01FFFFFF	16MByte	Synchronous cycle 32-bit address area
0x03000000~0x03FFFFFF	16MByte	Synchronous cycle 32-bit address area
0x14000000~0x17FFFFFF	64MByte	Synchronous cycle 32-bit address area

\*It is recommended that 0x03000000 through 0x03FFFFFF be the image for 0x01000000 to 0x01FFFFFF.

Table 4-12

The 1K spaces have a 32-byte basic register set that is common to the expansion devices, and which functions as configuration registers that are set when using expansion device interrupts or an area that exceeds 1K.

When requesting an area that is greater than 1K in size, the configuration register is used by the software for resource management in order to assign the area. In addition, because it is not possible to guarantee that an expansion device will occupy a fixed area (occupying specific addresses is prohibited), the software must be configured so that no problems arise no matter which area is allocated to an expansion device.

All expansion devices share one interrupt. In addition, the three G2-DMA transfer requests are used by all expansion devices on an exclusive basis.

### <<Configuration Registers>>

The configuration registers occupy a 32-byte area at the beginning of the 1K area that is the basic space for expansion devices. The contents of these registers are described below. A reset resets each register to "0."

Offset Address	Access	Contents
0x0000	R/-	ID0: G2 identifier low
0x0002	R/-	ID1: G2 identifier high
0x0004	R/-	ID2: Individual identifier
0x0006	R/-	ID3: Individual identifier
0x0008	R/-	ID4: Individual identifier
0x000A	R/-	ID5 : Device Category
0x000C	R/-	ID6 : Serial Number
0x000E	R/-	ID7 : Comaptible Number
0x0010	R/W	Reg0 : Address Space 0
0x0012	R/W	Reg1 : Address Space 1
0x0014	R/W	Reg2 : DMA Transfer Request Assign
0x0016	R/W	Reg3 : Device Enable Register
0x0018	R/W	Reg4 : Interrupt Mask Low
0x001A	R/W	Reg5 : Interrupt Mask High
0x001C	R/W	Reg6 : Interrupt Status Low
0x001E	R/W	Reg7 : Interrupt Status High

\*Access (read only) to 0x0004 through 0x000E differs for each device.

Tabke 4-13

#### ○ 0x0000 to 0x0002: ID0-1 G2 identifier

This identifier is used to determine whether the rest of the registers that follow are the correct registers.

The sequence of bytes, starting from 0x0000, is "G", "A", "P", and "S". (temporary)

```

bit31-24    (0x0002_bit15-8) : 0x53
bit23-16    (0x0002_bit7-0)  : 0x4D
bit15-8     (0x0000_bit15-8) : 0x41
bit7-0      (0x0000_bit7-0)  : 0x47

```

#### ○ 0x0004 to 0x0008: ID2 to 4 Individual identifier

This area can be used in any fashion desired.

#### ○ 0x000A : ID5 Device Category

This indicates the general category of the device. Each bit indicates a function category. If the device has multiple functions, each of the corresponding bits is set to "1".

```

bit15      : G2 bus bridge/repeater
bit14      : -
bit13      : Extra Bus Bridge (PCMCIA, ISA, etc.)
bit12-7    : -
bit6       : Miscellaneous I/O (Keyboard, MOUSE, etc.)
bit5       : LAN/Ethernet
bit4       : SCSI
bit3       : ATA/IDE/compact-FLASH
bit2       : parallel (IEEE1284-1994)
bit1       : serial/Modem/ISDN (165x0)
bito      : Memory (DRAM/DRAM)

```

○ 0x000C: ID6      Serial number

This is a product code that identifies the device.

This register consists of the manufacturer's code (8 bits) and a serial number (8 bits); the serial number is managed by the manufacturer. The manufacturer's code 0x00 and the serial number 0x00 can be used as a prototype code. The serial number correspondence is shown below.

“ 0xXXYY “

bit15-8	: maker code	;XX
bit7-0	: serial number	;YY
XX--	= 0x00--	: Standard device (internal specifications are public)
XXYY	= 0x0000	: Prototype
XXYY	= 0x0001	: DRAM interface
XXYY	= 0x0002	: Simple ISA interface
XXYY	= 0x0003	: G2 bus buffer
XXYY	= 0x0004	: Simple IDE interface
XX--	= 0x01--	: Sega
XXYY	= 0x0100	: Sega prototype

○ 0x000E: ID7      Compatible Number

This register is specified when the device indicated by the serial number is the same, or is upward compatible, and the control software can be used as is. If there is no compatible device, this register is 0x0000.

bit15-8	: maker code
bit7-0	: serial number

○ 0x0010: Reg0      Address Space 0

Set this register when an area larger than the basic 1K space is required. Specify addresses in units of 64K x 2[n] bytes. When 0x0000 is specified, device allocation is prohibited.

If the address space is not required, this register can be treated as a read-only register that returns 0x0000.

bits 15-13	: N.A.
bits 12-0	: Correspond to A28 through 16 of a CPU address

○ 0x0012: Reg1      Address Space 1

Address 0x0012 has the same function as address 0x0010; 0x0010 and 0x0012 can be used to allocate two different address spaces. Specify addresses and areas in units of 64K x 2[n] bytes.

If the address space is not required, this register can be treated as a read-only register that returns 0x0000.

This register is used when an area larger than the standard 1K space is required..

bits15-13	: N.A.
bits12-0	: Correspond to A28 through 16 of a CPU address

○ 0x0014: Reg2 DMA Transfer Request Assign

This register is used to select signal lines for outputting a transfer request to G2-DMA. When a bit is set to "1," the corresponding signal line is driven. If G2-DMA is not being used, this register can be treated as a read-only register that returns 0x0000.

bits15-4	: N.A.					
bits3	:	DMA	Request	Signal	Select	(G2RQDEVN)
		0-		Signal		High-Z
		1- Active				
bits2	:	DMA	Request	Signal	Select	(G2RQEX2N)
		0-		Signal		High-Z
		1- Active				
bit1	:	DMA	Request	Signal	Select	(G2RQEX1N)
		0-		Signal		High-Z
		1- Active				
bit0	: N.A.					

○ 0x0016: Reg3 Device Enable Reg

This register is used to enable a device and the area allocated by 0x0010 and 0x0012. Bit 1 permits "1" to be read from read/write registers. The 1K area that is standard can always be enabled.

bits15-2	: N.A.					
bit1	:	Device		Register		Mask
		0-		Mask		Off
		1- Mask On				
bit0	:	Device				Enable
		0-				Dis
		1- Enable				

○ 0x0018 - 0x001A: Reg4 - Reg5 Interrupt Mask

These mask registers control the output of interrupts when interrupts are generated. These registers can control up to 32 sources; when a bit is set to "1," the corresponding interrupt output is enabled. If interrupt are not being used, this register can be treated as a read-only register that returns 0x00000000.

These registers are packed, starting from the lowest bit.

bits 31-0 (0x001A\_bit15-0, 0x0018\_bit15-0)

○ 0x001C - 0x001E: Reg6 - Reg7 Interrupt Status

These registers reflect the status of interrupts that have been generated by the device. When a bit is "1," the corresponding interrupt is being generated. These registers are used in conjunction with the interrupt mask registers; when the bits that correspond to an interrupt output are both "1," that interrupt output is low. When there are multiple interrupt sources, one or more interrupts are generated, and the interrupt outputs go low if the corresponding mask bits are set to "1."

Depending on the device, it may also be possible to clear an interrupt by writing a "1" to the corresponding bit in this register. (In some cases, interrupts may be cleared by accessing a different register.)

bits 31-0 (0x001E\_bit15-0, 0x001C\_bit15-0)

The 32 bytes from 0x0000 to 0x001F described above comprise the configuration registers.

○ The area from 0x0020 to 0x3FF is used by each device.

<<Device Detection (Example)>>

When the device that exists in address 0x00620000 supports the bridge function, device detection is performed for the remaining areas as well. In this case, because expansion devices are not necessarily located in consecutive areas, detection must be performed for all areas.

If the device does not support the bridge function, only one device is connected, so device detection is not performed for any other area.

- (a) The detection start address is set as 0x00620000.
- (b) Read the address + 0x0000 and + 0x0002.
  - If the G2 identifier is found, a device is detected. →(c)
  - If the values that were read differ, no device is detected. →(z)
  - If a timeout occurs, no device is detected. →(z)
- (c) Read the address + 0x00C.
  - If the value is 0x0000, handling is unknown (a prototype). →(g)
  - If the value is not 0x0000, search for the device driver. →(d)
- (d) Search for the device driver.
  - If the driver is found, initialize the device accordingly. →(g)
  - If the device driver is not found, read the address + 0x000E. →(e)
- (e) Read the address + 0x000E.
  - If the value is 0x0000, handling is unknown. →(g)
  - If the value is not 0x0000, search for the device driver. →(f)
- (f) Search for the device driver.
  - If the driver is found, initialize the device accordingly. →(g)
  - If the device driver is not found, handling is unknown. →(g)
- (g) If the detected address is 0x00620000...
  - End if the device does not support the bus bridge function(when + 0x00A\_bit15 is "0"). →(z)
  - Search for the bridge destination if the device does support the bus bridge function. →(h)
  - If the detected address is not 0x00620000... →(h)
- (h) Repeat steps (b) through (f) 15 times, once for each 1K, starting from 0x00620400.
- (z) End

<<Device initialization processing (Example)>>

- (a) Write 0x0002 in the address + 0x0016, so that "1" can be read from the bits corresponding to the registers that can be set.
- (b) Read the address + 0x0010, + 0x0012, + 0x0014, + 0x0014, + 0x0018, and + 0x001A.
- (c) Write 0x0000 in the address + 0x0016 to return to normal operation.
- (d) Any value that was read in (b) that was not 0x0000 indicates a resource request that was being made, so allocate resources accordingly.
- (e) Write to the address + 0x0010, + 0x0012, + 0x0014, + 0x0014, + 0x0018, and + 0x001A, as necessary.
- (f) Write 0x0002 in the address + 0x0016, enabling the device.

<<G2 bus and expansion devices>>

The G2 bus is designed for about three devices to be connected, including a sound source IC and a modem, and the drive capabilities of the signal lines that are output from the main unit are limited. As a result, it is not possible to connect multiple external devices. If multiple external devices are connected, the signal lines must be buffered.

When connecting multiple devices by means of an expansion box, etc., connect them through a

device that has a repeater or bridge function. The connection diagrams below illustrate the connection of one expansion device and the connection of an expansion box.

[Connection between main unit and one expansion device]

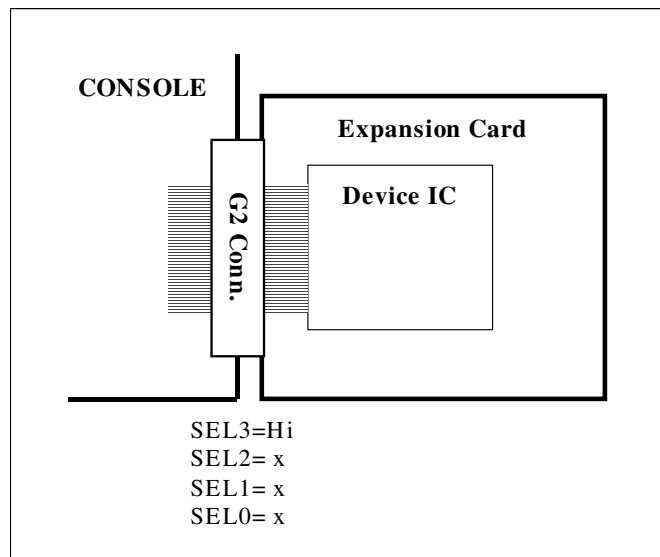


Fig. 4-4

[Connection between main unit and an expansion box]

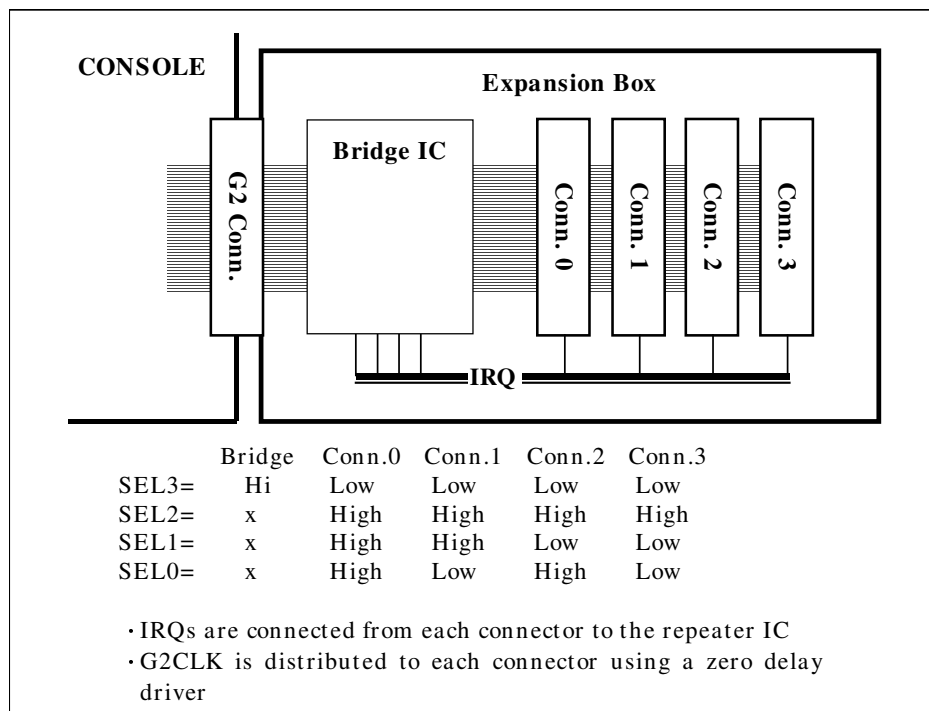


Fig. 4-5

## **§ 5 User Interface**

## § 5.1 Peripherals

The "Maple" peripheral interface is used for the Dreamcast control pads, etc. The "Maple" interface is described below.

### § 5.1.1 Overview

"Maple" is Sega's proprietary peripheral interface, and supports the connection of peripheral devices such as control pads and light guns through four ports. The Maple interface sends/receives serial data with the devices. The contents of the data are defined by the Maple Bus protocol. (The controller has no effect on the details of the protocol.) Protocol organization and analysis is handled by the SH4 CPU.

The hardware includes one port consisting of two lines, SDCKA and SDCKB, and data transfers are performed in synchronous serial mode. Data is transferred through half-duplex bi-directional transfer with a maximum data transfer rate of 2Mbps and a minimum data transfer rate of 250Kbps. The minimum data transfer unit is one frame, each of which begins with the START pattern that is indicated at the beginning of the data transfer, followed by a DATA pattern ranging in length from 4 to 1024 bytes, the parity bit, and then the END pattern. The eight parity bits are added automatically by the hardware when the data is sent, and are removed when the data is received.

The following register sets are provided for the Maple interface. (Details on each register are provided in the list of registers.)

- Maple-DMA Control Registers
- Maple I/F Block Control Registers
- Maple-DMA Secret Register
- Maple-DMA Debug Registers
- Maple I/F Block Hardware Control Register
- Maple I/F Block Hardware Test Registers
- Interrupt Control Registers      ...interrupt related registers (SB\_ISTNRM, etc.)

The basic operation of this interface is described below.

A command file is set up in system memory, containing the instructions (settings such as the communications port selection, the received data storage address, and the transfer data length) for the Maple controller and the transmission data. The command file consists of units formed by "instruction to the controller," "received data storage address," and "transmission data," in that order. Each of these units are located consecutively in system memory.

The controller can be started up by two methods: by software, or by hardware in synchronization with the V-BLANK signal. These methods are selected through the trigger selection register (SB\_MDTSEL). When startup by the V-BLANK signal is selected, delayed startup can be selected through the system register (SB\_MSYS) setting.

When the DMA enable register (SB\_MDEN) and the DMA start register (SB\_MDST) have been set by the SH4, the controller starts up and loads in the command file. The controller follows the instructions, sending the transmission data in system memory indicated by the DMA command table address register (SB\_MDSTAR) in the specified length to the target port, and then waits to receive a response. When data is received, the controller writes that data in system memory, starting from the received data store address that was set in the instructions. After receiving data, the controller continues executing the instructions in sequence until it detects the end of the command file. (Accesses between the controller and system memory are all performed through DMA in cho-DDT mode, and data is sent and received in units of 32 bytes.)

If, as a result of being disconnected or some other problem, the peripheral device does not respond (times out), then 0xFFFFFFFF is written to the first 32 bits of the received data storage address as "disconnected" processing. 0xFFFFFFF0 is written if a parity error occurs during reception of serial data.



- **Instruction Format**

bit 31	30-18	17-16	15-11	10-8	7-0
End Flag	000 0000 0000 00	Port Select	0000 0	Pattern	Transfer Length

Instructions to the Maple interface consist of 32 bits of data as shown above, and are set up in system memory.

An instruction consists of an End Flag bit, which indicates the end of the command file; the Port Select bits, which select the active port that is the target of the transmission/reception operation; the Pattern selection bits; and the Transfer Length bits.

When Maple detects a "1" in the End Flag bit, it terminates processing with this instruction. (The End Flag must be set to "1" in the last instruction in the transmission data.) When the pattern selection bits are set to "000," Maple outputs the data that is to be sent. If any other pattern is selected, the port outputs the information pattern only, and the transmission data length specification becomes invalid. "111" (NOP) is used to extend processing for a certain length of time. when the pattern "010" (Light-Gun mode) is selected, the End Flag bit of that instruction must be set to "1". All subsequent instructions are invalid until the pattern "100" (return from Light-Gun mode) is detected.

**End Flag:** Command file end bit

Setting	Meaning
0	Not end of command file
1	End of command file (Execution ceases after this command.)

**Port Select:** Port selection bits ... These bits select the port that is the target of the transmission/reception operation.

Setting	Selected port
0	Port A
1	Port B
2	Port C
3	Port D

**Pattern:** Pattern selection bits

Pattern			Pattern
bit2	bit1	bit0	
0	0	0	Normal data of the length indicated by Transfer Length
0	1	0	Light-Gun mode (Seizes SDCKB.)
0	1	1	RESET
1	0	0	Return from Light-Gun mode (Releases SDCKB.)
1	1	1	NOP (Waits after data is received before sending the next data.)

**Transfer Length:** Transfer data length selection bits

Setting	Transfer data length
0	4 Byte
1	8 Byte
:	:
0xFE	1020 Byte
0xFF	1024 Byte

- **Received data storage address**

Data is received from peripheral devices in 4-byte units, and is first loaded in a 32-byte reception FIFO. As soon as the FIFO becomes full, the data is transferred to the received data storage address in system memory. However, as soon as reception ends, even if the FIFO buffer is not full, an remaining data is regarded as invalid data and is transferred as 32 bytes.

The received data storage address area is from 0x00C00000 to 0x00FFFFE0 in system memory. (Specify "0" for the lower five bits of the address that indicates the 32 bytes that are transferred.)

- **Transmission data**

Transmission data consists of 4-byte units of data that are actually sent to a peripheral device by

the Maple protocol. The length of the data must be the transfer length (in 4-byte units) that is set by the instruction in the command file.

### § 5.1.2 Register Map

The registers that are used only by the Maple interface are in the area (from 0x005F 6C00 to 0x005F 6CFF) described in the system mapping table in section 2.1.

The mapping of the registers that are used by Maple is shown below. (Refer to section 8.4.1.2 for details on individual registers.)

Address	Register Name	Access	Function	Reset Initialize
Maple-DMA Control Registers (0x005F6C04, 0x005F6C14~18)				
0x005F 6C00	-		-	
0x005F 6C04	SB_MDSTAR	R/W	DMA Command Table Address	not initialize
0x005F 6C08	-		-	
0x005F 6C0C	-		-	
0x005F 6C10	SB_MDTSEL	R/W	DMA Trigger Selection	0
0x005F 6C14	SB_MDEN	R/W	DMA Enable	0
0x005F 6C18	SB_MDST	R/W	DMA Start / Status	0
Maple I/F Block Control Registers (0x005F6C80~84)				
0x005F 6C80	SB_MSYS	R/W	Maple System Control	0x3A980000
0x005F 6C84	SB_MST	R/-	Maple Status	
0x005F 6C88	SB_MSHTCL	/W	Maple Status Hard Trigger Clear	0
Maple-DMA Secret Register (0x005F6C8C)				
0x005F 6C8C	SB_MDAPRO	-/W	Maple Sys.Mem. Area Protection	0x00007Foo
Maple I/F Block Hardware Control Register (0x005F6CE8)				
0x005F 6CE8	SB_MMSEL	R/W	Maple MSB Selection	1
Maple-DMA Debug Registers (0x005F6CF4~FC)				
0x005F 6CF4	SB_MTXDAD	R/-	Maple TXD Address Counter	not initialize
0x005F 6CF8	SB_MRXDAD	R/-	Maple RXD Address Counter	not initialize
0x005F 6CFC	SB_MRXDBD	R/-	Maple RXD Base Address	not initialize

Table 5-1 Maple Register Map

\* In the above table, in the "Reset Initialize" column, "Not Initialized" indicates that the register value is undefined after a system reset. In all other cases, the value shown indicates the value that is set in that register after a system reset.

### § 5.1.3 Operating Sequence

There are two interface operating sequences: the normal sequence, and the "SDCKB seizure-release" sequence. Each sequence is described below.

#### <Normal Sequence>

The following chart shows the flow of data between the CPU, the peripheral controller, and the peripheral device.

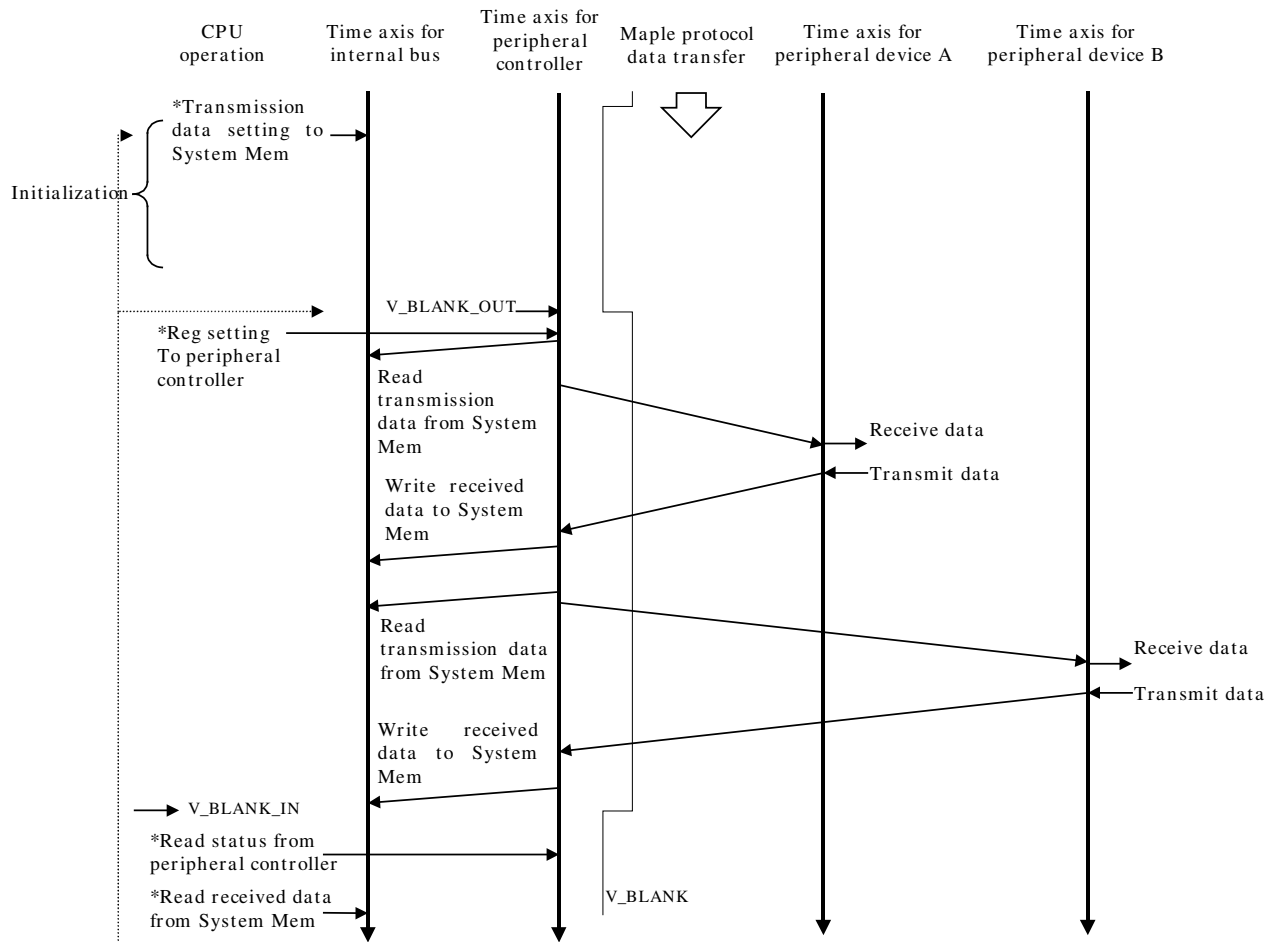


Fig. 5-1 Normal Sequence

**<SDCKB Seizure-Release Sequence>**

The SDCKB seizure-release sequence is used for latching the HV counter, primarily when using the Light Phaser Gun. The sequence is illustrated below.

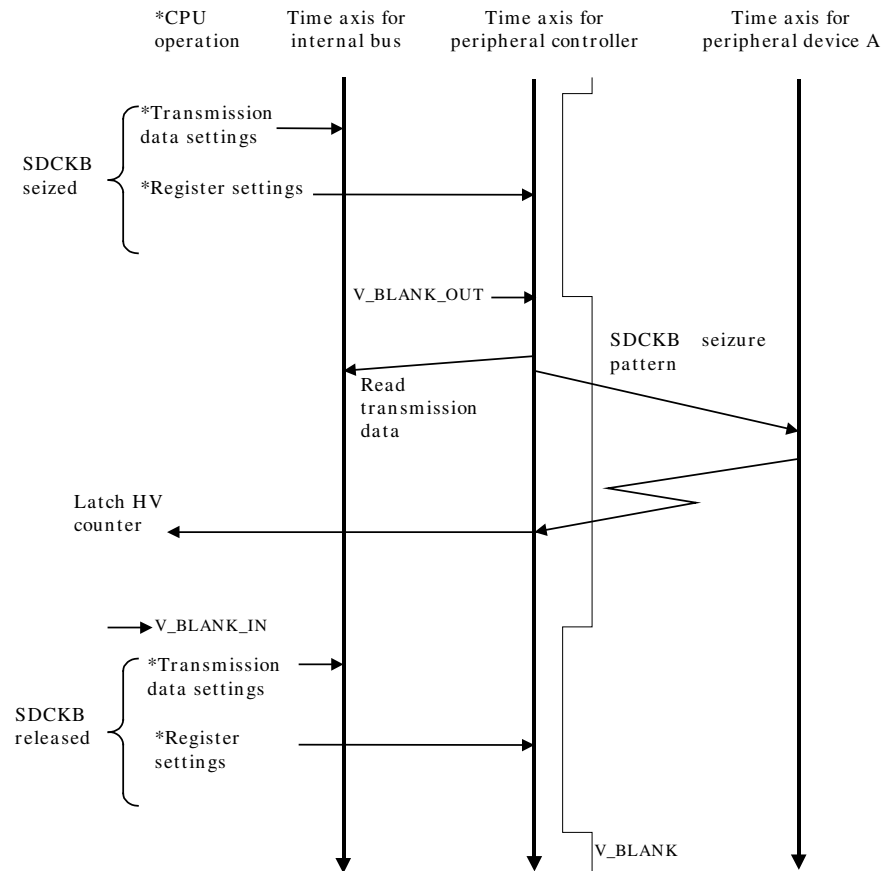


Fig. 5-2 SDCKB Seizure-Release Procedure

### § 5.1.4 Access Procedure

There are two access procedures, as described below: software initiated and hardware initiated.

- \* It is necessary for the initial settings for SH4-DMAC (setting DMA cho to DDT mode) to already have been made before initiating Maple-DMA. (Refer to the DMAC item in section 2.2.3.)

#### <<Software initiation>>

##### ~ Initialization ~

- (1) Work RAM area protection register setting  
Address: 0x005F 6C8C Write data: 0xXXXX XXXX
  - (2) System control register setting
  - (3) DMA trigger selection register settings  
Address: 0x005F 6C10 Write data: 0x00000000
- Initiation trigger → Software**

##### ~ Effective procedure ~

- (4) Data setting in system memory (DMA command table)  
Address: System memory area (0xnxxx nxxx) Write data: 0x8000 0000  
Send four bytes of data to port A and terminate  
  
Address: 0xnxxx nxxx + 4h Write data: System memory address (0xmxxx mxxx)  
Contents: Received data storage address (0xmxxx mxxx)  
  
Address: 0xnxxx nxxx + 8h Write data: 0xXXXX XXXX  
Contents: Data to be sent to port A 0xXXXX XXXX
- (5) DMA command table address register setting  
Address: 0x005F 6C04 Write data: 0xnxxx nxxx  
Contents: Starting address where transmission data is to be stored in system memory (0xnxxx nxxx)
- (6) DMA enable register setting  
Address: 0x005F 6C14 Write data: 0x0000 0001  
Contents: DMA enable
- (7) DMA start/status register setting  
Address: 0x005F 6C18 Write data: 0x0000 0001  
Contents: DMA initiation, transmission/reception start

##### ~ Confirmation of end ~

- (8) DMA start/status confirmation  
Address: 0x005F 6C18 Read data: 0x0000 0000 (transmission/reception end)  
Contents: Confirmation of transmission/reception end
- (9) Loading received data into system memory  
Address: 0xmxxx mxxx Read data: 0xXXXX XXXX  
Load data that was received from port A

<<Hardware initiation (auto-initiation at each trigger)>>

~Initialization~

1. System memory area protection register setting  
Address: 0x005F6C8C Write data: 0xFFFFFFFF
2. System control register setting  
Address: 0x005F6C80 Write data: 0x3A9800XX  
[Timeout: 300[micro]s; initiation at each V-Blank Out; transfer rate: 2Mbps; initiation delay setting: XX]
3. DMA trigger selection register setting  
Address: 0x005F6C10 Write data: 0x00000001  
Initiation trigger → Hardware trigger (V-Blank Out)

~Execution Procedure~

4. Setting of data in system memory (DMA command table)  
Address: System memory area 0xn timer Write data: 0x80000000  
Send four bytes of data to port A and terminate.  
Address: 0xn timer + 0x4: Write data system memory address (0xm timer)  
Received data store address (0xm timer)  
Address: 0xn timer + 0x8 Write data: 0xFFFFFFFF  
Data to be sent to port A: 0xFFFFFFFF
5. DMA command table address register setting  
Address: 0x005F6C04 Write data: 0xn timer  
Starting address in system memory where the transmission data is stored
6. DMA enable register setting  
Address: 0x005F6C14 Write data: 0x00000001  
DMA enabled
7. DMA start/status register setting  
Address: 0x005F6C18 Write data: 0x00000001  
DMA initiation, transmission/reception start

~Ending Confirmation~

8. DMA start/status confirmation  
Address: 0x005F6C18 Read data: Transmission/reception ends at 0x00000000  
Transmission/reception end confirmation
9. Loading received data into system memory  
Address: 0xm timer Read data: 0xFFFFFFFF  
Loading of received data from port A

### § 5.1.5 Example of Transmission and Reception Data

Examples of a transmission data command file stored in system memory and the corresponding reception data are shown below.

(16 bytes of data are sent to portA, and the received data is stored in address 0xC800000.)

Transmission data command file in system memory		
Address	Data	Contents
+0x0	0x03	Maple-Host Logic Command 32bit · PortA · Data 16Byte
+0x1	0x00	
+0x2	0x00	
+0x3	0x80	
+0x4	0x00	Recieve Data Destination Address 32bit
+0x5	0x00	
+0x6	0x80	
+0x7	0x0C	
+0x8	COMMAND	Protocol Data 8bit
+0x9	Destination AP	"
+0xA	Source AP	"
+0xB	Data Size	"
+0xC	DATA0	"
+0xD	DATA1	"
+0xE	DATA2	"
+0xF	DATA3	"
+0x10	DATA4	"
+0x11	DATA5	"
+0x12	DATA6	"
+0x13	DATA7	"
+0x14	Lower Byte0	" 16bit
+0x15	Upper Byte0	
+0x16	Lower Byte1	" 16bit
+0x17	Upper Byte1	

Table 5-3

Received data stored in system memory		
Address	Data	Contents
0xC800000	COMMAND	Protocol Data 8bit
0xC800001	Destination AP	"
0xC800002	Source AP	"
0xC800003	Data Size	"
0xC800004	DATA0	"
0xC800005	DATA1	"
0xC800006	DATA2	"
0xC800007	DATA3	"
0xC800008	DATA4	"
0xC800009	DATA5	"
0xC80000A	DATA6	"
0xC80000B	DATA7	"

---

---

B		
0xC80000 C	Lower Byte0	" 16bit
0xC80000 D	Upper Byte0	
0xC80000 E	Lower Byte1	" 16bit
0xC80000 F	Upper Byte1	

Table 5-3



### § 5.1.6 Notes Regarding Access

Notes that need to be observed in accesses concerning register settings and data transfers are described below.

#### <<Concerning Register Settings>>

- (1) If the controller is waiting for a single hardware trigger to clear, and then the initiation trigger is set to the software trigger and then back to a hardware trigger again, the hardware trigger that was set last is valid. (If DMA was not disabled at the moment that the switch was made to the hardware trigger, the trigger is not overwritten, and the wait for the single hardware trigger to clear becomes invalid.)
- (2) Regarding forced termination through the DMA enable register (*SB\_MDEN*) , when sending or receiving data, termination does not occur until transmission/reception on that port is completed. Therefore, it is possible for several DMA transfers to still occur after DMA is disabled. In addition, because a DMA end interrupt is not generated, the end must be detected only by polling the status. However, in the case of a forced termination as a result of an illegal error (for example, if system memory area protection was violated), the DMA ends at that point (when the error interrupt is generated).
- (3) The DMA trigger selection register (*SB\_MDTSEL*) and the system control register (*SB\_MSYS*) cannot be overwritten while DMA is enabled.
- (4) An illegal address error interrupt is generated when a value other than that specified by the system memory area protection register (*SB\_MDAPRO*) is written to the DMA command table address register (*SB\_MDSTAR*), and when an attempt is made to initiate DMA while in that state. An illegal address error interrupt is not generated when setting the received data store address that is written to system memory as a command, or when fetching a peripheral controller. An overrun error interrupt is generated when system memory is accessed. (It is not generated in the DMA write cycle.) The system control register cannot be overwritten while DMA is enabled.
- (5) The DMA start/status register (*SB\_MDST*) indicates that V-Blank Out initiated the operation during delayed initiation by the hardware trigger. Bit 31 of the status register (*SB\_MST*) indicates that operation is in progress based on the actual timing of transmission/reception after the delay. (This bit indicates that no operation is in progress from the time of V-Blank Out to the end of the delay.)
- (6) The system control register initiation delay setting is valid only for the hardware trigger, and is invalid for the software trigger.

### **Concerning data transfer**

- (1) When more than one frame of data (1024 bytes) has been sent, forced termination results and processing continues as if a parity error had occurred.
- (2) Repeated transmission/reception is possible by placing transmission commands consecutively in system memory. In addition, consecutive transmission/reception through the same port is possible by inserting several NOP instructions. (One instruction generates an interval of about 160[micro]s between accesses.)
- (3) Received data must be written in units of 32 bytes. If, for example, 36 bytes of data are received, valid data will be written in the first 36 bytes following the "received data storage address," and invalid data will be written in the remaining 28 bytes. Transmission commands can be stored consecutively in units of 4 bytes.
- (4) Regarding the reception buffer in system memory, the received data is asynchronous, and a maximum of 1024 bytes of data can be received. The length of the received data is normally controlled by the protocol, but it is possible that the actual length will exceed the intended length due to errors, etc. Therefore, important data should not be stored in the 1024 bytes after the final "received data storage address."
- (5) Data transfers between the peripheral controller and peripheral devices are performed in units of 32 bits, but the transmission data in this case is sent starting from the MSB (bit 31). Therefore, in a system that uses the Little Endian configuration, the data is sent starting from the MSB (bit 7) of the uppermost byte in four bytes of data, working down towards the lower bytes. In the same manner, received data is stored in units of 4 bytes, from the upper bytes to the lower bytes, starting with the data that was received first.
- (6) When a data transmission/reception spans a V-Blank, a V-Blank Over interrupt is generated, but the transmission/reception continues and the data is guaranteed.

## § 5.2 Control Pad

The standard controller device IDs and data format are shown below.

<Device ID>

The device ID starts from the first data as shown below.

0x00-0x00-0x00-0x01-0x00-0x06-0x0F-0xFE-0x00-0x00-0x00-0x00-0x00-0x00-0x00-0x00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
1st Data	0	0	0	0	0	0	0	0
2nd Data	0	0	0	0	0	0	0	0
3rd Data	0	0	0	0	0	0	0	0
4th Data	0	0	0	0	0	0	0	1
5th Data	0	0	0	0	0	0	0	0
6th Data	0	0	0	0	1	1	1	1
7th Data	0	0	0	0	0	1	1	0
8th Data	1	1	1	1	1	1	1	0
9th Data	0	0	0	0	0	0	0	0
10th Data	0	0	0	0	0	0	0	0
11th Data	0	0	0	0	0	0	0	0
12th Data	0	0	0	0	0	0	0	0
13th Data	0	0	0	0	0	0	0	0
14th Data	0	0	0	0	0	0	0	0
15th Data	0	0	0	0	0	0	0	0
16th Data	0	0	0	0	0	0	0	0

Table 5-4

<Read Data Format>

The data format size is 8 bytes.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
1st Data	Ra	La	Da	Ua	Start	A	B	1
2nd Data	1	1	1	1	1	X	Y	1
3rd Data	A1 <sub>7</sub>	A1 <sub>6</sub>	A1 <sub>5</sub>	A1 <sub>4</sub>	A1 <sub>3</sub>	A1 <sub>2</sub>	A1 <sub>1</sub>	A1 <sub>0</sub>
4th Data	A2 <sub>7</sub>	A2 <sub>6</sub>	A2 <sub>5</sub>	A2 <sub>4</sub>	A2 <sub>3</sub>	A2 <sub>2</sub>	A2 <sub>1</sub>	A2 <sub>0</sub>
5th Data	A3 <sub>7</sub>	A3 <sub>6</sub>	A3 <sub>5</sub>	A3 <sub>4</sub>	A3 <sub>3</sub>	A3 <sub>2</sub>	A3 <sub>1</sub>	A3 <sub>0</sub>
6th Data	A4 <sub>7</sub>	A4 <sub>6</sub>	A4 <sub>5</sub>	A4 <sub>4</sub>	A4 <sub>3</sub>	A4 <sub>2</sub>	A4 <sub>1</sub>	A4 <sub>0</sub>
7th Data	1	0	0	0	0	0	0	0
8th Data	1	0	0	0	0	0	0	0

Table 5-5

In the table, "Ra" indicates "right," "La" indicates "left," "Da" indicates "Down," and "Ua" indicates "Up."

1st: Digital button data (On = 0, Off = 1)

2nd: Digital button data (On = 0, Off = 1)

3rd: Analog axis 1 data (value of 0x00 ↔ 0xFF)

4th: Analog axis 2 data (value of 0x00 ↔ 0xFF)

5th: Analog axis 3 data (value of 0x00 ↔ 0x80 ↔ 0xFF)

6th: Analog axis 4 data (value of 0x00 ↔ 0x80 ↔ 0xFF)

7th: Analog axis 5 data (value of 0x00 ↔ 0x80 ↔ 0xFF)

8th: Analog axis 6 data (value of 0x00 ↔ 0x80 ↔ 0xFF)

<Write data format>

Because the target is a controller, there is no write data format. Writing data to the controller generates no response.

## § 5.3 Light Phaser Gun

## **§ 5.4 Backup (Option)**

## **§ 5.5 Sound Recognition (Option)**

## **§ 6 Peripheral Devices**

## § 6.1 DVE (Digital Video Encoder)

This system supports a variety of video modes: NTSC/PAL, VGA, etc.

The video mode is selected by plugging in the cable to the expansion A/V connector for the corresponding monitor type; the information is then reflected in the SH4's PIO port (the SH4\_PDTR register, "PB"). In response to that information, the SH4 sets the video mode setting registers in the HOLLY graphic core and the AICA, which sets the DVE. HOLLY and AICA then output RGB signals and video mode setting signals that correspond to that mode to the Digital Video Encoder (DVE) that actually generates the video signals that are sent to the external monitor.

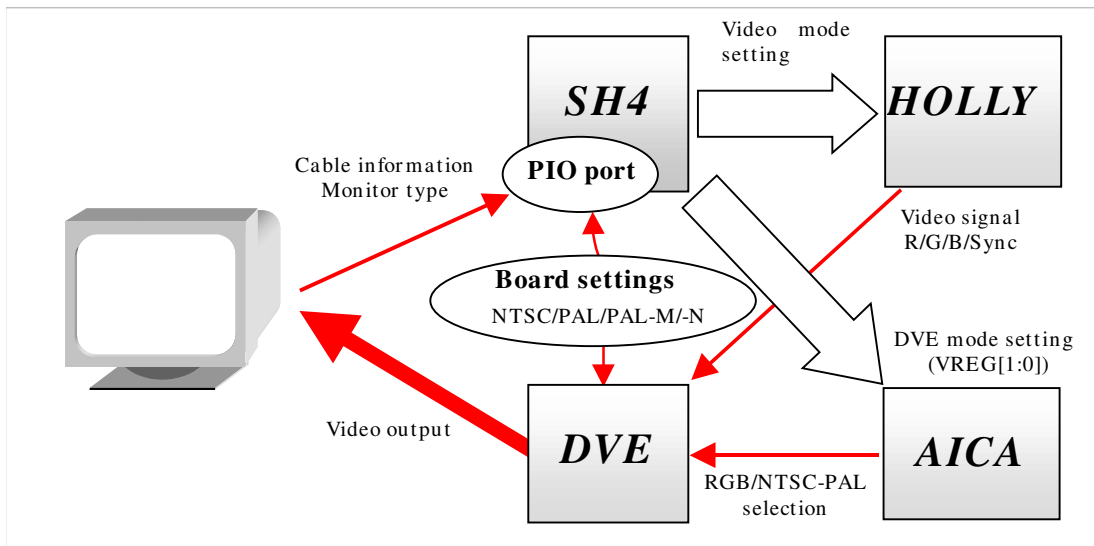


Fig. 6-1

\* For a list of video display modes, refer to section 3.1.3. for details on drawing CORE register settings, refer to section 8.4.2. For details on register settings for the DVE, refer to the explanation of common data/VREG in section 8.4.5.

Operation when the cable corresponding to the video mode in question is connected is described below. (Only stereo AV cables are supported as standard; cables marked with an asterisk (\*) are optional.)

Regarding the switching of modes, confirm the mode while the power is on; the system does not support changing cables while in operation. Therefore, if the cable connections are changed while the power is on, the screen will no longer be displayed normally.

### <When a VGA cable\* is connected>

1. The SH4 obtains the cable information from the PIO port. (PB[9:8] = "00")
2. Set the HOLLY synchronization register for VGA. (The SYNC output is H-Sync and V-Sync.)
3. When VREG<sub>1</sub> = 0 and VREG<sub>0</sub> = 0 are written in the AICA register, VIDEO<sub>1</sub> = 0 and VIDEO<sub>0</sub> = 1 are output. VIDEO<sub>0</sub> is connected to the DVE-DACH pin, and handles switching between RGB and NTSC/PAL.

### <When an RGB(NTSC/PAL) cable\* is connected>

1. The SH4 obtains the cable information from the PIO port. (PB[9:8] = "10")
2. Set the HOLLY synchronization register for NTSC/PAL. (The SYNC output is H-Sync and V-Sync.)
3. When VREG<sub>1</sub> = 0 and VREG<sub>0</sub> = 0 are written in the AICA register, VIDEO<sub>1</sub> = 1 and VIDEO<sub>0</sub> = 0 are output. VIDEO<sub>0</sub> is connected to the DVE-DACH pin, and handles switching between RGB and NTSC/PAL.

**<When a stereo A/V cable, an S-jack cable\* or an RF converter\* is connected>**

1. The SH4 obtains the cable information from the PIO port. (PB[9:8] = "11")
2. Set the HOLLY synchronization register for NTSC/PAL. (The SYNC output is H-Sync and V-Sync.)
3. When VREG1 = 1 and VREG0 = 1 are written in the AICA register, VIDEO1 = 0 and VIDEO0 = 0 are output. VIDEO0 is connected to the DVE-DACH pin, and handles switching between RGB and NTSC/PAL.

Among the video modes, the screen modes NTSC/PAL/PALM/PALN that are used in different countries are set through the DVE's PAL, **PALM-H** and **PALN-H** pins on the board. These settings are reflected as is in the SH4's PIO port (PB[4:2]), and the SH4 selects the screen mode by setting that information in HOLLY.

The following table shows the settings for the target DVE pins for the video modes used in different regions.

Region	Video mode	DVE pins (SH4 PIO port)		
		PALN-H (PB4)	PALM-H (PB3)	PAL (PB2)
Japan	NTSC	0	0	0
ASIA NTSC				
North America				
South Korea				
Europe	PAL(B,G,D,I)	0	0	1
Brazil	PAL-M(525)	0	1	1
Argentina	PAL-N	1	0	1
	Forced NTSC interlacing	1	1	0
	Forced PAL interlacing	1	1	1

Table 6-1

\*1 Because forced interlacing is set for the DVE, misoperation may result under some HOLLY settings.

\*2 Operation is not guaranteed for any combinations of pins settings that are not shown above.

For details concerning SECAM and other video modes, refer to the AV specifications.

## § 7 Debugger



Description pending

## **§ 8 Appendix**

## § 8.1 Technical Explanations

A supplemental explanation of the technologies that are used in this system is provided in this section.

### § 8.1.1 Technical Explanation Concerning Audio

#### § 8.1.1.1 Loop Control

The lop data and loop-related addresses are set as shown below.

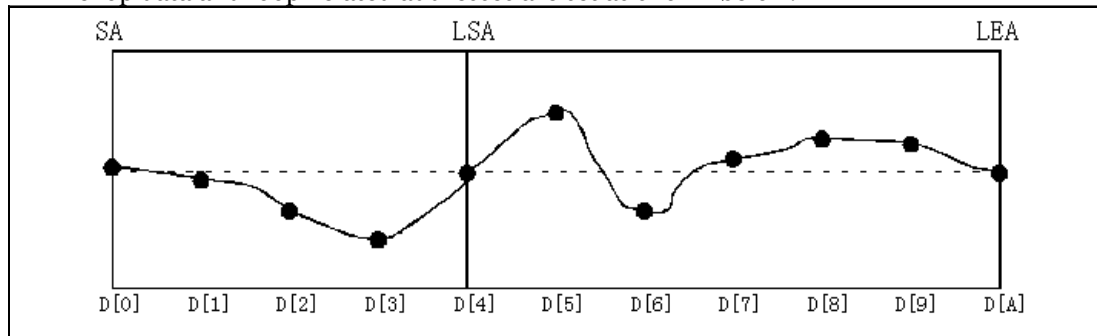


Fig. 8-1 Data Waveform

The setting for LSA is "0x3" and the setting for LEA is "0xA."

If SA is "0x100," sound memory ("wave memory") is allocated as shown below.

(Little Endian)

When PCMS = 2 (ADPCM)					When PCMS = 1			When PCMS = 0	
	15-12	11-8	7-4	3-0		15-8	7-0		15-0
0x100	D[3]	D[2]	D[1]	D[0]	0x100	D[1]	D[0]	0x100	D[0]
0x102	D[7]	D[6]	D[5]	D[4]	0x102	D[3]	D[2]	0x102	D[1]
0x104	--	D[A]	D[9]	D[8]	:			:	
					0x108	D[9]	D[8]	0x112	D[9]
					0x10A	--	D[A]	0x114	D[A]

Table 8-1 Sound Memory Allocation

Assuming that the sound data is read each time that it is sampled, the reading sequences in each loop mode are as shown below.

- Loop OFF  
D[0]→D[1]→D[2]→ . . . →D[A]
- Loop ON  
D[0]→D[1]→D[2]→ . . . →D[A]→D[5]→D[6]→ . . . →D[A]→D[5]→ . . .

<Notes on loop processing>

- In loop processing, the LSA and LEA data (in the case of ADPCM, the data after decoding) is processed based on the assumption that the values (in the case of ADPCM, the data after encoding) are the same. If necessary, set the data (before encoding) so that they will be the same value.
- If the pitch is increased for short loop data (waveform data in which there is only an extremely small amount of data corresponding to the loop from LSA to LEA), it is possible that the data corresponding to the loop portion will not be read even once. In this case, loop processing is not performed correctly. In order to permit the processing, and taking into consideration the effects of FNS, PLFO, etc., on pitch, it may be necessary to set the data so that  $LEA - LSA \geq OCT(\text{signed}) + 2$ .

<Notes concerning ADPCM long stream processing>

ADPCM references the previous data when it creates the next data.

- Set the lower two bits of LSA and LEA to "00".
- Set PCMS to "0x3".
- Just as with loop processing, set LSA so that the LSA data that is next in the stream is identical to the LEA data that is current in the stream.

0x0000	...	0xFFFF0					1st Stream
		0x0000	...	0xFFFF0			2nd Stream
				0x0000	...	0xFFFF0	3rd Stream
0x0000	...	0xFFFF0	...	0x1FFFE0	...	0x2FFD0	Data Stream

Table 8-2

### § 8.1.1.2 ADPCM

The audio IC that is used in the Dreamcast audio system uses ADPCM (Adaptive Differential Pulse Code Modulation) for its audio data compression method. The ADPCM system is a data compression system that prevents a loss of audio quality by encoding the differential between the audio data and the expected data according to a quantization width that adapts flexibly to changes in the waveform.

Because the ADPCM method stores the difference between the current sample and the data from the previous sample as the data, playback must begin at "key on" (the data in the start address). In addition, it is not possible to change the playback method (PCM or noise) or change the data (except during long sequence) while in the middle of playback.

#### Encoding Method

In the Dreamcast audio system, 4-bit ADPCM data is expanded into 16-bit PCM data. The encoding system follows the procedure described below.

- (1) Convert the data to be encoded into 16-bit PCM data for each sampling interval.

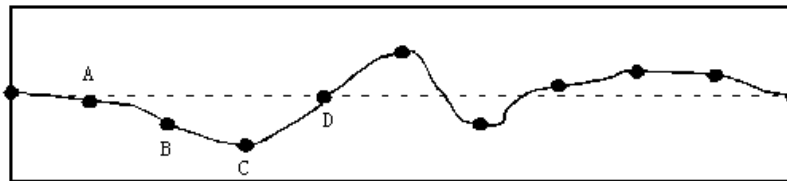


Fig. 8-2

- (2) Compare the PCM data at point B and the expected value at point B ( $X_n$ ), and determine the differential ( $dn$ ). If the differential is positive, the MSB ( $L_4$ ) of the ADPCM data becomes "0," and if the differential is negative, the MSB becomes "1."

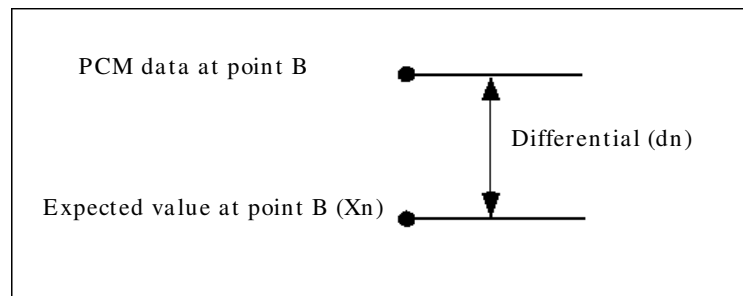


Fig. 8-3

- (3) Next, compare the quantization with ( $\Delta n$ ) and the absolute value of the differential (absolute value  $|dn|$ ), and determine the remaining three bits ( $L_3$ ,  $L_2$ , and  $L_1$ ) of the ADPCM data at point B from the ADPCM data correspondence table (Table 8-3).

- Example 1  
If the differential (absolute value  $|dn|$ ) is equal to the quantization with  $(\Delta n) \times 7/4$  (as shown in Fig. a below), the remaining three bits of the ADPCM data are  $L_3 = 1$ ,  $L_2 = 1$ , and  $L_1 = 1$ .
- Example 2  
If the differential absolute value  $|dn|$  is equal to the quantization with  $(\Delta n) \times 5/4$  (as shown in Fig. b below), the remaining three bits of the ADPCM data are  $L_3 = 1$ ,  $L_2 = 0$ , and  $L_1 = 1$ .

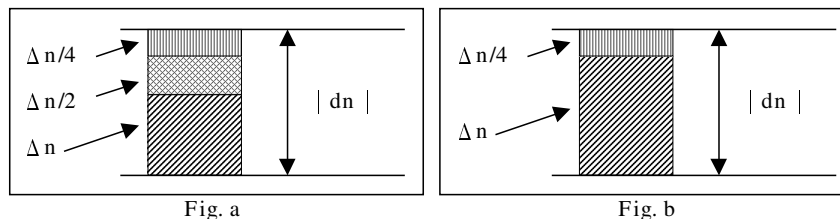


Fig. 8-4

- (4) After obtaining the ADPCM data for point B, derive the expected value ( $X_{n+1}$ ) and the quantization width ( $\Delta n + 1$ ) for the next point (point C) in order to derive ADPCM data

for point C.

- Point C expected value  $(X_n + 1) = (1 - 2 \times L_4) \times (L_3 + L_2/2 + L_1/4 + 1/8) \times \text{quantization width } (\Delta n) + \text{point B expected value}$
- Quantization width  $(\Delta n + 1) = f(L_3, L_2, L_1) \times \text{quantization width } (\Delta n)$

\* "f(L1, L2, L3) is a quantization width change factor from Table 8-4 below. The initial value for the expected value is 0, the initial value for the quantization width is 127, the minimum value for the quantization width is 127, and the maximum value for the quantization width is 24,576.

(5) Derive the rest of the ADPCM encoded data by repeating the above procedure.

L4		L3	L2	L1	Conditions
$dn \geq 0$	$dn \leq 0$				
0	1	0	0	0	$ dn  < \Delta n / 4$
		0	0	1	$\Delta n / 4 \leq  dn  < \Delta n / 2$
		0	1	0	$\Delta n / 2 \leq  dn  < \Delta n \times 3/4$
		0	1	1	$\Delta n \times 3/4 \leq  dn  < \Delta n$
		1	0	0	$\Delta n \leq  dn  < \Delta n \times 5/4$
		1	0	1	$\Delta n \times 5/4 \leq  dn  < \Delta n \times 3/2$
		1	1	0	$\Delta n \times 3/2 \leq  dn  < \Delta n \times 7/4$
		1	1	1	$\Delta n \times 7/4 \leq  dn $

Table 8-3 ADPCM Data Correspondence Table

L3	L2	L1	f
0	0	0	0.8984375
0	0	1	0.8984375
0	1	0	0.8984375
0	1	1	0.8984375
1	0	0	1.19921875
1	0	1	1.59765625
1	1	0	2.0
1	1	1	2.3984375

Table 8-4 Quantization Width Change Factor

### Decoding Method

- The decoding method derives the expected value and the quantization width through equations that are similar to those that are used during encoding. The procedure is described below.

- (1) Derive the decoded value ( $X_n$ ) for point B from the 4-bit ADPCM data, the quantization width ( $\Delta_n$ ), and the decoded value for point A ( $X_{n-1}$ ).

ADPCM DATA			
L4	L3	L2	L1

**Point B decoded value ( $X_n$ ) =  $(1 - 2 \times L4) \times (L3 + L2/2 + L1/4 + 1/8) \times$   
quantization width ( $\Delta_n$ ) + point A decoded value**

- (2) Update the quantization width ( $\Delta_{n+1}$ ) in order to derive the decoded value ( $X_{n+1}$ ) for the next point (point C).

**Quantization width ( $\Delta_{n+1}$ ) =  $f(L3, L2, L1) \times$  quantization width ( $\Delta_n$ )**

- (3) Decode the rest of the data by repeating the above procedure.

### § 8.1.1.3 AEG

- Effective rate and AEG change time

The rate of change in AEG changes according to the key scale value. After determining the effective rate from the equation shown below, use the table to find the actual change time that corresponds to that effective rate value.

The change times that are shown in the table below for the attack rate are for a change from -96dB to 0dB, while the other table is for a change from 0dB to -96dB.

$$\text{Effective rate} = (\text{KRS}[3:0] + \text{OCT}[3:0]) \times 2 + \text{FNS}[\text{bit } 9] + \text{Rate}[\text{register setting}] \times 2$$

- The ranges for each register are listed below:

KRS[3:0]: 0 to 15; OCT[3:0]: -8 to +7; FNS[9]: 0, 1; Rate [register setting]: 0 to 31

Attack State				Decay 1, Decay 2, Release State			
Effective rate	Change time [ms]	Effective rate	Change time [ms]	Effective rate	Change time [ms]	Effective rate	Change time [ms]
0	∞	32	47.	0	∞	32	690.
1	∞	33	38.	1	∞	33	550.
2	8100.	34	31.	2	118200.	34	460.
3	6900.	35	27.	3	101300.	35	390.
4	6000.	36	24.	4	88600.	36	340.
5	4800.	37	19.	5	70900.	37	270.
6	4000.	38	15.	6	59100.	38	230.
7	3400.	39	13.	7	50700.	39	200.
8	3000.	40	12.	8	44300.	40	170.
9	2400.	41	9.4	9	35500.	41	140.
10	2000.	42	7.9	10	29600.	42	110.
11	1700.	43	6.8	11	25300.	43	98.
12	1500.	44	6.0	12	22200.	44	85.
13	1200.	45	4.7	13	17700.	45	68.
14	1000.	46	3.8	14	14800.	46	57.
15	860.	47	3.4	15	12700.	47	49.
16	760.	48	3.0	16	11100.	48	43.
17	600.	49	2.4	17	8900.	49	34.
18	500.	50	2.0	18	7400.	50	28.
19	430.	51	1.8	19	6300.	51	25.
20	380.	52	1.6	20	5500.	52	22.
21	300.	53	1.3	21	4400.	53	18.
22	250.	54	1.1	22	3700.	54	14.
23	220.	55	0.93	23	3200.	55	12.
24	190.	56	0.85	24	2800.	56	11.
25	150.	57	0.65	25	2200.	57	8.5
26	130.	58	0.53	26	1800.	58	7.1
27	110.	59	0.44	27	1600.	59	6.1
28	95.	60	0.40	28	1400.	60	5.4
29	76.	61	0.35	29	1100.	61	4.3
30	63.	62	0.0	30	920.	62	3.6
31	55.	63	0.0	31	90.	63	3.1

Change time from -96dB to 0dB

Change time from 0dB to -96dB

Table 8-5



#### § 8.1.1.4 PG

- OCT[3:0] setting  
Specify the octave in two's complement format. Values shown in parentheses are one octave higher in ADPCM.

OCT	8	9	0xA	0xB	0xC	0xD	0xE	0xF	0	1	2	3	4	5	6	7
Interval	-8	-7	-6	-5	-4	-3	-2	-1	0	+1	(+2)	(+3)	(+4)	(+5)	(+6)	(+7)

Table 8-6

- FNS and OCT settings (example for the F number table setting when the C4 note is sampled at 44.1KHz)

$$\text{FNS (dec)} = 2^{10} \times (2^{(P/1200)} - 1)$$

Note	Note number	Pitch P[CENT]	FNS[9:0] (dec)	FNS[9:0] (hex)	OCT[3:0] (hex)
B3	59	-100	909.1	0x38D	0xF
C4	60	0	0.0	0	0
C4#	61	100	60.9	0x3D	0
D4	62	200	125.4	0x7D	0
D4#	63	300	193.7	0xC2	0
E4	64	400	266.2	0x10A	0
F4	65	500	342.9	0x157	0
F4#	66	600	424.2	0x1A8	0
G4	67	700	510.3	0x1FE	0
G4#	68	800	601.5	0x25A	0
A4	69	900	698.2	0x2BA	0
A4#	70	1000	800.6	0x321	0
B4	71	1100	909.1	0x38D	0
C5	72	0	0.0	0	1

Table 8-7

### § 8.1.1.5 LFO

- LFOF[4:0] oscillation frequencies
 

0x00 ⇒ 0.17 Hz	0x10 ⇒ 2.87 Hz
0x01 ⇒ 0.19 Hz	0x11 ⇒ 3.31 Hz
0x02 ⇒ 0.23 Hz	0x12 ⇒ 3.92 Hz
0x03 ⇒ 0.27 Hz	0x13 ⇒ 4.79 Hz
0x04 ⇒ 0.34 Hz	0x14 ⇒ 6.15 Hz
0x05 ⇒ 0.39 Hz	0x15 ⇒ 7.18 Hz
0x06 ⇒ 0.45 Hz	0x16 ⇒ 8.6 Hz
0x07 ⇒ 0.55 Hz	0x17 ⇒ 10.8 Hz
0x08 ⇒ 0.68 Hz	0x18 ⇒ 14.4 Hz
0x09 ⇒ 0.78 Hz	0x19 ⇒ 17.2 Hz
0x0A ⇒ 0.92 Hz	0x1A ⇒ 21.5 Hz
0x0B ⇒ 1.10 Hz	0x1B ⇒ 28.7 Hz
0x0C ⇒ 1.39 Hz	0x1C ⇒ 43.1 Hz
0x0D ⇒ 1.60 Hz	0x1D ⇒ 57.4 Hz
0x0E ⇒ 1.87 Hz	0x1E ⇒ 86.1 Hz
0x0F ⇒ 2.27 Hz	0x1F ⇒ 172.3 Hz
- ALFO waveform according to ALFOWS[1:0]
- PLFO waveform according to PLFOWS[1:0]

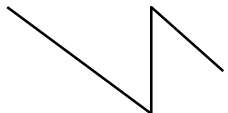
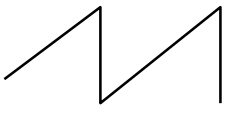
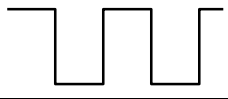
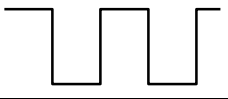
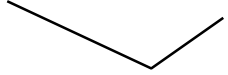
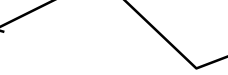
ALFOWS	AM modulation (ALFO)			PLFOWS	PM modulation (PLFO)		
	Volume	ALFO[7:0]			Pitch	PLFO[7:0]	
0	- 0 dB	0 0xFF		0	+ 0 -	0x7F 00 0x80	
1	- 0 dB	0 0xFF		1	+ 0 -	0x7F 00 0x80	
2	- 0 dB	0 0xFF		2	+ 0 -	0x7F 00 0x80	
3	- 0 dB	0 0xFF	***** ***Noise*** *****	3	+ 0 -	0x7F 00 0x80	***** ***Noise*** *****

Table 8-8

- Degree of mixing according to ALFOS[2:0]
- Degree of effect on pitch of PLFOS[2:0]

ALFOS	Mixing to EG	PLFOS	Effect on pitch
0	No effect	0	No effect
1	- 0.4dB displacement	1	- 3 - + 2 CENT displacement
2	- 0.8dB displacement	2	- 7 - + 5 CENT displacement
3	- 1.5dB displacement	3	- 14 - + 12 CENT displacement
4	- 3dB displacement	4	- 27 - + 25 CENT displacement
5	- 6dB displacement	5	- 55 - + 52 CENT displacement
6	- 12dB displacement	6	- 112 - + 103 CENT displacement
7	- 24dB displacement	7	- 231 - + 202 CENT displacement

Table 8-9

#### § 8.1.1.6 Mixer

A block diagram of the mixer section is shown below.

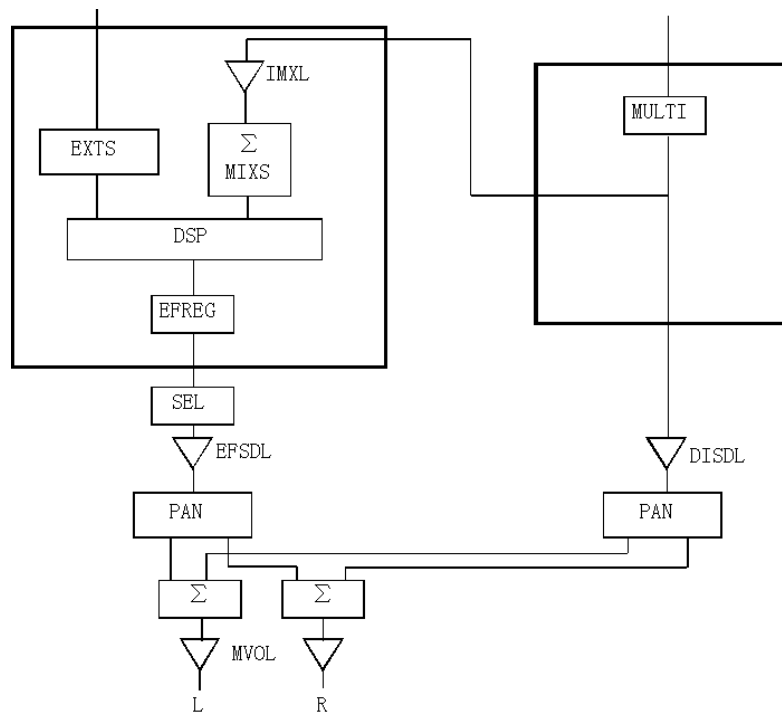


Fig. 8-5

The correspondence between the register value and the volume is shown below.

**TL[7:0]**

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Volume	-48dB	-24dB	-12dB	-6dB	-3dB	-1.5dB	-0.8dB	-0.4dB

Table 8-10

**IMXL[3:0], DISDL[3:0], EFSDL[3:0], MVOL[3:0]**

Register value	Volume
0	-MAXdB
1	-42dB
2	-39dB
:	:
0xD	-6dB
0xE	-3dB
0xF	0dB

Table 8-11

**DIPAN[4:0], EFPAN[4:0]**

Register value	L	R
0	0dB	0dB
1	-3dB	0dB
2	-6dB	0dB
:	:	:
0xD	-39dB	0dB
0xE	-42dB	0dB
0xF	-MAXdB	0dB
0x10	0dB	0dB
0x11	0dB	-3dB
0x12	0dB	-6dB
:	:	:
0x1D	0dB	-39dB
0x1E	0dB	-42dB
0x1F	0dB	-MAXdB

Table 8-12

**Correspondence between the slot that should be set in EFSDL and EFPAN and the effect source**

Slot	Output mixer source data
0~0xF	EFREG[0]~EFREG[15]
0x10	EXTS[0]: Digital audio 1L
0x11	EXTS[0]: Digital audio 1R

Table 8-13

### § 8.1.1.7 FEG

(Time variation filter)

Using the IIR filter, it is possible to direct sound on each channel through an LPF.  
Using a dedicated EG, time variation for the LPF cutoff frequency becomes possible.  
The LPF permits setting of a fixed (time does not vary) Q (resonance) for each channel.  
For details on "Q," refer to section 8.4.5, "AICA Registers."

- Effective rate and FEG change time  
Effective rate =  $(KRS[3:0] + OCT[3:0]) \times 2 + FNS[9] + (Rate[register\ setting]) \times 2$   
(KRS[3:0]: +0 to +0xF; OCT[3:0]: -8 to +7; FNS[9]: +0, +1; Rate [register setting]: +0 to +0x1F)

Effective rate	Change time [ms]	Effective rate	Change time [ms]
0	∞	32	2760.
1	∞	33	2200.
2	472800.	34	1840.
3	405200.	35	1560.
4	354400.	36	1360.
5	283600.	37	1080.
6	236400.	38	920.
7	202800.	39	800.
8	177200.	40	680.
9	142000.	41	560.
10	118400.	42	440.
11	101200.	43	392.
12	88800.	44	340.
13	70800.	45	272.
14	59200.	46	228.
15	50800.	47	196.
16	44400.	48	172.
17	35600.	49	34.
18	29600.	50	136.
19	25200.	51	100.
20	22000.	52	88.
21	17600.	53	72.
22	14800.	54	56.
23	12800.	55	48.
24	11200.	56	44.
25	8800.	57	34.
26	7200.	58	28.
27	6400.	59	24.
28	5600.	60	22.
29	4400.	61	17.
30	3680.	62	14.
31	3160.	63	12.

Change Time from 0x0008 to 0xFF8

Table 8-14



described below.

**RBL[1:0]** (W): Specifies the length of the ring buffer.

- 0 : 8K words
- 1 : 16K words
- 2 : 32K words
- 3 : 64K words

**RBP[22:11]**(W): Specifies the starting address of the ring buffer (at a 4K word boundary).

(Generation of the modulation waveforms used in the DSP)

There are three means for generating the modulation wave signals that are used by the DSP:

1. The CPU writes the modulation wave into the DSP's memory (COEF).
2. Store the modulation wave data in wave memory ("sound memory," in the diagram), lower the pitch, and use the data buffered in MIXS as the modulation wave.
3. The CPU writes the modulation wave into the DSP's internal buffer (MEMS).
  - Option 1 offers 13-bit precision and adds to the load on the CPU, but permits the creation of any waveform that is desired.
  - Option 2 offers 16-bit precision, and permits the amplitude and pitch to be changed through the EG and LFO. (However, if SDIR = 1, then EG = 0x000, ALFOS = 0x0, and TL = 0x00; however, the precision increases to the equivalent of 20-bit precision.)
  - Option 3 offers 24-bit precision and adds to the load on the CPU, but permits the creation of any waveform that is desired.

(DSP's internal RAM)

**MIXS[19:0]** (R/W): Sound data buffer from the input mixture (number of data items: 16)

(Note) Writing to MIXS[19:0] is used for LSI testing purposes.

Writes that are not performed in test mode are invalid for the following reasons:

- Regardless of the register settings, only data written from the sound source is valid.
- Second-generation data is retained for the purpose of integrating all of the slots, but it is not possible to specify the generation when accessing this buffer.

**EXTS[15:0]** (R): Digital audio input data buffer (number of data items: 2)

**MEMS[23:0]** (R/W) : Wave memory input data buffer (number of data items: 32)

(Actual writes to MEMS[7:0] are executed simultaneously with writes to MEMS[23:16].)

Only one of the above three buffers can be selected by the DSP program as the input data INPUTS. Differences in bit length are handled by shifting the data left.

All three of the above buffers permit access from the CPU; the access timing is described below. (Timing: T<sub>0</sub> & T<sub>1</sub>, T<sub>2</sub> & T<sub>3</sub>, ... are equivalent to one step for the DSP.)

	T <sub>0</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>	T <sub>5</sub>	T <sub>6</sub>	T <sub>7</sub>
MIXS	DSPR	**IMXRD**	DSPR	DMSP	DSPR	**IMXWT**	DSPR	DMSP
EXTS	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP
MEMS	DSPR	DMSP	DSPR	DMSP/DSPW	DSPR	DMSP	DSPR	DMSP/DSPW

DMSP: Read/Write by DMA, SH4: ARM    DSPR: Read by DSP    DSPW: Write by DSP

IMXRD: Read MIXS.    IMXWT: Write to MIXS.

(Note) The access request to MIXS by the DMSP in T<sub>1</sub> and T<sub>5</sub> is delayed.

(Note) Because T<sub>2</sub> & T<sub>3</sub> and T<sub>6</sub> & T<sub>7</sub> represent the sound memory read timing for PCM sound data, DSP access is not possible. Therefore, wave memory access requests must be coded on odd-numbered steps (line 2, line 4, line 6, etc.).

Table 8-15

**TEMP[23:0]** (R/W) : DSP work buffer (number of data items: 128)

This has a ring buffer configuration; the pointer is decremented by "1" for each sample.

**COEF[12:0]** (R/W) : DSP coefficient buffer (number of data items: 128)

(Note) In order to maintain compatibility in the future if the data width is expanded to 16 bits, write zeroes to the lower three bits that are undefined in the register map.

**MADRS[16:1]**(R/W) : DSP address buffer (number of data items: 64)

**MPRO[63:0]** (R/W) : DSP microprogram buffer (number of data items: 128)

**EFREG[15:0]** (R/W) : DSP output buffer (number of data items: 16)

All five of the above buffers permit access from the CPU; the access timing is described below.  
(Timing: T<sub>0</sub> & T<sub>1</sub>, T<sub>2</sub> & T<sub>3</sub>, ... are equivalent to one step for the DSP.)

	T <sub>0</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>	T <sub>5</sub>	T <sub>6</sub>	T <sub>7</sub>
TEMP	DSPR	DMSP/DSPW	DSPR	DMSP/DSPW	DSPR	DMSP/DSPW	DSPR	DMSP/DSPW
COEF	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP
MADRS	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP
MPRO	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP	DSPR	DMSP
EFREG	MIXR	DMSP/DSPW	----	DMSP/DSPW	----	DMSP/DSPW	----	DMSP/DSPW

MIXR: Read by Output MIXTER.

Table 8-16

An overview of the DSP program (total: 55 bits) is provided below.

**MASA[5:0]** : Specifies the MADRS read address.

**IWA[4:0]** : Specifies the write address for the input data (INPUTS).

**IWT**: DSP input data write request.

**IRA[5:0]** : Specifies the read address for the input data (INPUTS).

INPUTS Map (Addresses are DSP Program Addresses)

Address (hex)	Contents of INPUTS
0x00~0x1F	MEMS
0x20~0x2F	MIXS
0x30	EXTSo(L)
0x31	EXTSo(R)
0x32~0x37	Future expansion (cannot be set)
0x38~0x3F	Undefined (cannot be set)

Table 8-17

**TWA[6:0]** : Specifies the TEMP write address.

**TWT**: TEMP input data write request.

**TRA[6:0]** : Specifies the TEMP read address.

**EWA[3:0]** : Specifies the output EFREG address.

**EWT**: Request to write output data to EFREG.

**BSEL** : 0 = TEMP data select; 1 = accumulator select

**ZERO** : 1 = Assume the adder input as "0."



INPUTS[23:4].

**YSEL1 :** Multiplier Y input select 1

YSEL1	YSEL0	Selected input
0	0	FRC_REG
0	1	COEF
1	0	Y_REG[23:11]
1	1	"0"   Y_REG[15:4] (MSB is "0")

Table 8-18

1 = INPUTS data select

(Note) Memory access-related flags (MRD, MWT, NOFL, TABLE, NXADR, ADREB, and MASA[4:0]) may only exist in odd steps (line 2, 4, 6, etc.) of the microprogram.

Set to "1" when storing linear format data in wave memory.

MDEC\_CT is decremented by one for each sample; when its value reaches "0," a value that corresponds with the loop length specified by RBL is loaded into MDEC\_CT.

NXADR is used in primary interpolation mode in order to interpolate adjacent values.

This is used when writing data to a ring buffer, etc.

SHFT1: Shifter control 1

SHFT1	SHFT0	Shift amount	In event of an overflow
0	0	$\times 1$	Protected
0	1	$\times 2$	Protected
1	0	$\times 2$	Not protected
1	1	$\times 1$	Not protected

Table 8-19

The data that is selected by F\_SEL and A\_SEL in interpolation mode (SHFT1 = SHFT0 = 1) and non-interpolation mode (SHFT1 ≠ 1 and SHFT0 ≠ 1), respectively, is shown below.

F_SEL			A_SEL		
Register output	Non-interpolation mode	Interpolation mode	Register output	Non-interpolation mode	Interpolation mode

FRC_REG12	SFTREG23	'o'	ADRS_REG11	INPUTS23	SFTREG23
FRC_REG11	SFTREG22	SFTREG11	ADRS_REG10	INPUTS23	SFTREG22
FRC_REG10	SFTREG21	SFTREG10	ADRS_REG9	INPUTS23	SFTREG21
FRC_REG9	SFTREG20	SFTREG9	ADRS_REG8	INPUTS23	SFTREG20
FRC_REG8	SFTREG19	SFTREG8	ADRS_REG7	INPUTS23	SFTREG19
FRC_REG7	SFTREG18	SFTREG7	ADRS_REG6	INPUTS22	SFTREG18
FRC_REG6	SFTREG17	SFTREG6	ADRS_REG5	INPUTS21	SFTREG17
FRC_REG5	SFTREG16	SFTREG5	ADRS_REG4	INPUTS20	SFTREG16
FRC_REG4	SFTREG15	SFTREG4	ADRS_REG3	INPUTS19	SFTREG15
FRC_REG3	SFTREG14	SFTREG3	ADRS_REG2	INPUTS18	SFTREG14
FRC_REG2	SFTREG13	SFTREG2	ADRS_REG1	INPUTS17	SFTREG13
FRC_REG1	SFTREG12	SFTREG1	ADRS_REG0	INPUTS16	SFTREG12
FRC_REG0	SFTREG11	SFTREG0			

\* Interpolation mode is used when high-precision processing is required, such as when changing the pitch.

Table 8-20

Example of how to implement a ring buffer and a filter table in a DSP

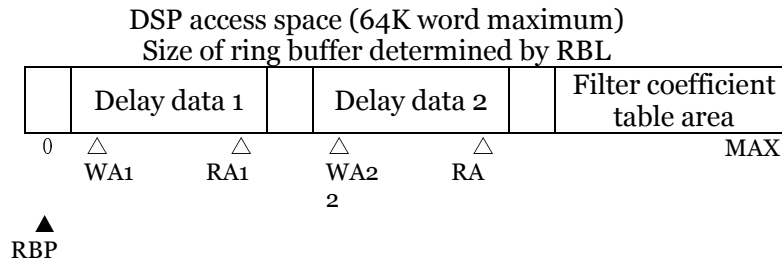


Fig. 8-7 DSP Access Space

- WA1 is the write address for delay data 1, when ADREB = 0.
- RA1 is the read address for delay data 1; when ADREB = 0, a delay of fixed duration is obtained, and when ADREB = 1, the data that accompanied the delay time change equivalent to the change in ADRS\_REG is obtained.
- WA2 and RA2 apply to delay data 2, and must reside apart from delay data 1. Especially when conducting a memory read with ADREB = 1, both must be kept apart, giving due consideration to the amount of change in the address.
- The ring buffer area is accessed with TABLE = 0.  
In this case, if the relative access address ( $\text{MADRS}[16:1] + \text{ADRS\_REG}[16:1] (+1)$ ) exceeds the size of the ring buffer, the relative address wraps around to "0." (However, given that the size of the ring buffer is subject to change, using the ring buffer without the wraparound feature is recommended.)  
(Supplement) In this case, the actual access address is expressed below:  
Access address:  $\text{MADRS}[16:1] + \text{ADRS\_REG}[16:1] + \text{MDEC\_CT}[16:1] (+1)$
- The filter coefficient table area is accessed with TABLE = 1.  
In this case, even if the relative access address ( $\text{MADRS}[16:1] + \text{ADRS\_REG}[16:1] (+1)$ ) exceeds the size of the ring buffer, the relative address does not wrap around. (However, the maximum size is 64K words.)

### § 8.1.2 Reset Sequence

The reset sequences for the Dreamcast System are illustrated in the following charts.

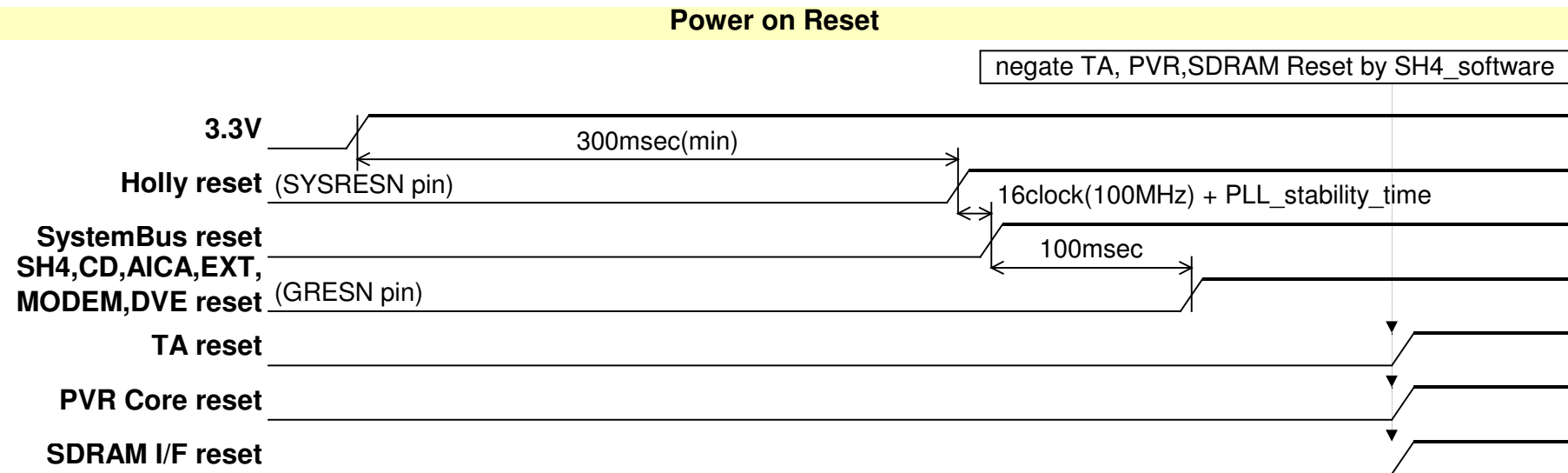


Fig. 8-8 Reset Sequence (Power On Reset)

\* SYSRESN and GRESN in the above chart are both pins on the HOLLY IC, which is the graphics system core.

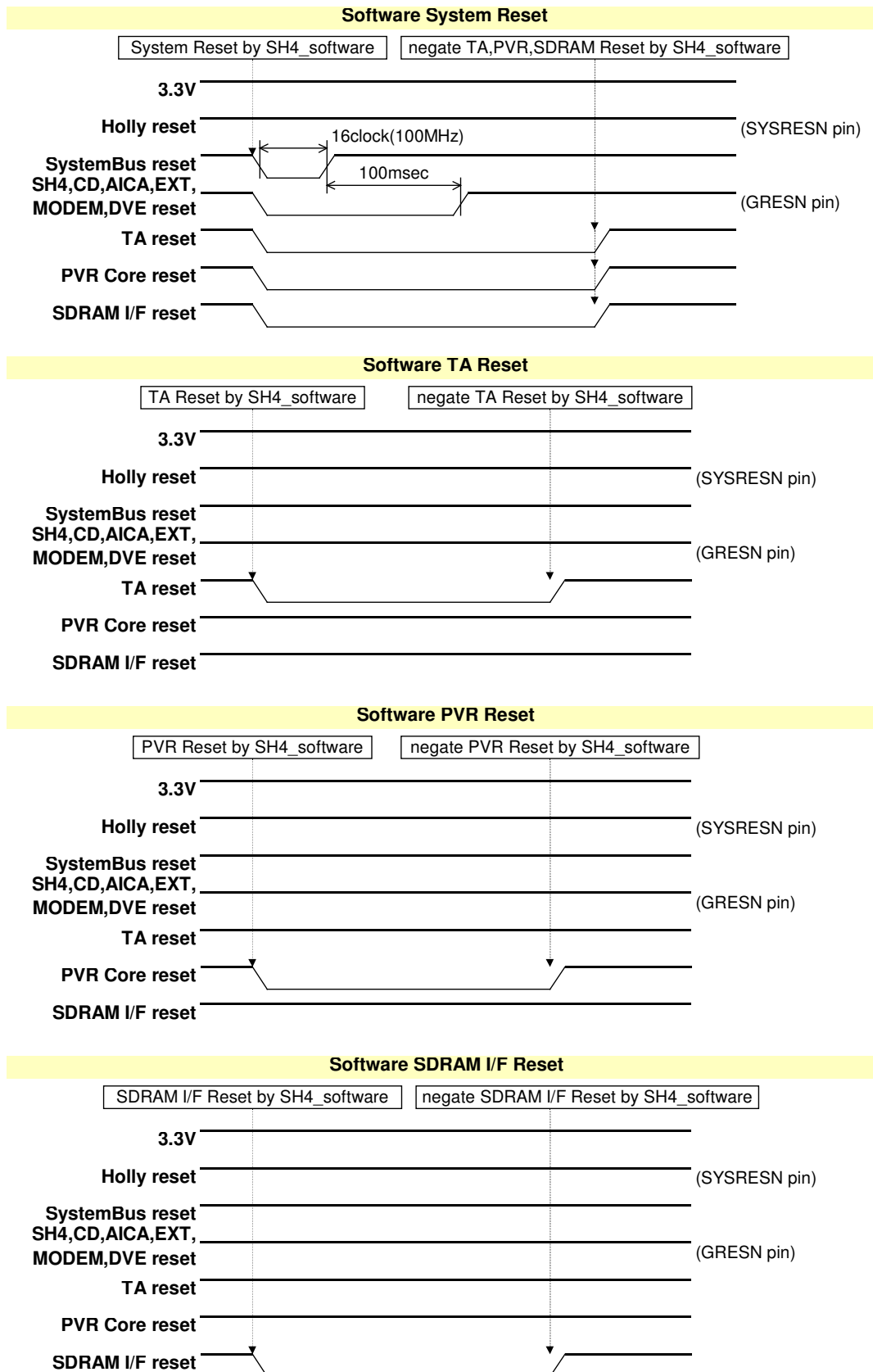


Fig. 8-9 Reset Sequence (Software Reset)

The **Dreamcast** System has two reset sequences: power-on reset (Fig. 8-8) and software reset (Fig. 8-9). Each is explained below.

- **Power-On-Reset**

- (1) Because each reset signal is cleared when the system power is turned on, each block and device remains in the reset state. While the SH4 is in the reset state, a 33MHz clock signal is input to the SH4 from an external PLL. In response to this signal, the SH4's internal PLL outputs a 100MHz clock signal to the HOLLY chip.
- (2) Once an interval of about 300msec elapses after the power signal rises after power-on, the Reset IC releases the reset state for the Holly graphics/peripheral core. (Pin: SYSRESN "L" → "H")
- (3) Once the HOLLY's reset state is released in step 2, then after the stabilization time for the HOLLY's internal PLL elapses and the PLL generates 16 100MHz clock pulses, the reset state for the HOLLY's internal "system bus" is released (followed by the interfaces, including the SH4 interface, the PVR interface, the G1 interface, etc.).
- (4) Approximately 100msec later, Reset Control in the System Bus Block releases the reset state for various system devices: the SH4, the SDRAM system memory, the GD-ROM, devices on the G2 Bus such as the AICA audio IC, and the Digital Video Encoder (video output). (Pin: GRESN "L" → "H")
- (5) Once the reset state for the SH4 (the CPU) is released, the SH4 releases the reset state for the TA (Tile Accelerator), the PVR core, and the SDRAM interface in the HOLLY chip by setting their respective reset bits.

- **Software-Reset**

- (1) The SH4 (the CPU) can apply a reset to the entire system through software. The reset is initiated for HOLLY's internal "system bus" (followed by the interfaces, including the SH4 interface, the PVR interface, the G1 interface, etc.) by accessing HOLLY's SB\_SFRES (0x005F6890).
- (2) Once a reset has been applied to the System Bus and all devices in the system are in the reset state, then after the HOLLY's internal PLL generates 16 100MHz clock pulses, the reset state that was established in step 1 is released.
- (3) After the System Bus reset state has been released, the reset state for various devices such as the SH4 and the GD-ROM is released during a period of approximately 100msec, as described in step 4 of the power-on reset sequence. Then, the software releases the reset state for the PVR core, the TA and the SDRAM interface in the HOLLY chip.
- (4) The SH4 can also initiate and release the reset state for the PVR core, the TA and the SDRAM interface in the HOLLY chip individually by setting their respective Reset bits.

Fig. 8-10 is a relational diagram between the reset signals and the clock, and Fig. 8-11 shows the hardware reset system.

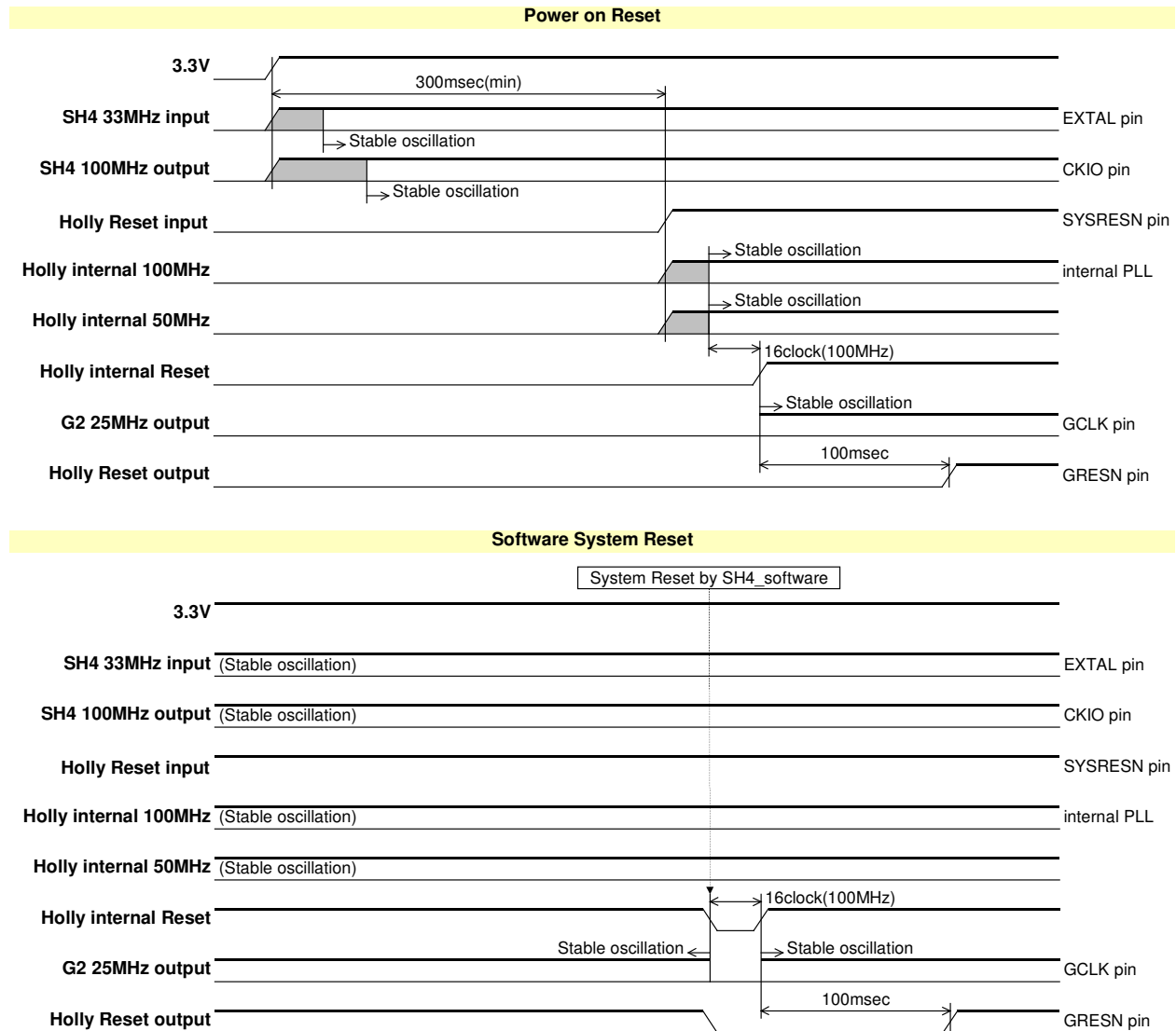


Fig. 8-10 Relational Diagram between the Reset Signals and the Clock

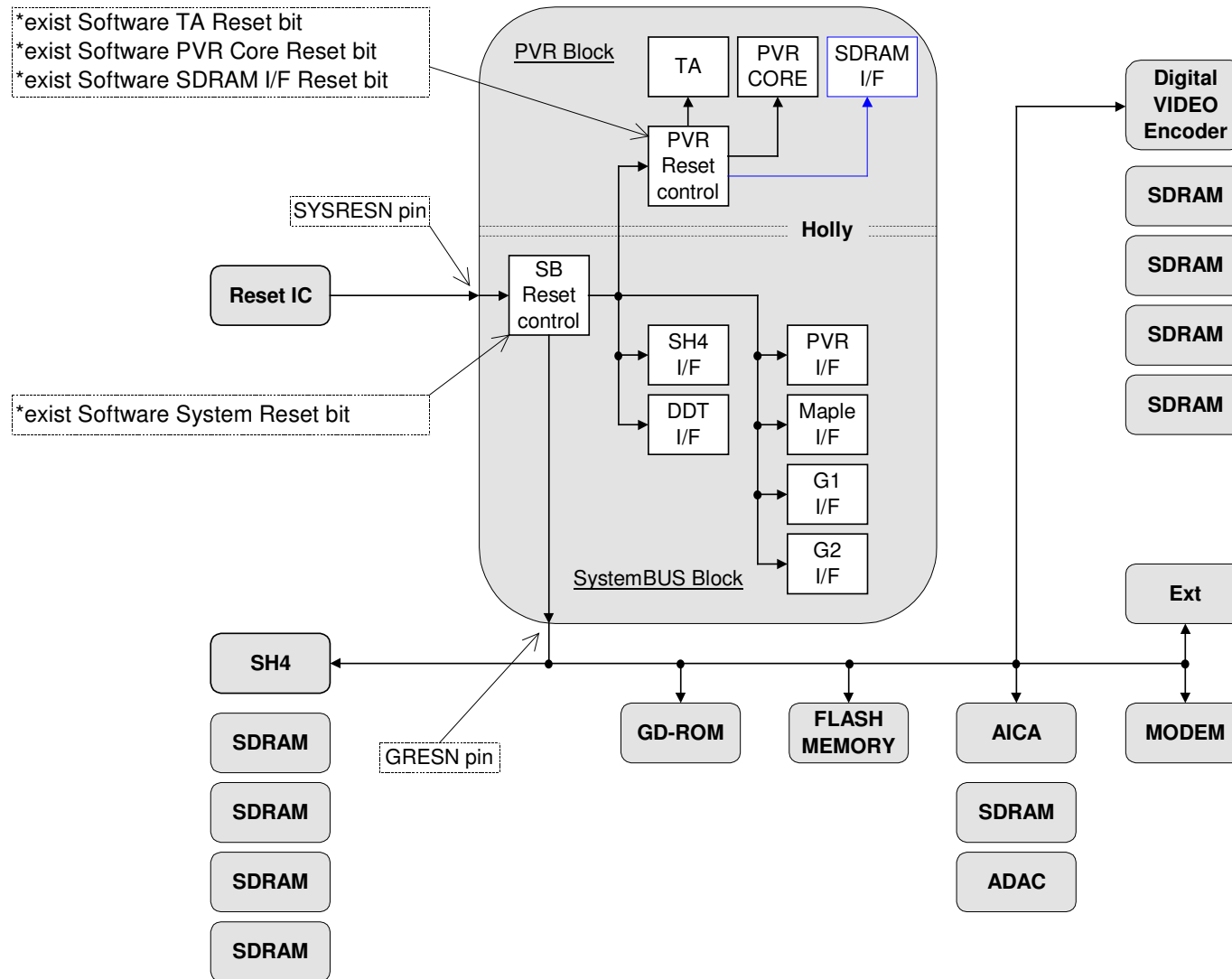


Fig. 8-11 Reset System Diagram

### **§ 8.1.3 Clock**

#### **§ 8.1.3.1 PLL**

#### **§ 8.1.3.2 Clock Tree**

Fig. 8-12 on the next page shows the clock tree for the Dreamcast System.



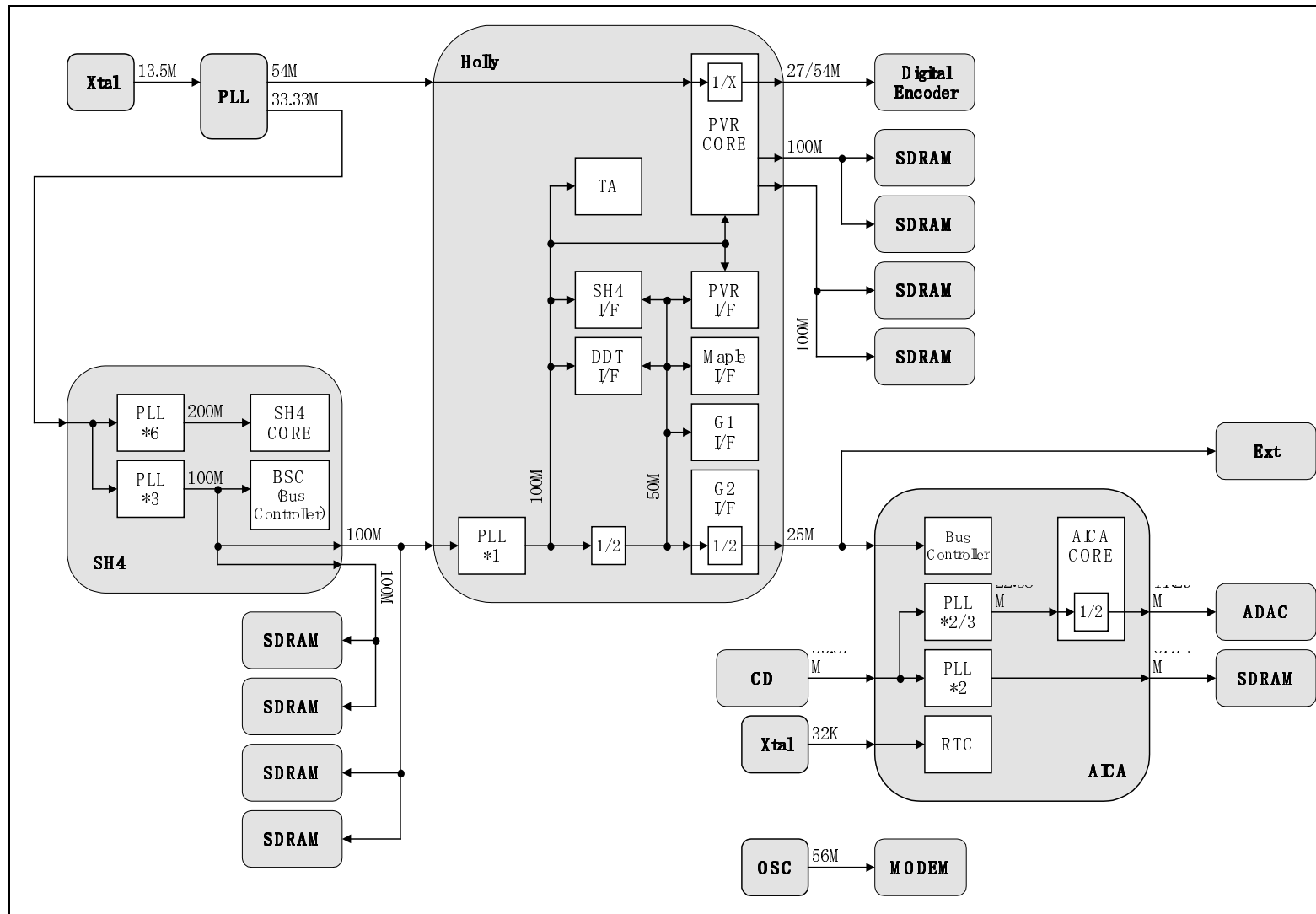


Fig. 8-12 System Clock Tree

#### **§ 8.1.4 JTAG Interface**

##### **§ 8.1.4.1 SH4**

##### **§ 8.1.4.2 HOLLY**

##### **§ 8.1.4.3 AICA**

### **§ 8.2 Individual Block Diagrams**

#### **§ 8.2.1 Detailed Block Diagram of Entire System**

#### **§ 8.2.2 CPU Subsystem (Including System Memory)**

#### **§ 8.2.3 HOLLY Subsystem**

#### **§ 8.2.4 GD-ROM Subsystem**

#### **§ 8.2.5 AICA Subsystem**

#### **§ 8.2.6 Digital Video Encoder Subsystem**

#### **§ 8.2.7 16Mbit SDRAM (16bit)**

#### **§ 8.2.8 64Mbit SGRAM (32bit)**

#### **§ 8.2.9 Power Supply**

### **§ 8.3 Pin Assignments (with Descriptions of Pins) Pin Assignments for Each Chip**

#### **§ 8.3.1 CPU**

#### **§ 8.3.2 HOLLY**

#### **§ 8.3.3 GD-ROM**

#### **§ 8.3.4 AICA**

#### **§ 8.3.5 Digital Video Encoder**

#### **§ 8.3.6 16Mbit SDRAM (16bit)**

#### **§ 8.3.7 64Mbit SGRAM (32bit)**

## § 8.4 List of Registers

A list of the registers in this system is shown below. "[R]" in the tables is an abbreviation for "Reserved." For unused bits in each register, specify "0" when writing to the register. When reading the register, an undefined value ("X") is returned.

Note that the register addresses that are given are the SH4's external (physical) memory addresses; in actual accesses, these addresses change according to the cache.

### § 8.4.1 System Bus Register

The System Bus-related registers are divided into groups as shown below.

Writing to registers not shown in this list is prohibited. If such a register is read, an undefined value "X" is returned.

#### **(System Registers... DMA, DDT, Interrupts, System Controller)**

- ch2-DMA control registers
- Sort-DMA control registers
- DDT I/F block control registers
- System control registers
- Interrupt control registers
- DMA hard trigger control registers

#### **(Maple Peripheral Interface... GamePad etc. )**

- Maple-DMA control registers
- Maple I/F Block Control Registers
- Maple-DMA secret/debug register
- Maple I/F Block hardware control registers

#### **(G1 Interface... GD-ROM, System-ROM, Flash-ROM etc. )**

- GD-DMA control registers
- GD-DMA secret/debug register
- G1 I/F block hardware control registers

#### **(G2 Interface... AICA, External Devices, Development Tools etc.)**

- G2-DMA control registers (Including AICA-DMA,Ext-DMA1,Ext-DMA2,Dev-DMA)
- G2-DMA secret register
- G2-DMA debug registers (Including AICA-DMA,Ext-DMA1,Ext-DMA2,Dev-DMA)
- G2 I/F block hardware control registers

#### **(PowerVR Interface... PowerVR Core)**

- PVR-DMA control registers
- PVR-DMA secret/debug register
- PVR I/F block hardware test register

### § 8.4.1.1 System Registers

***(The ch2-DMA Control Registers are described below.)***

#### **SB\_C2DSTAT**

Address : 0x005F 6800

bit 31-26	25-5	4-0
000100	ch2-DMA Texture Memory start address	Reserved

This register specifies the ch2-DMA destination address for transfers to the TA FIFO buffer.

#### **<Addresses that can be specified>**

0x10000000 ~ 0x107FFFFE0 : TA FIFO - Polygon Path (8MB)  
 0x10800000 ~ 0x10FFFFFFE0 : TA FIFO - YUV Converter Path (8MB)  
 0x11000000 ~ 0x11FFFFFFE0 : TA FIFO - Direct Texture Path (16MB)  
 (When Direct Texture Path is specified, the value in this register is incremented automatically while DMA is being executed.)

The following are the mages for the above areas:

0x12000000 ~ 0x127FFFFE0 : TA FIFO - Polygon Path (8MB)  
 0x12800000 ~ 0x12FFFFFFE0 : TA FIFO - YUV Converter Path (8MB)  
 0x13000000 ~ 0x13FFFFFFE0 : TA FIFO - Direct Texture Path (16MB)  
 (When Direct Texture Path is specified, the value in this register is incremented automatically while DMA is being executed.)

#### Notes:

- This register is not initialized after a power-on reset or a software reset.
- If 0x0000 0000 is specified for an address, 0x1000 0000 is accessed.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- When Direct Texture Path is specified, the value in this register is incremented in accordance with DMA execution.
- When transferring data to the texture memory via the TA FIFO buffer and Direct Texture Path, either 64-bit access or 32-bit access can be specified by setting the *SB\_LMMODE0* and *1* registers.
- When using the Polygon Path and the YUV Converter path for the TA FIFO buffer, the value specified in this register is maintained.

#### **SB\_C2DLEN**

Address : 0x005F 6804

bit 31-24	23-5	4-0
Reserved	ch2-DMA transfer length	Reserved

This register specifies the ch2-DMA length for transfers to TA FIFO.

Setting (32 bits)	Length
0x0000 0020	32 bytes
0x0000 0040	64 bytes
.....	.....
0x00FF FFE0	16M bytes – 32 bytes
0x0000 0000	16M bytes (default)

#### Notes:

- This register is not initialized after a power-on reset or a software reset.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- The value in this register is decremented during DMA execution.

#### **SB\_C2DST**

Address : 0x005F 6808

bit 31-1	0
Reserved	start/status

This register initiates ch2-DMA. When read, this register returns the ch2-DMA transfer status.

ch2-DMA can be initiated by writing this register. The ch2-DMA status can be checked by reading this register. If DMA terminates, this bit is automatically cleared to "0".

When writing	
Setting	Meaning
0	ch2-DMA stop (default)
1	ch2-DMA start

When reading	
Setting	Meaning
0	ch2-DMA not in progress. (default)
1	ch2-DMA in progress.

**(The Sort-DMA Control Registers are described below.)**

**SB\_SDSTAW**

Address : 0x005F 6810

bit 31-27	26-5	4-0
00001	Sort-DMA Start Link Address Table Start Address	Reserved

This register specifies the start address of the Sort-DMA start link address table in system memory.

**Notes:**

- This register is not initialized after a power-on reset or a software reset.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- The value in this register is incremented automatically during DMA execution.
- Note that if 0x0000 0000 is specified in this register, it is handled as 0x0800 0000.

**SB\_SDBAAW**

Address : 0x005F 6814

bit 31-27	26-5	4-0
00001	Sort-DMA Link Base Address	Reserved

This register specifies the Sort-DMA link base address in system memory.

**Notes:**

- This register is not initialized after a power-on reset or a software reset.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- The hardware does not change the data in this register.
- Note that if 0x0000 0000 is specified in this register, it is handled as 0x0800 0000.

**SB\_SDWLT**

Address : 0x005F 6818

bit 31-1	0
Reserved	number of bits

This register specifies the bit width of the link address that is stored in the Sort-DMA start link address table. A setting of "0" indicates a width of 16 bits; a setting of "1" indicates a width of 32 bits.

**Notes:**

- This register is not initialized after a power-on reset or a software reset.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- The hardware does not change the data in this register.

### SB\_SDLAS

Address : 0x005F 681C

bit 31-1	0
Reserved	shift control

This register controls shifting of the Sort-DMA link address. This register is not initialized after a reset.

Setting	Meaning
0	Referenced address = $SB\_SDBAAW + \text{Link address}$
1	Referenced address = $SB\_SDBAAW + \text{Link address} * 32$ (The CPU must specify a value that is the link address divided by 32.)

Notes:

- This register is not initialized after a power-on reset or a software reset.
- Because the hardware uses this register directly, overwriting this register while DMA is being executed is prohibited. (The register may be read.)
- The hardware does not change the data in this register.

### SB\_SDST

Address : 0x005F 6820

bit 31-1	0
Reserved	start /status

This register controls the start of Sort-DMA. When read, this register returns the DMA transfer status.

When writing	
Setting	Meaning
0	Sort-DMA stop (default)
1	Sort-DMA start

When reading	
Setting	Meaning
0	Sort-DMA not in progress. (default)
1	Sort-DMA in progress.

Notes:

- If Sort-DMA is interrupted, it is not possible to resume from where the transfer was halted; instead, it is necessary to start over, beginning with the setting of the Sort-DMA registers.

**(The DDT I/F Block Control & System Control Register is described below.)**

**SB\_DBREQM**

Address : 0x005F 6840

bit 31-1	0
Reserved	mask control

This register controls the masking of the output of the **DBREQ#** signal to the SH4.

Setting	Meaning
0	Not masked. (default)
1	Masked

- When the signal is masked, a fairly long wait occurs in DMA transfers that use cho-DDT. This wait could trigger a Maple-DMA timeout error.

**SB\_BAVLWC**

Address : 0x005F 6844

bit 31-5	4-0
Reserved	BAVL# wait count value

This register specifies the cho-DDT priority, and specifies the maximum wait for the **BAVL#** signal to be asserted by the SH4 in units of clock pulses. If the **BAVL#** signal is not asserted even though the specified number of clock pulses have elapsed, the DDT controller outputs the **DBREQ#** signal to request that the **BAVL#** signal be asserted.

Reducing this setting speeds up cho-DDT, but has an adverse effect on the efficiency of ch2-DMA and SH4 external accesses.

Setting values	Meaning
0x01	1 clock: cho-DDT wait time is short.
0x02	2clock
:	:
0x1F	31clock
0x00	32 clocks: cho-DDT wait time is long. (default)

**Notes**

- The above default value assigns the lowest priority to DDT. In other words, SH4 or polygon/texture data transfers have higher priority.



### SB\_C2DPRYC

Address : 0x005F 6848

bit 31-4	3-0
Reserved	DMA(TA/RootBus ) priority count

This specifies the number of times transfer requests from ch2-DMA or Sort-DMA to the Tile Accelerator should be accepted with priority ahead of transfer requests from the Root Bus (DDT interface).

Setting	Meaning
0x1	TA: Root Bus = 1:1 ...Waiting time for ch2-DMA or Sort-DMA is long.
0x2	TA: RootBus=2:1
:	:
0xF	TA: RootBus=15:1
0x0	TA:RootBus=16:1 ...Waiting time for ch2-DMA or Sort-DMA is short. (default)

#### Notes:

- The above default value assigns the lowest priority to DDT from the Root Bus (except for Sort-DMA). In other words, SH4 or polygon/texture data transfers have higher priority.

### SB\_C2DMAXL

Address : 0x005F 684C

bit 31-2	1-0
Reserved	ch2-DMA Maximum burst length

This register specifies the maximum burst length for ch2-DMA. (The maximum burst length setting is needed in order to prevent ch2-DMA from continually occupying the bus, causing long waits for cho-DDT and SH4 external accesses.)

Setting	Meaning
1	128 bytes: CPU wait time is short. (default)
2	256Byte
3	(Setting prohibited)
0	1024 bytes: CPU wait time is long.

#### Notes

- Never write to this register while ch2-DMA is in progress (when *SB\_C2DST* is 0x00000001).
- Because the DDT controller always samples the free space in the TA FIFO while conducting a transfer, a burst can be ended because the FIFO is full, even if the set value has not been reached. The benefits of the setting manifest themselves the most in direct texture transfer.
- When there is free space in the TA FIFO, ch2-DMA is performed using this maximum value for its burst length. For example, if ch2-DMA is being used to transfer 1024 bytes of texture data into texture memory, and the length set by this register is "1," the data is transferred 128 bytes at a time.
- If there is no free space in the TA FIFO, the DMA waits until free space develops. (DMA is interrupted.)
- The default value indicated above gives the CPU the highest priority. If the setting is increased, ch2-DMA becomes faster, but cho-DDT and SH4 external access become slower.
- The range of settings for this register is from 0x0 to 0x2; do not set 0x3.

### SB\_TFREM (Read Only)

Address : 0x005F 6880

bit 31-3	3-0
Reserved	TA FIFO remain

This register returns the remaining free space in the Tile Accelerator's FIFO buffer, in units of 32 bytes.

Setting	Remaining free space
0x0	0 byte
0x1	32 bytes
.....	.....
0x7	224 bytes
0x8	256 bytes

Note:

- This register is not initialized after a power-on reset or a software reset.

### SB\_LMMODE0

Address : 0x005F 6884

bit 31-1	0
Reserved	bus select-0

This register determines the data size when writing to the area from 0x1100 0000 to 0x11FF FFFF in texture memory via the TA FIFO buffer - Direct Texture Path.

Setting	Meaning
0	64 bit (default)
1	32 bit

### SB\_LMMODE1

Address : 0x005F 6888

bit 31-1	0
Reserved	bus select-1

This register determines the data size when writing to the area from 0x1300 0000 to 0x13FF FFFF in texture memory via the TA FIFO buffer. The meanings of the settings are the same as for SB\_LMMODE0. (The default is also the same.)

Note:

- The area from 0x1300 0000 to 0x13FF FFFF is the image area for 0x1100 0000 to 0x11FF FFFF in texture memory.

---

**SB\_FFST (Read Only)**

Address : 0x005F 688C

bit 31-6	5-0
Reserved	FIFO Status

This register indicates the FIFO status. If one of the bits shown below is read and returns a "0," the corresponding FIFO is empty; if the bit returns a "1," the corresponding FIFO is not empty.

- bit 5 = SH4 i/f FIFO
- bit 4 = G2 i/f FIFO (CPU write FIFO)
- bit 3 = Ext dev. transfer request input
- bit 2 = Ext2 transfer request input
- bit 1 = Ext1 transfer request input
- bit 0 = AICA transfer request input (connected to the AICA chip's FIFO empty pin)

**Note**

- This register is not initialized after a power-on reset or a software reset.

**SB\_SFRES (Write Only)**

Address : 0x005F 6890

bit 31-16	15-0
Reserved	Software Reset (Code : 0x7611)

This register is used to reset the entire system. A system software reset is initiated by writing "0x00007611" to this register.

**Notes:**

- If a value other than 0x0000 7611 is written to this register, it is ignored.
- Because this register resets the entire system, including external devices, it should be used with caution.

**SB\_SBREV (Read Only)**

Address : 0x005F 689C

bit 31-8	7-0
Reserved	SB Revision Number

This register indicates the revision number of the system bus block. For details on the register value, refer to section 1.4.

**SB\_RBSPLT**

Address : 0x005F 68A0

bit 31	30-0
SH4 RootBus Split enable	Reserved

***SH4 Root Bus Split enable***

- "0" : Single Write Burst (Default)
- "1" : Single Write Split

**(The Interrupt Control Registers are described below.)**

**SB\_ISTNRM**

Address : 0x005F 6900

bit 31	30	29-22	21-0
Error status	G1,G2,Ext status	Reserved	Normal interrupt clear/status

This register returns the interrupt status of the CORE, the System Bus, G1, G2 devices, etc. If a "1" is read in any bit, that indicates that the corresponding interrupt is being generated. In addition, an interrupt can be cleared by writing "1" to the corresponding bit, from **bit 21 to bit 0**. Bits 31 and 30 are read-only bits; those interrupts cannot be cleared by writing a "1" to those bits. ~~This depends on the HOLLY version; in HOLLY2, bit 21 is added.~~

**bit 31 = Error interrupt 'OR' status**

This bit is set to "1" when any of the following error interrupts are being generated (default = 0): RENDER ISP out of cache, RENDER Rendering aborted by FRAME change, etc. (default = 0; refer to the *SB\_ISTEXT* register)

**bit 30 = G1,G2,External interrupt 'OR' status**

This bit is set to "1" when any external interrupt (for the CD-ROM, AICA, modem, or external device) is being generated. (default = 0)  
- Error interrupt: (Bit0)ISP out of Cache, (bit1) Hazard Processing of Strip Buffer (refer to the *SB\_ISTERR* register)

**Normal interrupt clear/status**

If a "1" is read in any of the following bits, that indicates that the corresponding normal interrupt is being generated. (default = 0x000000) In addition, these interrupts can be cleared by writing "1" to the corresponding bit.

**bit 21 = End of Transferring interrupt : Punch Through List** ~~(\*only for HOLLY2)~~

bit 20 = End of DMA interrupt : Sort-DMA (Transferring for alpha sorting)

bit 19 = End of DMA interrupt : ch2-DMA

bit 18 = End of DMA interrupt : Dev-DMA(Development tool DMA)

bit 17 = End of DMA interrupt : Ext-DMA2(External 2)

bit 16 = End of DMA interrupt : Ext-DMA1(External 1)

bit 15 = End of DMA interrupt : AICA-DMA

bit 14 = End of DMA interrupt : GD-DMA

bit 13 = Maple V blank over interrupt

bit 12 = End of DMA interrupt : Maple-DMA

bit 11 = End of DMA interrupt : PVR-DMA

bit 10 = End of Transferring interrupt : Translucent Modifier Volume List

bit 9 = End of Transferring interrupt : Translucent List

bit 8 = End of Transferring interrupt : Opaque Modifier Volume List

bit 7 = End of Transferring interrupt : Opaque List

bit 6 = End of Transferring interrupt : YUV

bit 5 = H Blank-in interrupt

bit 4 = V Blank-out interrupt

bit 3 = V Blank-in interrupt

bit 2 = End of Render interrupt : TSP

bit 1 = End of Render interrupt : ISP

bit 0 = End of Render interrupt : Video

(For details on each interrupt, refer to section 8.5, "List of Interrupts.")

**SB\_ISTEXT (Read Only)**

Address : 0x005F 6904

bit 31-4	3-0
Reserved	Ext. interrupt clear/status

This register returns the status of the external interrupts. If a "1" is read in any of the following bits, that indicates that the corresponding interrupt is being generated. The bit status after a reset is determined by signals from external devices.

bit 3 = External Device interrupt

bit 2 = Modem interrupt

bit 1 = AICA interrupt

bit 0 = GD-ROM interrupt

(For details on each interrupt, refer to section 8.5, "List of Interrupts.")

**Note:**

- This register is a read-only register; these interrupts cannot be cleared by writing this register. In order to clear any of these bits, it is necessary to directly clear the interrupt in the device that is the source of the interrupt.

**SB\_ISTERR**

Address : 0x005F 6908

bit 31-0
Error Interrupt clear/status

This register returns the interrupt status of the CORE, the System Bus, G1, G2 devices, etc. If a "1" is read in any bit, that indicates that the corresponding interrupt is being generated. In addition, an interrupt can be cleared by writing "1" to the corresponding bit. (default = 0x00000000).

bit 31 = SH4 i/f : accessing to Inhibited area

bit 30 = Reserved bit 29 = Reserved

bit 28 = DDT i/f : Sort-DMA

**Command Error**

bit 27 = G2 : Time out in CPU accessing

bit 26 = G2 : Dev-DMA Time out

bit 25 = G2 : Ext-DMA2 Time out

bit 24 = G2 : Ext-DMA1 Time out

bit 23 = G2 : AICA-DMA Time out

bit 22 = G2 : Dev-DMA over run

bit 21 = G2 : Ext-DMA2 over run

bit 20 = G2 : Ext-DMA1 over run

bit 19 = G2 : AICA-DMA over run

bit 18 = G2 : Dev-DMA Illegal Address set

bit 17 = G2 : Ext-DMA2 Illegal Address set

bit 16 = G2 : Ext-DMA1 Illegal Address set

bit 15 = G2 : AICA-DMA Illegal Address set

bit 14 = G1 : ROM/FLASH access at GD-DMA

bit 13 = G1 : GD-DMA over run

bit 12 = G1 : Illegal Address set

bit 11 = MAPLE : Illegal command

bit 10 = MAPLE : Write FIFO over flow

bit 9 = MAPLE : DMA over run

bit 8 = MAPLE : Illegal Address set

bit 7 = PVRIF : DMA over run

bit 6 = PVRIF : Illegal Address set

bit 5 = TA : FIFO Overflow

bit 4 = TA : Illegal Parameter

bit 3 = TA : Object List Pointer Overflow

bit 2 = TA : ISP/TSP Parameter Overflow

bit 1 = RENDER : Hazard Processing of Strip Buffer

bit 0 = RENDER : ISP out of Cache(Buffer over flow)

(For details on each interrupt, refer to section 8.5, "List of Interrupts.")

**SB\_IML2NRM**

Address : 0x005F 6910

**SB\_IML4NRM**

Address : 0x005F 6920

**SB\_IML6NRM**

Address : 0x005F 6930

bit31-22	21-0
Reserved	Level (2/4/6) normal interrupt mask control

These are the mask control registers for normal interrupts. Each interrupt can be masked in each priority level (2/4, or 6). In addition, in all three priority levels, the arrangement of the bits in the register is the same as that of bits 21 through 0 in the *SB\_ISTNRM* register. (default = 0x000000) When a bit is set to "0," the corresponding interrupt is masked. When a bit is set to "1," that interrupt is enabled. ~~This depends on the HOLLY version; in HOLLY2, bit 21 is added.~~

bit 21 = End of Transferring interrupt : Punch Through List ~~(\*only for HOLLY2)~~

bit 20 = End of DMA interrupt : Sort-DMA (Transferring for alpha sorting)

bit 19 = End of DMA interrupt : ch2-DMA

bit 18 = End of DMA interrupt : Dev-DMA

bit 17 = End of DMA interrupt : Ext-DMA2

bit 16 = End of DMA interrupt : Ext-DMA1

bit 15 = End of DMA interrupt : AICA-DMA

bit 14 = End of DMA interrupt : GD-DMA

bit 13 = Maple V blank over interrupt

bit 12 = End of DMA interrupt : Maple-DMA

bit 11 = End of DMA interrupt : PVR-DMA

bit 10 = End of Transferring interrupt : Translucent Modifier Volume List

bit 9 = End of Transferring interrupt : Translucent List

bit 8 = End of Transferring interrupt : Opaque Modifier Volume List

bit 7 = End of Transferring interrupt : Opaque List

bit 6 = End of Transferring interrupt : YUV

bit 5 = H-Blank in interrupt

bit 4 = V-Blank out interrupt

bit 3 = V-Blank in interrupt

bit 2 = End of Render interrupt : TSP

bit 1 = End of Render interrupt : ISP

bit 0 = End of Render interrupt : Video

Note:

- After a power-on reset or a software reset, all interrupts are disabled.

**SB\_IML2EXT**

Address : 0x005F 6914

**SB\_IML4EXT**

Address : 0x005F 6924

**SB\_IML6EXT**

Address : 0x005F 6934

bit 31-4	3-0
Reserved	level (2/4/6) Ext. interrupt mask control

These registers control masking of the external interrupts at each priority level. When a bit is set to "0," the corresponding interrupt is masked. When a bit is set to "1," that interrupt is enabled. (default = 0x0) The arrangement of the bits in the register is the same as that in *SB\_ISTEXT*.

bit 3 = External Device interrupt

bit 2 = Modem interrupt

bit 1 = AICA interrupt

bit 0 = GD-ROM interrupt

Note:

- After a power-on reset or a software reset, all interrupts are disabled.

**SB\_IML2ERR**

Address : 0x005F 6918

**SB\_IML4ERR**

Address : 0x005F 6928

**SB\_IML6ERR**

Address : 0x005F 6938

bit 31-0
Level (2/4/6) Error interrupt mask control

These registers control masking of the error interrupts at each priority level. When a bit is set to "0," the corresponding interrupt is masked. When a bit is set to "1," that interrupt is enabled.

(default = 0x0) The arrangement of the bits in the register is the same as that of bits 31 through 0 in *SB\_ISTERR*.

bit 31 = SH4 i/f : accessing to Inhibited area		
bit 30 = Reserved	bit 29 = Reserved	bit 28 = DDT i/f : Sort-DMA

Command Error

- bit 27 = G2 : Time out in CPU accessing
- bit 26 = G2 : Dev-DMA Time out
- bit 25 = G2 : Ext-DMA2 Time out
- bit 24 = G2 : Ext-DMA1 Time out
- bit 23 = G2 : AICA-DMA Time out
- bit 22 = G2 : Dev-DMA over run
- bit 21 = G2 : Ext-DMA2 over run
- bit 20 = G2 : Ext-DMA1 over run
- bit 19 = G2 : AICA-DMA over run
- bit 18 = G2 : Dev-DMA Illegal Address set
- bit 17 = G2 : Ext-DMA2 Illegal Address set
- bit 16 = G2 : Ext-DMA1 Illegal Address set
- bit 15 = G2 : AICA-DMA Illegal Address set
- bit 14 = G1 : ROM/FLASH access at GD-DMA
- bit 13 = G1 : GD-DMA over run
- bit 12 = G1 : Illegal Address set
- bit 11 = MAPLE : Illegal command
- bit 10 = MAPLE : Write FIFO over flow
- bit 9 = MAPLE : DMA over run
- bit 8 = MAPLE : Illegal Address set
- bit 7 = PVRIF : DMA over run
- bit 6 = PVRIF : Illegal Address set
- bit 5 = TA : FIFO Overflow
- bit 4 = TA : Illegal Parameter
- bit 3 = TA : Object List Pointer Overflow
- bit 2 = TA : ISP/TSP Parameter Overflow
- bit 1 = RENDER : Hazard Processing of Strip Buffer
- bit 0 = RENDER : ISP out of Cache(Buffer over flow)

(For details on each interrupt, refer to section 8.5, "List of Interrupts.")

Note:

- After a power-on reset or a software reset, all interrupts are disabled.

***(The DMA Hard Trigger Control Registers are described below.)***

**SB\_PDTNRM**

Address : 0x005F 6940

bit31-22	21-0
Reserved	PVR-DMA Trigger mask – Normal interrupt

This register indicates the PVR DMA trigger mask for normal interrupts. (default = 0x000000) For details on the bit arrangement, refer to the description of the *SB\_IML2(/4/6)NRM* register.

~~\*In HOLLYe, bit 21 is added.~~

Setting	Meaning
0	interrupt mask
1	interrupt enable

**SB\_PDTEXT**

Address : 0x005F 6944

bit 31-4	3-0
Reserved	PVR-DMA Trigger mask - External interrupt

This register indicates the PVR DMA trigger mask for external interrupts. (default = 0x0) For details on the bit arrangement, refer to the description of the *SB\_IML2(/4/6)EXT* register.

Setting	Meaning
0	interrupt mask
1	interrupt enable

**SB\_G2DTNRM**

Address : 0x005F 6950

bit31-22	21-0
Reserved	G2-DMA Trigger mask – Normal interrupt

This register indicates the G2-DMA trigger mask for normal interrupts. (default = 0x000000) For details on the bit arrangement, refer to the description of the *SB\_IML2(/4/6)NRM* register.

~~\*In HOLLYe, bit 21 is added.~~

Setting	Meaning
0	interrupt mask
1	interrupt enable

**SB\_G2DTEXT**

Address : 0x005F 6954

bit 31-4	3-0
Reserved	G2-DMA Trigger mask - External interrupt

This register indicates the G2-DMA trigger mask for external interrupts. (default = 0x0) For details on the bit arrangement, refer to the description of the *SB\_IML2(/4/6)EXT* register.

Setting	Meaning
0	interrupt mask
1	interrupt enable

**§ 8.4.1.2 Maple Peripheral Interface**

***(The Maple-DMA Control Registers are described below.)***

**SB\_MDSTAR**

Address : 0x005F 6C04



bit 31-29	28-5	4-0
000	Maple-DMA command table address	Reserved

This register specifies the address of the peripheral controller command table in system memory.

Notes

- This register is not initialized after a power-on reset or a software reset.
- The hardware does not change the data in this register.

**SB\_MDTSEL**

Address : 0x005F 6C10

bit 31-1	0
Reserved	Maple-DMA Trigger select

Selects the initiation source (software, V-Blank) for Maple-DMA (transmission/reception with a peripheral).

Setting	Initiation trigger	Meaning
0	Software initiation (default)	Maple-DMA is initiated by an access from the SH4. Initiation is possible by writing a "1" to the <i>SB_MDST</i> register.
1	V-Blank initiation	Maple-DMA is initiated automatically one line before the start of the screen display (V-Blank Out).

**SB\_MDEN**

Address : 0x005F 6C14

bit 31-1	0
Reserved	Maple-DMA enable

This is the Maple-DMA (transmission/reception with peripherals) enable register. (default = 0)  
o) When this bit is set to "1", DMA can be initiated by setting bit 0 (DMA start bit) of the *SB\_MDST* register to "1".

When writing	
Setting	Meaning
0	Abort Maple DMA
1	Enable

When reading	
Setting	Meaning
0	Disable
1	Enable

Notes:

- Transmission/reception is not performed if this bit is not set to "1".
- DMA is forcibly terminated if a "0" is written to this bit while a Maple-DMA transfer is in progress.
- The hardware does not change the data in this register.

---

**SB\_MDST**

Address : 0x005F 6C18

bit 31-1	0
Reserved	Maple-DMA start/status

This register starts the transmission/reception software. (default = 0)  
When read, this register shows a status bit indicating the transmission/reception status.

When writing	
Setting	Meaning
0	ignored
1	Maple DMA start

When reading	
Setting	Meaning
0	Maple-DMA not in progress.
1	Maple-DMA in progress.

**Notes:**

- Writing to this register is valid only when the Maple-DMA initiation setting in the *SB\_MDTSEL* register is for software initiation.
- A "1" must not be written to this register while Maple-DMA is prohibited in the *SB\_MDEN* register (bit 31 = 0).

***(The Maple Interface Block Control Registers are described below.)***

**SB\_MSYS**

Address : 0x005F 6C80

bit 31-16	15-13	12	11-10	9-8	7-4	3-0
Time Out Counter	R	Single Hard Trigger	R	Sending Rate	R	Delay Time

***Time Out Counter***

This field sets the timeout duration from the start of data output to a peripheral device. (default = 0x3A98) A value of "1" is equivalent to 20nsec.

Example

$20\text{nsec} \times 0x3A98 = 300 / 1000000 \text{ sec}$  (1 screen is 16.7msec.)

$20\text{nsec} \times 0xC350 = 1\text{msec}$

***Single Hard Trigger***

This bit is set by selecting either to re-initiate automatically in response to V-Blank when Maple-DMA has been initiated by V-Blank, or to stop V-Blank initiation (manual) until the *SB\_MSHTCL* register has been cleared.

Setting	Description
0	Automatic (default)
1	Manual

***Sending Rate (between HOLLY and Peripherals)***

This field sets the data transfer rate for transfers between the system and peripherals.

Setting	Meaning
00	2M bps (default)
01	1M bps

***Delay Time***

These bits set the interval (delay time) until Maple-DMA is initiated after V-Blank Out when V-Blank has been selected as the initiation trigger. (default = 0x0)

"1" sets an interval of 1.3msec.

Example

$1.3\text{msec} \times 4 = 5.2\text{msec}$  (One screen requires 16.7msec.)

Note:

- The Single Hard Trigger and Delay Time settings are valid only when V-Blank initiation is set for Maple-DMA in the *SB\_MDTSEL* register.
- Setting a value of 11 or higher for the Delay Time setting is prohibited.
- The hardware does not change the data in this register.

### SB\_MST (Read Only)

Address : 0x005F 6C84

bit 31	30-27	26-24	23-22	21-16	15-8	7-0
Move Status	R	Internal Frame Monitor	R	Internal State Monitor	Reserved	Line Monitor

This register indicates the Maple interface status.

#### **Move Status**

This bit indicates the operating status (sending/receiving) of the peripheral controller. (default = 0x0)

Setting	Meaning
0	Controller is not in operation. (Received data was finalized.)
1	Controller is in operation. (Received data is not yet finalized. Transmission data may not be overwritten.)

#### **Internal Frame Monitor**

This is the internal block frame counter monitor. (default = 0x0)

#### **Internal State Monitor**

This is the internal block state counter monitor. (default = 0x0)

#### **Line Monitor**

This is the input/output line monitor for each port. (default = 0xFF)  
The correspondence with each bit is shown below.

Bit	Line that is monitored
bit7	Port D SDCKA
bit6	Port D SDCKB
bit5	Port C SDCKA
bit4	Port C SDCKB
bit3	Port B SDCKA
bit2	Port B SDCKB
bit1	Port A SDCKA
bit0	Port A SDCKB

### SB\_MSHTCL (Write Only)

Address : 0x005F 6C88

bit 31-1	0
Reserved	Maple-DMA Hard Trigger Clear

Re-initiation is enabled when V-Blank initiation is selected for Maple-DMA.

Setting	Description
0	ignored
1	Hardware trigger clear (V-Blank re-initiation)

#### Notes:

- Writing to this register is valid only when V-Blank initiation is set for Maple-DMA in the *SB\_MDTSEL* register and V-Blank re-initiation is set to "manual" in the *SB\_MSYS* register.
- Writing a "0" to this register is invalid.

**(The Maple-DMA Secret Register is described below.)**

**SB\_MDA PRO (Write Only)**

Address : 0x005F 6C8C

bit 31-16	15	14-8	7	6-0
Security code : 0x6155	R	Top address	R	Bottom address

This register specifies the address range for Maple-DMA involving the system (work) memory.

**Security code : 0x6155**

When updating bits 14 through 8 and bits 6 through 0, it is necessary to add "0x6155." (default = 0x0000) If this value is not added, bits 14 through 8 and bits 6 through 0 will not be updated.

**Top address**

This field specifies the starting address of the address range where received data will be stored in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x7F)

**Bottom address**

This field specifies the ending address of the address range where received data will be stored in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x00)

Examples of settings for **the top address** and **the bottom address** are shown below. (These examples also apply to the \*\*APRO register.)

- 40-7F → 0x0C000000-0x0FFFFFFF
- 7F-7F → 0x0FF00000-0x0FFFFFFF
- 7F-00 → Specification prohibited

**Notes:**

- Maple-DMA involving system memory will be performed only within the above address range.
- If a DMA transfer is generated outside of this range, an overrun error results and the Maple-DMA overrun error interrupt is generated. (Refer to bit 9 of the *SB\_ISTERR* register.)

**(The Maple Interface Block Hardware Control Register is described below.)**

<b>SB_MMSEL</b>		Address : 0x005F 6CE8
bit31-1		0
Reserved		Maple MSB Selection

This register specifies the MSB of data that is sent/received by Maple.

**Maple MSB Selection(Default=1)**

- |     |           |
|-----|-----------|
| "0" | MSB bit7  |
| "1" | MSB bit31 |

**(The Maple-DMA Debug Registers are described below.)**

**SB\_MTXDAD (Read Only)**

Address : 0x005F 6CF4

bit31-29	28-5	4-0
Reserved	Maple TxD Address Counter	Reserved

**Maple TxD Address Counter**

This is the address of the data that is to be loaded into the Maple controller through Maple-DMA.

Note:

- This register is not initialized after a power-on reset or a software reset.

**SB\_MRXDAD (Read Only)**

Address : 0x005F 6CF8

bit31-29	28-5	4-0
Reserved	Maple RxD Address Counter	Reserved

**Maple RxD Address Counter**

This is the address of the data that is to be written by the Maple controller through Maple-DMA.

Note:

- This register is not initialized after a power-on reset or a software reset.

**SB\_MRXDBD (Read Only)**

Address : 0x005F 6CFC

bit31-29	28-5	4-0
Reserved	Maple RxD Base Address	Reserved

**Maple RxD Base Address**

Note:

- This register is not initialized after a power-on reset or a software reset.

### § 8.4.1.3 G1 Interface

***(The GD-DMA Control Registers are described below.)***

#### **SB\_GDSTAR**

Address : 0x005F 7404

bit 31-29	28-5	4-0
000	GD-DMA start address	Reserved

Data transfers between the GD-ROM and the following areas are possible using cho-DMA. This register specifies the starting address in 32-byte units. (default = 0xFFFFFFFF)

0x00700000~0x00707FFE0 :32KByte : G2 AICA -Register  
 0x00800000~0x009FFFFE0 :2MByte : G2 AICA -Wave Memory  
 0x01000000~0x01FFFFFE0 :16Mbyte : G2 External Devices #1  
 0x02700000~0x02FFFFFE0 :9MByte : G2 AICA (Image area)  
 0x03000000~0x03FFFFFE0 :16Mbyte : G2 External Devices #2  
 0x04000000~0x047FFFFE0 :8MByte : PowerVR Texture Memory 64bit access area  
 0x05000000~0x057FFFFE0 :8MByte : PowerVR Texture Memory 32bit access area  
 0x06000000~0x067FFFFE0 :8MByte : PowerVR Tex. Mem. 64bit access area (Image area)  
 0x07000000~0x077FFFFE0 :8MByte : PowerVR Tex. Mem. 32bit access area (Image area)  
 0x0C000000~0x0CFFFFFE0 :16Mbyte : System Memory  
 0x0E000000~0x0EFFFFFE0 :16Mbyte : System Memory (Image area)  
 0x14000000~0x17FFFFFE0 :64Mbyte : G2 External Devices #3

#### Notes:

- This register is not initialized after a power-on reset or a software reset.
- The hardware does not change the data in this register.
- For details on address mapping, refer to section 2.1, "System Mapping."

#### **SB\_GDLEN**

Address : 0x005F 7408

bit 31-25	24-0
Reserved	GD-DMA Transfer Length

This register specifies the length for cho-DMA to the GD-ROM.

In a DMA transfer involving the GD-ROM, a transfer of a length indicated below is made from the GD-ROM to the starting address specified by the *SB\_GDSTAR* register. However, the basic unit for data transfer is 32 bytes.

Setting (32 bits)	Length
0x00000001	1 Byte
0x00000020	32 Byte
.....	.....
0x01FFFFFF	32M Byte – 1 Byte
0x00000000	32M Byte

#### Notes:

- This register is not initialized after a power-on reset or a software reset.
- The hardware does not change the data in this register.

**SB\_GDDIR**

Address : 0x005F 740C

bit 31-1	0
Reserved	GD-DMA direction

This register specifies the direction of the cho-DMA transfer involving the GD-ROM. In either case, the opposite side of the DMA transfer is the area specified by the *SB\_GDSTAR* register.

Setting	Meaning
0	DMA transfer to GD-ROM (default)
1	DMA transfer from GD-ROM Except in special cases, this is the mode that is normally used.

Note:

- The hardware does not change the data in this register.

**SB\_GDEN**

Address : 0x005F 7414

bit 31-1	0
Reserved	GD-DMA Enable

This register enables cho-DMA transfer involving the GD-ROM. This register can also be used to forcibly terminate such a DMA transfer that is in progress, by writing a "0" to this register.

When writing	
Setting	Meaning
0	Abort GD-DMA (default)
1	Enable

When reading	
Setting	Meaning
0	Disable (default)
1	Enable

Notes:

- This bit must be set to "1" in order to initiate a cho-DMA transfer involving the GD-ROM.
- If this bit is enabled and a DMA start request is made in the *SB\_GDST* register, the DMA transfer starts as soon as the data has been loaded into the GD-ROM buffer.
- A DMA transfer can be forcibly terminated by setting this register to "Disable" from the "Enable" state.
- The hardware does not change the data in this register.

**SB\_GDST**

Address : 0x005F 7418

bit 31-1	0
Reserved	GD-DMA Start/Status

This register requests the start of a cho-DMA transfer involving the GD-ROM. The status of the DMA transfer can be determined by reading this register. If a "0" is written to this register, it is ignored. If GD-DMA is set to "Disable" in the *SB\_GDEN* register, writing a "1" to this register is illegal.

When writing	
Setting	Meaning
0	ignored (default)
1	Start DMA

When reading	
Setting	Meaning
0	GD-DMA not in progress. (default)
1	GD-DMA in progress.

**(The G1 Interface Block Hardware Control Registers are described below.)**

**SB\_G1RRC (Write Only)**

Address : 0x005F 7480



---

---

bit 31-13	12	11-8	7-4	3	2-0
Reserved	OE Pulse delay	CS Pulse width	Address setup	R	Address hold

This register controls the timing for read accesses to system ROM.

**OE Pulse delay**

This field sets the OE signal delay versus the rising edge of the ROM CS signal. (default = 1 = 2 cycles)

The OE signal becomes active after  $(\text{pulse delay} + 1) \times 1$  cycles.

**CS Pulse width**

This field specifies the pulse width of the ROM CS signal. (default = 0xF = 18 cycles)

The CS signal is active for  $(\text{pulse width} + 3) \times 1$  cycles.

**Address setup**

This field specifies the address setup time versus the falling edge of the ROM CS signal. (default = 0xF = 16 cycles)

The read address becomes valid within  $(\text{address setup} + 1) \times 1$  cycles after the falling edge of the CS signal.

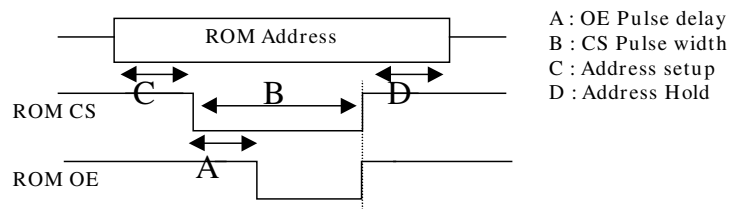
**Address hold**

This field specifies the address hold time versus the falling edge of the ROM CS signal. (default = 0x7 = 8 cycles)

The read address is valid for  $(\text{address hold} + 1) \times 1$  cycles after the falling edge of the CS signal.

Notes:

- 1 cycle = 20nsec.
- Based on the above settings, the timing for read accesses to ROM is as shown below.



The length of time needed for accesses to ROM is equal to  $B + C + D$ . Under the default settings, reading one byte of data requires 42 cycles (= 840nsec).

- The hardware does not change the data in this register.

---

**SB\_G1RWC (Write Only)**

Address : 0x005F 7484

bit 31-13	12	11-8	7-4	3	2-0
Reserved	WR Pulse delay	CS Pulse width	Address setup	R	Address hold

This register controls the timing for write accesses to system ROM.

**WR Pulse delay**

This field sets the WR signal delay versus the rising edge of the ROM CS signal.  
(default = 1 = 2 cycles)

The WR signal becomes active after  $(\text{pulse delay} + 1) \times 1$  cycles.

**CS Pulse width**

This field specifies the pulse width of the ROM CS signal. (default = 0xF = 18 cycles)

The CS signal is active for  $(\text{pulse width} + 3) \times 1$  cycles.

**Address setup**

This field specifies the address setup time versus the falling edge of the ROM CS signal.  
(default = 0xF = 16 cycles)

The write address becomes valid within  $(\text{address setup} + 1) \times 1$  cycles after the falling edge of the CS signal.

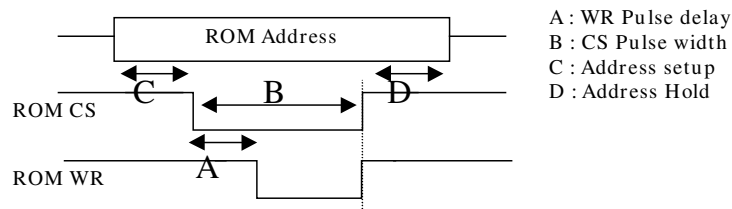
**Address hold**

This field specifies the address hold time versus the falling edge of the ROM CS signal.  
(default = 0x7 = 8 cycles)

The write address is valid for  $(\text{address hold} + 1) \times 1$  cycles after the falling edge of the CS signal.

Notes:

- 1 cycle = 20nsec.
- Based on the above settings, the timing for write accesses to ROM is as shown below.



The length of time needed for accesses to ROM is equal to  $B + C + D$ . Under the default settings, writing one byte of data requires 42 cycles (= 840nsec).

- The hardware does not change the data in this register.
- It is not possible to write to system ROM except in special cases, such as during Boot-ROM development work.

---

**SB\_G1FRC (Write Only)**

Address : 0x005F 7488

bit 31-13	12	11-8	7-4	3	2-0
Reserved	OE Pulse delay	CS Pulse width	Address setup	R	Address hold

This register adjusts the timing for read accesses to flash memory.

***OE Pulse delay***

default=1=2cyc

***CS Pulse width***

default=0xF=18cyc

***Address setup***

default=0xF=16cyc

***Address hold***

default=0x7=8cyc

For details on the function of each bit, refer to the description of the SB\_G1RRC register.

**SB\_G1FWC (Write Only)**

Address : 0x005F 748C

bit 31-13	12	11-8	7-4	3	2-0
Reserved	WR Pulse delay	CS Pulse width	Address setup	R	Address hold

This register adjusts the timing for write accesses to flash memory.

***WR Pulse delay***

default=1=2cyc

***CS Pulse width***

default=0xF=18cyc

***Address setup***

default=0xF=16cyc

***Address hold***

default=0x7=8cyc

For details on the function of each bit, refer to the description of the SB\_G1RWC register.

**SB\_G1CRC (Write Only)**

Address : 0x005F 7490

bit 31-13	11-8	7-4	3	2-0
Reserved	G1DIOR# Pulse width	Address setup	R	Address hold

This register adjusts the timing for GD PIO read accesses.

**G1DIOR# Pulse width**

This field specifies the pulse width of the GD PIO read signal (G1DIOR#).  
(default = 0xF = 18cycles)  
The signal is active for (pulse width + 3) × 1 cycles.

**Address setup**

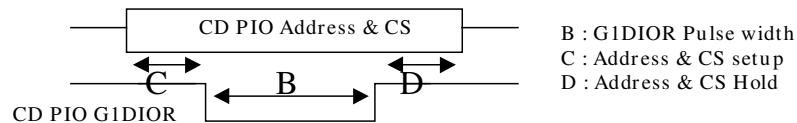
This field specifies the address setup time and the chip select time versus the falling edge of the **G1DIOR#** signal. (default = 0xF = 16 cycles)  
The **G1DIOR#** signal falls (address setup + 1) × 1 cycles after the address is output.

**Address hold**

This field specifies the address hold time and the chip select time versus the falling edge of the **G1DIOR#** signal. (default = 0x7 = 8 cycles)  
The read address is output until (address hold + 1) × 1 cycles after the falling edge of the **G1DIOR#** signal.

Notes:

- Based on the above settings, the timing for GD PIO read accesses is as shown below.



- The length of time needed for accesses to GD PIO is equal to B + C + D. Under the default settings, reading one word of data requires 42 cycles (= 840nsec).
- The hardware does not change the data in this register.

### SB\_G1CWC (Write Only)

Address : 0x005F 7494

bit 31-13	11-8	7-4	3	2-0
Reserved	G1DIOW# Pulse width	Address setup	R	Address hold

This register adjusts the timing for GD PIO write accesses.

#### ***G1DIOR# Pulse width***

This field specifies the pulse width of the GD PIO write signal (**G1DIOW#**). (default = 0xF = 18cycles)

The signal is active for (pulse width + 3) × 1 cycles.

#### ***Address setup***

This field specifies the address setup time and the chip select time versus the falling edge of the **G1DIOW#** signal. (default = 0xF = 16 cycles)

The **G1DIOW#** signal falls (address setup + 1) × 1 cycles after the address is output.

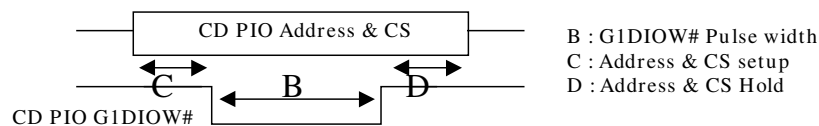
#### ***Address hold***

This field specifies the address hold time and the chip select time versus the falling edge of the **G1DIOW#** signal. (default = 0x7 = 8 cycles)

The write address is output until (address hold + 1) × 1 cycles after the falling edge of the **G1DIOW#** signal.

#### Notes:

- Based on the above settings, the timing for GD PIO write accesses is as shown below.



The length of time needed for accesses to GD PIO is equal to B + C + D. Under the default settings, writing one word of data requires 42 cycles (= 840nsec).

- The hardware does not change the data in this register.

### SB\_G1GDRC (Write Only)

Address : 0x005F 74A0

bit 31-16	15-12	11-8	7-4	3-0
Reserved	G1DIOR# Negate time width	Acknowledge delay time	G1DIOR# Pulse delay	G1DIOR# Pulse width

This register adjusts the timing for GD-DMA read accesses.

#### ***G1DIOR# Negate time width***

This field specifies the negate time for the GD read signal (**G1DIOR#**). (default = 0xF = 18 cycles)

The **G1DIOR#** signal becomes active after  $(\text{negate time width} + 3) \times 1$  cycles.

#### ***Acknowledge delay time***

This field specifies the negate delay for the GD acknowledge signal (**G1DACK#**) versus the falling edge of the **G1DIOR#** signal. (default = 0xF = 18 cycles)

**G1DACK#** is negated in  $(\text{acknowledge delay time} + 3) \times 1$  cycles.

#### ***G1DIOR# pulse delay***

This field specifies the **G1DIOR#** falling edge delay time versus the falling edge of the **G1DACK#** signal. (default = 0xF = 18 cycles) **G1DIOR#** is asserted  $(\text{G1DIOR# pulse delay} + 1) \times 1$  cycles after the falling edge of **G1DACK#**.

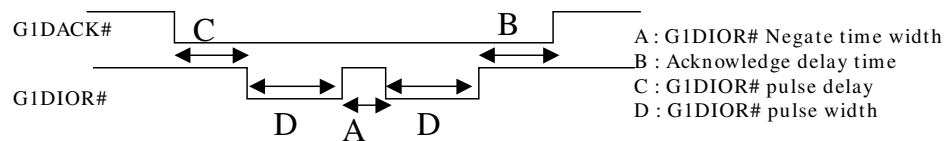
#### ***G1DIOR# pulse width***

This field specifies the pulse width of the **G1DIOR#** signal. (default = 0xF = 18 cycles)

**G1DIOR#** is asserted for  $(\text{G1DIOR# pulse width} + 1) \times 1$  cycles.

#### Note:

- Based on the above settings, the timing for GD-DMA read accesses is as shown below.



The length of time needed for GD-DMA read accesses is equal to  $(B + C) + (A + D) \times (\text{number of words}) - A$ . Under the default settings, reading one word of data requires 50 cycles (= 1000nsec).

- The hardware does not change the data in this register.

### SB\_G1GDWC (Write Only)

Address : 0x005F 74A4

bit 31-16	15-12	11-8	7-4	3-0
Reserved	G1DIOW# Negate time width	Acknowledge delay time	G1DIOW# Pulse delay	G1DIOW# Pulse width

This register adjusts the timing for GD-DMA write accesses.

#### **G1DIOW# Negate time width**

This field specifies the negate time for the GD write signal (**G1DIOW#**). (default = 0xF = 18 cycles)

The **G1DIOW#** signal becomes active after (negate time width + 3) × 1 cycles.

#### **Acknowledge delay time**

This field specifies the negate delay for the GD acknowledge signal (**G1DACK#**) versus the falling edge of the **G1DIOW#** signal. (default = 0xF = 18 cycles)

**G1DACK#** is negated in (acknowledge delay time + 3) × 1 cycles.

#### **G1DIOW# pulse delay**

This field specifies the G1DIOW# falling edge delay time versus the falling edge of the **G1DACK#** signal. (default = 0xF = 18 cycles) **G1DIOW#** is asserted (**G1DIOW#** pulse delay + 1) × 1 cycles after the falling edge of **G1DACK#**.

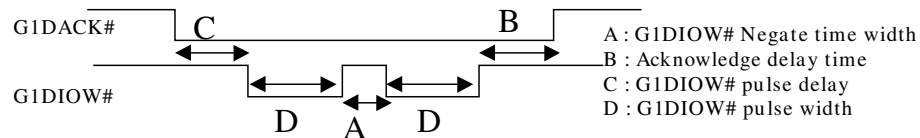
#### **G1DIOW# pulse width**

This field specifies the pulse width of the **G1DIOW#** signal. (default = 0xF = 18 cycles)

**G1DIOW#** is asserted for (**G1DIOW#** pulse width + 1) × 1 cycles.

#### Note:

- Based on the above settings, the timing for GD-DMA write accesses is as shown below.



The length of time needed for GD-DMA read accesses is equal to (B + C) + (A + D) × (number of words) - A. Under the default settings, reading one word of data requires 50 cycles (= 1000nsec).

- The hardware does not change the data in this register.

### SB\_G1SYSM (Read Only)

Address : 0x005F 74B0

bit 31-8	7-0
Reserved	System Mode

This register returns the system mode/configuration (by reading the contents of the address/mode pins (G1MRA[18:11] when HOLLY is reset). Refer to section 4.1.4, "System Modes," for the system mode correspondence table.

### SB\_G1CRDYC (Write Only)

Address : 0x005F 74B4

bit 31-1	0
Reserved	GD PIO RDY control

This register enables/disables the **G1IORDY** signal for GD PIO reads and writes.

Setting	Meaning
0	Disable
1	Enable (default)

**(The GD-DMA Secret Register is described below.)**

**SB\_GDAPRO (Write Only)**

Address : 0x005F 74B8

bit 31-16	15	14-8	7	6-0
Security code : 0x8843	R	Top address	R	Bottom address

This register specifies the address range for GD-DMA involving system (work) memory.

**Security code : 0x8843**

When updating bits 14 through 8 and bits 6 through 0, it is necessary to add "0x8843." (default = 0x0000) If this value is not added, bits 14 through 8 and bits 6 through 0 will not be updated.

**Top address**

This field specifies the starting address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x7F)

**Bottom address**

This field specifies the ending address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x00)

Note:

- GD-DMA involving system memory will be performed only within the above address range.
- If a DMA transfer is generated outside of this range, an overrun error results and the GD-DMA overrun error interrupt is generated. (Refer to bit 13 of the *SB\_ISTERR* register.)



**(The GD-DMA Debug Registers are described below.)**

**SB\_GDSTARD (Read Only)**

Address : 0x005F 74F4

bit 31-29	28-5	4-0
000	GD-DMA address count value	Reserved

This returns the current GD-DMA address, allowing you to determine the extent to which the GD-DMA address on the other size of a GD-DMA transfer has advanced.

This register counts up from the value that was set in the *SB\_GDSTAR* register, and stops at the last address of the DMA after the GD-DMA transfer is completed.

0x00700000~0x00707FFe0 :32KByte : G2 AICA -Register  
 0x00800000~0x009FFFFE0 :2MByte : G2 AICA -Wave Memory  
 0x01000000~0x01FFFFFE0 :16Mbyte G2 External Devices #1  
 0x02700000~0x027FFFFE0 :9MByte : G2 AICA (Image area)  
 0x03000000~0x03FFFFFE0 :16Mbyte G2 External Devices #2  
 0x04000000~0x047FFFFE0 :8MByte : PowerVR Texture Memory 64bit access area  
 0x05000000~0x057FFFFE0 :8MByte : PowerVR Texture Memory 32bit access area  
 0x06000000~0x067FFFFE0 :8MByte : PowerVR Tex.Mem. 64bit access area(Image area)  
 0x07000000~0x077FFFFE0 :8MByte : PowerVR Tex.Mem. 32bit access area(Image area)  
 0x0C000000~0x0CFFFFFE0 :16Mbyte System Memory  
 0x0D000000~0x0DFFFFFE0 :16Mbyte System Memory(Not supported)  
 0x0E000000~0x0EFFFFFE0 :16Mbyte System Memory (Image area)  
 0x0F000000~0x0FFFFFE0 :16Mbyte System Memory(Not supported)  
 0x14000000~0x17FFFFFE0 :64Mbyte G2 External Devices #3

**Note**

- This register is not initialized after a power-on reset or a software reset.

**SB\_GDLEND**

**(Read Only)**

Address : 0x005F 74F8

bit 31-25	24-5	4-0
Reserved	GD-DMA remainder	Reserved

This register returns the size of the GD-DMA transfer in bytes. (Note that this register counts up.)

***GD-DMA Remainder***

0x00000020 : 32 Byte  
 0x00000040 : 64 Byte  
 :  
 0x01FFFFFE0 : 32M Byte—32 Byte  
 0x000000000 : 32M Byte

**Note**

- This register is not initialized after a power-on reset or a software reset.

#### § 8.4.1.4 G2 Interface

The modem unit connects to the same G2 Bus as the AICA, etc., but because it only exchanges data with the SH4 via single reads and writes, it does not have any DMA control registers. For details on accessing the modem, refer to section 4.2, "Modem," or the modem unit specifications.

There are four channels of DMA engines for the G2 interface, but they all use the same types of units. Only the DMA hardware triggers are different.

AICA-DMA (cho), Ext1-DMA (ch1), Ext2-DMA (ch2), Dev-DMA (ch3)

Note that these are functionally the same, so the explanations for the numerous G2-DMA registers have been grouped together, and the explanations that apply to each channel are summarized in the description of AICA-DMA.

G2-DMA Engine →	AICA (cho)	External 1 (ch1)	External 2 (ch2)	DevTools (ch3)
G2-DMA start address	<b>ADSTAG</b>	<b>E1STAG</b>	<b>E2STAG</b>	<b>DDSTAG</b>
G2-DMA Sys.Mem. or Tex.Mem. address	<b>ADSTAR</b>	<b>E1STAR</b>	<b>E2STAR</b>	<b>DDSTAR</b>
G2-DMA Length	<b>ADLEN</b>	<b>E1LEN</b>	<b>E2LEN</b>	<b>DDLEN</b>
G2-DMA Direction	<b>ADDIR</b>	<b>E1DIR</b>	<b>E2DIR</b>	<b>DDDIR</b>
G2-DMA Trigger selection	<b>ADTSEL</b>	<b>E1TSEL</b>	<b>E2TSEL</b>	<b>DDTSEL</b>
G2-DMA Enable	<b>ADEN</b>	<b>E1EN</b>	<b>E2EN</b>	<b>DDEN</b>
G2-DMA Start and Status	<b>ADST</b>	<b>E1ST</b>	<b>E2ST</b>	<b>DDST</b>
G2-DMA Suspend Request and Status	<b>ADSUSP</b>	<b>E1SUSP</b>	<b>E2SUSP</b>	<b>DDSUSP</b>
Hardware trigger signal of G2	<b>G2RQAIC</b> #	<b>G2RQEXO</b> #	<b>G2RQEX1</b> #	<b>G2RQDEV</b> #

***(The G2 DMA Control Registers are described below.)***

<b>SB_ADSTAG</b>	Address : 0x005F 7800
<b>SB_E1STAG</b>	Address : 0x005F 7820
<b>SB_E2STAG</b>	Address : 0x005F 7840
<b>SB_DDSTAG</b>	Address : 0x005F 7860

bit 31-29	28-5	4-0
000	G2-DMA G2 start address	Reserved

These registers specify starting address for G2-DMA transfers involving External-1, External-2, External-3, wave memory, and the AICA register. The G2-DMA transfer is performed between areas specified by these registers and the *SB\_ADSTAR*, *SB\_E1STAR*, *SB\_E2STAR*, and *SB\_DDSTAR* registers.

0x00700000~0x00707FE0 :32KByte : G2 AICA Registers

0x00800000~0x009FFFFE0 :2MByte : G2 Wave Memory

0x02700000~0x02FFFFFE0 :9MByte : G2 AICA (Image area)

\* The 3 areas listed above are addresses that can be specified for channel 0 (AICA-DMA).

0x01000000~0x01FFFFFE0 :16MByte : G2 External Devices #1

0x03000000~0x03FFFFFE0 :16MByte : G2 External Devices #2

0x14000000~0x17FFFFFE0 :64MByte : G2 External Devices #3

\* The 3 areas listed above are addresses can be specified for channels 1 through 3.

Note:

- This register is not initialized after a power-on reset or a software reset.
- The address value must be specified in 32-byte units.
- When the DMA enable register (*SB\_ADEN*, etc.) is "0", the G2-DMA block's internal values are updated.
- If a value that is not included above is set, an invalid setting interrupt is generated.

<b>SB_ADSTAR</b>	Address : 0x005F 7804
<b>SB_E1STAR</b>	Address : 0x005F 7824

**SB\_E2STAR**  
**SB\_DDSTAR**

Address : 0x005F 7844  
Address : 0x005F 7864

bit 31-29	28-5	4-0
000	G2-DMA System Mem. or Texture Mem. start address	Reserved

This register specifies the starting address for G2 DMA transfers involving system memory or texture memory. The G2 DMA transfer is performed between areas specified by these registers and the *SB\_ADSTAG*, *SB\_E1STAG*, *SB\_E2STAG*, and *SB\_DDSTAG* registers. When the DMA enable register (*SB\_ADEN*, etc.) is "0", the value in the G2-DMA block is updated.

0x0C000000~0x0CFFFFE0 :16Mbyte : System Memory  
 0x04000000~0x047FFFFE0 :8Mbyte : Texture Mem. 64bit access area  
 0x04800000~0x04FFFFE0 :8Mbyte : Texture Mem. 64bit access area (the latter half)  
 0x05000000~0x057FFFFE0 :8Mbyte : Texture Mem. 32bit access area

Note:

- This register is not initialized after a power-on reset or a software reset.
- The address value must be specified in 32-byte units.
- When the DMA enable register (*SB\_ADEN*, etc.) is "0", the G2-DMA block's internal values are updated.
- If a value that is not included above is set, an invalid setting interrupt is generated.

**SB\_ADLEN**  
**SB\_E1LEN**  
**SB\_E2LEN**  
**SB\_DDLLEN**

Address : 0x005F 7808  
Address : 0x005F 7828  
Address : 0x005F 7848  
Address : 0x005F 7868

bit 31	30-25	24-5	4-0
DMA Transfer End/Restart	Reserved	G2-DMA Transfer Length	Reserved

**DMA Transfer End//Restart**

This field sets the DMA transfer operation.

Setting	Meaning
0	DMA restart (Restart after a DMA transfer ended) * When a transfer ends, the DMA enable register remains set to "1".
1	DMA end * When a transfer ends, the DMA enable register is set to "0".

**Transfer Length**

This field specifies the G2-DMA data transfer length in 32-byte units.

Setting (32bit)	Transmission Length
0x00000020	32Byte
0x00000040	64Byte
.....	.....
0x01FFFFE0	32MByte – 32Byte
0x00000000	32MByte

Note:

- This register is not initialized after a power-on reset or a software reset.
- When the DMA enable register (*SB\_ADEN*, etc.) is "0", the value in the G2-DMA block is updated.

<b>SB_ADDIR</b>	Address : 0x005F 780C
<b>SB_E1DIR</b>	Address : 0x005F 782C
<b>SB_E2DIR</b>	Address : 0x005F 784C
<b>SB_DDIR</b>	Address : 0x005F 786C

bit 31-1	0
Reserved	G2-DMA transfer direction

This register specifies the direction of DMA transfers between the Root Bus and G2 devices. The G2 device indicated here is the device that corresponds to the area specified by the *SB\_ADSTAR*, *SB\_E1STAR*, *SB\_E2STAR*, or *SB\_DDSTAR* register.

Setting	Meaning
0	DMA transfer from the Root Bus to a G2 device
1	DMA transfer from a G2 device to the Root Bus

Note:

- This register is not initialized after a power-on reset or a software reset.
- When the DMA enable register (*SB\_ADEN*, etc.) is "0", the value in the G2-DMA block is updated.

<b>SB_ADTSEL</b>	Address : 0x005F 7810
<b>SB_E1TSEL</b>	Address : 0x005F 7830
<b>SB_E2TSEL</b>	Address : 0x005F 7850
<b>SB_DDTSEL</b>	Address : 0x005F 7870

bit 31-3	2-0
Reserved	Trigger selection

This register specifies the G2-DMA transfer trigger.

**bit2 : Trigger Selection2**

- 0: Disables the DMA suspend function
- 1: Enables the DMA suspend function (In the case of AICA-DMA, the *SB\_ADSUSP* register is enabled.)

**bit1 : Trigger Selection1**

- 0: CPU initiation (DMA transfer is initiated by writing to the *SB\_\*\*ST* register in the SH4.)
- 1: Hardware trigger (DMA transfer is initiated according to the interrupt setting)

**bito : Trigger Selection0**

- 0: Disables control of transfer through an external pin (transfer request input) (→ continuous transfer)
- 1: Enables control of transfer through an external pin

The DMA mode combinations are listed below. (The register names shown are for AICA-DMA.)

SB_ADLEN	SB_ADTSEL			Meaning
bit31	bit2	bit1	bit0	
0	0	0	0	CPU initiation
	0	0	1	Prohibited
	0	1	0	Interrupt initiation
	0	1	1	Prohibited
1	0	0	0	When DMA ends, <i>SB_ADEN</i> = 0 + CPU initiation
	0	0	1	When DMA ends, <i>SB_ADEN</i> = 0 + CPU initiation + external pin control
	0	1	0	When DMA ends, <i>SB_ADEN</i> = 0 + CPU initiation + interrupt initiation
	0	1	1	Prohibited
0	1	0	0	Suspend enabled + CPU initiation
	1	0	1	Suspend enabled + CPU initiation + external pin control
	1	1	0	Suspend enabled + interrupt initiation
	1	1	1	Suspend enabled + interrupt initiation + external pin control
1	1	0	0	Suspend enabled + When DMA ends, <i>SB_ADEN</i> = 0 + CPU initiation
	1	0	1	Suspend enabled + When DMA ends, <i>SB_ADEN</i> = 0 + CPU initiation + external pin control
	1	1	0	Suspend enabled + When DMA ends, <i>SB_ADEN</i> = 0 + interrupt initiation
	1	1	1	Suspend enabled + When DMA ends, <i>SB_ADEN</i> = 0 + interrupt initiation + external pin control

Here, "CPU initiation" means that G2-DMA is initiated through software by the SH4; each type can be initiated by writing "1" to either the *SB\_ADST*, *SB\_E1ST*, *SB\_E2ST*, or *SB\_E3ST* register.

External pin control permits initiation upon reception of a trigger from a device that is connected on the HOLLY's G2 bus. (The transfer request input signal **G2RQAIC#** is input to the G2-AICA-DMA engine as an external trigger. In addition, the signals **G2RQEX0#**, **G2RQEX1#**, and **G2RQDEV#** are each input as external triggers to the G2-External1, External2, and Dev.Tools DMA engines, respectively. For details on how to drive these signals, refer to the AICA specifications.)

"Interrupt initiation" refers to the automatic initiation of G2-DMA in response to the interrupt designated by the *SB\_G2DTNRM* register and the *SB\_G2DTEXT* register. When the interrupts designated by both registers are generated, G2-DMA is initiated by the interrupt that was generated first.

Bit 2 of this register indicates whether the G2 suspend function is enabled or not. For details on the suspend function, refer to the description of the *SB\_ADSUSP* register (for AICA-DMA).

Notes:

- This register is not initialized after a power-on reset or a software reset.
- When the DMA enable register (*SB\_ADEN*, etc.) is "0", the value in the G2-DMA block is updated.

<b>SB_ADEN</b>	Address : 0x005F 7814
<b>SB_E1EN</b>	Address : 0x005F 7834
<b>SB_E2EN</b>	Address : 0x005F 7854
<b>SB_DDEN</b>	Address : 0x005F 7874

bit 31-1	0
Reserved	G2-DMA enable

This register enables G2-DMA.

G2-DMA is forcibly terminated by writing a "0" to this register while G2-DMA is in progress.

When writing	
Setting	Meaning
0	Disables G2-DMA. (default)
1	Enables G2-DMA.

When reading	
Setting	Meaning
0	G2-DMA is disabled. (default)
1	G2-DMA is enabled

<b>SB_ADST</b>	Address : 0x005F 7818
<b>SB_E1ST</b>	Address : 0x005F 7838
<b>SB_E2ST</b>	Address : 0x005F 7858
<b>SB_DDST</b>	Address : 0x005F 7878

bit 31-1	0
Reserved	G2-DMA start/status

This register initiates G2-DMA when the transfer initiation registers (*SB\_ADTSEL*, etc.) are set to permit initiation by the SH4.

The DMA status can be determined by reading this register. When G2-DMA is disabled through the *SB\_ADEN* (in the case of AICA-DMA) register, writing a "1" to this register is not allowed.

When writing	
Setting	Meaning
0	Prohibited
1	Initiates DMA.

When reading	
Setting	Meaning
0	DMA not in progress
1	DMA in progress

Note:

- If an invalid value is set in the *SB\_ADSTAG* or *SB\_ADSTAR* register (when both are set for cho-AICA), and then G2-DMA is enabled, an invalid setting interrupt is generated.

<b>SB_ADSUSP</b>	Address : 0x005F 781C
<b>SB_E1SUSP</b>	Address : 0x005F 783C
<b>SB_E2SUSP</b>	Address : 0x005F 785C
<b>SB DDSUSP</b>	Address : 0x005F 787C

bit 31-6	5-0
Reserved	DMA Suspend Request/Status

This register temporarily stops G2-DMA. This register is valid when bit 2 in the *SB\_ADTSEL* (in the case of AICA-DMA) register is "1" (invalid when "0"). The value of these bits after a reset is determined by signals from external devices.

**bit5 : DMA Request Input State (Read Only)**

- 0: The DMA transfer request is high (transfer not possible), or bit 2 of the *SB\_ADTSEL* register is "0"
- 1: The DMA transfer request is low (transfer possible)

**bit4 : DMA Suspend or DMA Stop (Read Only)**

- 0: DMA transfer is in progress, or bit 2 of the *SB\_ADTSEL* register is "0"  
1: DMA transfer has ended, or is stopped due to a suspend

\* When bit 2 of the *SB\_ADTSEL* register is "1" and bit 0 of the *SB\_ADSUSP* register is "1", and data is not being transferred due to being in the suspended state, this bit becomes "1" when G2-DMA ends.

**bit3-1 : Reserved** (Specify "0".)

**bito : DMA Suspend Request** (Write Only)

- 0: Continues DMA transfer without going to the suspended state. Or, bit 2 of the *SB\_ADTSEL* register is "0"  
1: Suspends and terminates DMA transfer

**(The G2 I/F Block Hardware Control Registers are described below.)**

**SB\_G2ID (Read Only)**

Address : 0x005F 7880

bit 31-8	7-4	3-0
all '0'	HOLLY Version	G2 Version

This register returns the G2 bus version. The current versions are as follows (refer to section 1.4):

- 0x00000000 : Prototype  
0x00000012 : Holly ver. 1.0 (CLX1) and later

**SB\_G2DSTO**

Address : 0x005F 7890

bit 31-12	11-0
Reserved	G2 DS time out

This is a special register that is used for debugging. This register sets the DS# signal timeout time for G2 devices.

**G2 DS TIMEOUT (default - 0x3FF)**

When there is no response from the DS# signal for a certain period of time, a timeout error interrupt is generated.

- 0x000: Shortest wait time (after 0 clocks)  
:  
0xFFFF: Longest wait time (after 4095 clocks)

The respective G2-DMA timeout error interrupts for AICA, External 1/2, and Dev tool are the same as in the case of the *SB\_G2TRTO* register. Refer to bits 26 through 23 in the *SB\_ISTERR* register.

Notes:

- The setting in this register affects all G2 timeout error interrupts (bits 27 through 23 in the *SB\_ISTERR* register).
- If either the DS# signal or the TR# signal times out, it is then necessary to check that the target device is actually connected on the G2 bus, and that the device has not failed.

### SB\_G2TRTO

Address : 0x005F 7894

bit 31-12	11-0
Reserved	G2 TR time out

This is a special register that is used for debugging. This register sets the TR# signal timeout time for G2 devices.

#### **G2 TR TIMEOUT (default - 0x3FF)**

When there is no response from the TR# signal for a certain period of time, a timeout error interrupt is generated.

0x000: Shortest wait time (after 0 clocks)

:

0xFFFF: Longest wait time (after 4095 clocks)

The respective G2-DMA timeout error interrupts for AICA, External 1/2, and Dev tool are the same as in the case of the SB\_G2DSTO register. Refer to bits 26 through 23 in the SB\_ISTERR register.

#### Notes:

- The setting in this register affects all G2 timeout error interrupts (bits 27 through 23 in the SB\_ISTERR register).
- If either the DS# signal or the TR# signal times out, it is then necessary to check that the target device is actually connected on the G2 bus, and that the device has not failed.

### SB\_G2MDMTO

Address : 0x005F 7898

bit 31-8	7-0
Reserved	G2 Modem Timeout

This register sets wait insertion for asynchronous cycles (modem).

Setting	Meaning
0x00	External wait input disabled (default)
0x01	Setting prohibited
:	:
0xFE	Setting prohibited
0xFF	External wait input disabled

#### Notes:

- Setting a value from 0x01 to 0xFE in bits 7 through 0 is prohibited.
- If a modem will not be used, setting 0x00 in bits 7 through 0 is recommended; if a modem will be used, setting 0xFF in bits 7 through 0 is recommended.
- If the wait lasts 10.2μs or more, the operation times out and access is terminated. For details on timeout errors, refer to bit 27 of the SB\_ISTERR register.



**SB\_G2MDMW**

Address : 0x005F789C

bit 31-8	7-0
Reserved	G2 Modem Wait

This register sets the wait time for asynchronous cycles (modem).

Setting	Meaning
0x00	80 nsec (default)
0x01	120 nsec
0x02	160 nsec
.....	Each increase of 0x01 in the setting extends the wait time by 40ns.

Notes:

- Setting a value that results in a wait longer than 1μsec (0x17) is prohibited.
- In the case of a device that requires an access time in excess of 1μs, initiate the access after the write FIFO has become empty. In this case, confirm that bit 4 of the *SB\_FFST* (0x005F688C) register is "0". If an access is made without confirming this setting, an unnecessary wait will be generated for the SH4.
- In interrupt processing, when the G2 bus is accessed, the first process must be to confirm that the write FIFO is empty. (As in the previous item, check bit 4 of the *SB\_FFST* register.)
- This register requires that the modem wait insertion setting be made in the *SB\_G2MDMTO* register.

***(The G2-DMA Secret Registers is described below.)*****SB\_G2APRO (Write Only)**

Address : 0x005F78BC

bit 31-16	15	14-8	7	6-0
Security code 0x4659	R	Top address	R	Bottom address

This register specifies the address range for G2-DMA involving the system memory.  
This setting is common to channels 0 through 3.

***Security code 0x4659***

When updating bits 14 through 8 and bits 6 through 0, it is necessary to add "0x4659." (default = 0x0000) If this value is not added, bits 14 through 8 and bits 6 through 0 will not be updated.

***Top address***

This field specifies the starting address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x7F)

***Bottom address***

This field specifies the ending address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x00)

Notes:

- If a DMA transfer is generated outside of this range, an overrun error results and the G2-DMA overrun error interrupt is generated. (Refer to bits 22 through 19 of the *SB\_ISTERR* register.)

**(The G2-DMA Debug Registers are described below.)**

The following registers are special registers that are used for debugging purposes.

**SB\_ADSTAGD (Read Only)** Address : 0x005F 78C0  
**SB\_E1STAGD (Read Only)** Address : 0x005F 78D0  
**SB\_E2STAGD (Read Only)** Address : 0x005F 78E0  
**SB\_DDSTAGD (Read Only)** Address : 0x005F 78F0

bit 31-29	28-5	4-0
Reserved	G2-DMA address Counter	Reserved

This register can be used to check the current G2 device address in a G2-DMA transfer. For details concerning areas, refer to the *SB\_ADSTAG*, *SB\_E1STAG*, *SB\_E2STAG*, and *SB\_DDSTAG* registers.

Note:

- This register is not initialized after a power-on reset or a software reset.

**SB\_ADSTARD (Read Only)** Address : 0x005F 78C4  
**SB\_E1STARD (Read Only)** Address : 0x005F 78D4  
**SB\_E2STARD (Read Only)** Address : 0x005F 78E4  
**SB\_DDSTARD (Read Only)** Address : 0x005F 78F4

bit 31-29	28-5	4-0
Reserved	G2-DMA System Mem. or Texture Mem. address Counter	Reserved

This register can be used to check the current system memory or texture memory address in a G2-DMA transfer. For details concerning areas, refer to the *SB\_ADSTAG*, *SB\_E1STAG*, *SB\_E2STAG*, and *SB\_DDSTAG* registers.

Note:

- This register is not initialized after a power-on reset or a software reset.

**SB\_ADLEND (Read Only)** Address : 0x005F 78C8  
**SB\_E1LEND (Read Only)** Address : 0x005F 78D8  
**SB\_E2LEND (Read Only)** Address : 0x005F 78E8  
**SB\_DDLLEND (Read Only)** Address : 0x005F 78F8

bit 31-25	24-5	4-0
Reserved	G2-DMA length remain Counter	Reserved

This register is used to check the current amount of data remaining in a G2-DMA transfer. Note that this value returns to its original setting immediately after DMA terminates.

***Length remainder***

0x00000020 : 32 Byte  
0x00000040 : 64 Byte  
:  
0x01FFFFE0 : 32M Byte minus 32 Byte  
0x00000000 : 32M Byte

Note:

- This register is not initialized after a power-on reset or a software reset.

### § 8.4.1.5 PowerVR Interface

***(The PVR-DMA Control Registers are described below.)***

#### **SB\_PDSTAP**

Address : 0x005F 7C00

bit 31-29	27-5	4-0
000	PVR-DMA start address on PVR	Reserved

This register specifies the starting address on the PVR side for cho-DMA involving PVR. Data transfers between the system memory and the following PVR areas are possible using cho-DMA.

0x005F8000~0x005F9FE0 :8KByte : PowerVR registers  
 0x025F8000~0x025F9FE0 :8KByte : PowerVR registers (Image area)  
 0x04000000~0x047FFFE0 :8MByte : PowerVR Tex.Mem. 64bit access area  
 0x05000000~0x057FFFE0 :8MByte : PowerVR Tex.Mem. 32bit access area  
 0x06000000~0x067FFFE0 :8MByte : PowerVR Tex.Mem. 64bit access area (Image area)  
 0x07000000~0x077FFFE0 :8MByte : PowerVR Tex.Mem. 32bit access area (Image area)

**Note:**

- This register is not initialized after a power-on reset or a software reset.
- The start address must be specified in 32-byte units.  
If the specified value is outside of the specifiable area, the "PVR i/f: Illegal Address set" error interrupt is generated, and the DMA transfer is not performed. (Refer to bit 6 of the *SB\_ISTERR* register.)
- The hardware does not change the data in this register.
- Even while a DMA operation is in progress, the next address can be set without affecting the current DMA operation.

#### **SB\_PDSTAR**

Address : 0x005F 7C04

bit 31-28	27-5	4-0
000	PVR-DMA start address on System Memory	Reserved

This register specifies the starting address on the system memory side for cho-DMA involving PVR.

**Note:**

- This register is not initialized after a power-on reset or a software reset.
- The start address must be specified in 32-byte units.  
If the specified value is outside of the specifiable area, the "PVR i/f: Illegal Address set" error interrupt is generated, and the DMA transfer is not performed. (Refer to bit 6 of the *SB\_ISTERR* register.)
- The hardware does not change the data in this register.
- Even while a DMA operation is in progress, the next address can be set without affecting the current DMA operation.

**SB\_PDLLEN**

Address : 0x005F 7Co8

bit 31-24	23-5	4-0
Reserved	PVR-DMA Length	Reserved

This register specifies the length of cho DMA between PVR and system memory in 32-byte units.

Setting (32bit)	Length
0x00000020	32 byte
0x00000040	64 byte
.....	.....
0x00FFFE0	16M byte – 32 byte
0x00000000	16M byte

Notes:

- This register is not initialized after a power-on reset or a software reset.
- The length must be specified in 32-byte units.
- If the PVR-DMA length is too large and the transfer exceeds the work area in system memory that was specified by the *SB\_PDAPRO* register, an error results, the "PVR i/f: DMA overrun" error interrupt is generated, and the DMA transfer is not performed. (Refer to bit 7 of the *SB\_ISTERR* register.)
- The hardware does not change the data in this register.
- Even while a DMA operation is in progress, the next address can be set without affecting the current DMA operation.

**SB\_PDDIR**

Address : 0x005F 7CoC

bit 31-1	0
Reserved	PVR-DMA direction

This register specifies the direction of a PVR-DMA (cho-DMA) transfer to the area set in the *SB\_PDSTAP* register.

Setting	Meaning
0	From system memory to PVR area (default)
1	From PVR area to system memory

**SB\_PDTSEL**

Address : 0x005F 7C10

bit 31-1	0
Reserved	PVR-DMA trigger selection

This register selects the PVR-DMA trigger.

Setting	Meaning
0	Software trigger (default) The SH4 triggers PVR-DMA manually. PVR-DMA can be triggered by writing to the <i>SB_PDST</i> register.
1	Hardware trigger... PVR-DMA is automatically initiated by the interrupts set in the <i>SB_PDTNRM</i> and <i>SB_PDTEXT</i> registers. If the interrupts set in both registers are both generated, initiation is triggered by the interrupt that was generated first.

Note:

- The hardware does not change the data in this register.
- Even while a DMA operation is in progress, the next address can be set without affecting the current DMA operation.

### SB\_PDEN

Address : 0x005F 7C14

bit 31-1	0
Reserved	PVR-DMA enable

This register enables PVR-DMA. Initiation sources are not accepted until a "1" is written to this bit.

PVR-DMA is forcibly terminated by writing a "0" to this register while PVR-DMA is in progress.

When writing	
Setting	Meaning
0	Abort PVR-DMA (default)
1	Enable

When reading	
Setting	Meaning
0	Disable (default)
1	Enable

Note:

- When "software trigger" is selected, DMA is triggered by writing a "1" to the *SB\_PDST* register.  
When "hardware trigger" is selected, DMA is triggered by the interrupts that are set in the *SB\_PDTNRM* and *SB\_PDTEXT* registers.
- The hardware does not change the data in this register.

### SB\_PDST

Address : 0x005F 7C18

bit 31-1	0
Reserved	PVR-DMA start/status

This register requests the start of PVR-DMA. The status of the DMA transfer can be checked by reading this register.

DMA is initiated by setting the *SB\_PDEN* register to "Enable" (1) and then writing a "1" to this bit. If the *SB\_PDEN* register is set to "Disable," then writing a "1" to this bit has no effect.

When writing	
Setting	Meaning
0	ignored (default)
1	Start DMA

When reading	
Setting	Meaning
0	PVR-DMA not in progress. (default)
1	PVR-DMA in progress.

***(The PVR-DMA Secret Registers are described below.)***

**SB\_PDAPRO (Write Only)**

Address : 0x005F 7C80

bit 31-16	15	14-8	7	6-0
security code 0x6702	R	Top address	R	Bottom address

This register specifies the address range for PVR-DMA involving system memory.

***Security code 0x6702***

When updating bits 14 through 8 and bits 6 through 0, it is necessary to add "0x6702." If this value is not added, bits 14 through 8 and bits 6 through 0 will not be updated. (default = 0x0000)

***Top address***

This field specifies the starting address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0x7F)

***Bottom address***

This field specifies the ending address of the address range in system memory. (This field corresponds to A26 to A20; A28 and A27 are treated as "0x01".) Specify the address in units of 1MB. (default = 0)

***Note:***

- PVR-DMA involving system memory will be performed only within the above address range.  
If a DMA transfer is generated outside of this range, an overrun error results and the PVR-DMA overrun error interrupt is generated. (Refer to bit 7 of the *SB\_ISTERR* register.)

## § 8.4.2 CORE Registers

### ID (Read Only)

Address : 0x005F 8000

bit 31-16	15-0
Device ID (0x17FD)	Vender ID (0x11DB)

This register returns the device ID and the vendor ID.

### REVISION (Read Only)

Address : 0x005F 8004

bit 31-16	15-0
Reserved	Chip Revision

This register indicates the chip revision number. (0x01 in the case of an ES.) For details on the CORE revision, refer to section 1.4.

### SOFT RESET

Address : 0x005F 8008

bit 31-3	2	1	0
Reserved	sdram I/F soft reset	Pipeline soft reset	TA soft reset

This register specifies a soft reset. After power is supplied to the system, the system enters this reset state.

Setting	Meaning
0	Not reset
1	Reset (default)

#### ***sdram interface soft reset***

This field resets the texture memory interface.

#### ***Pipeline soft reset***

This field resets the CORE pipeline.

#### ***TA soft reset***

This field resets the Tile Accelerator.

### STARTRENDER

Address : 0x005F 8014

bit 31-0
Start Render

Writing to this register initiates rendering. If rendering is to be started again before the rendering end interrupt (End of TSP) is output, a CORE reset must be executed through the SOFTRESET register. The CORE reset time in this case is at least 32 clocks.

### TEST\_SELECT

Address : 0x005F 8018

bit 31-9	9-5	4-0
Reserved	diagdb_data	diagda_data

This is a test register. Writing to this register is prohibited.

### PARAM\_BASE

Address : 0x005F 8020

bit 31-24	23-20	19-0
Reserved	Base Address	0000 0000 0000 0000 000

This register specifies, in 1MB units, the base address for the ISP/TSP Parameters that the CORE loads in for drawing (default = 0x0). The absolute address of the ISP/TSp parameter that is read is the sum of the object start address in the Object List data and this base address.

### REGION\_BASE

Address : 0x005F 802C

bit 31-24	23-2	1-0
Reserved	Base Address	00

This register specifies, in 32-bit units, the base address for the Region Array that the CORE loads in for drawing. (default = 0x0000000)

### SPAN\_SORT\_CFG

Address : 0x005F 8030

bit 31-17	16	15-9	8	7-1	0
Reserved	Cache Bypass	Reserved	Offset Sort enable	Reserved	Span Sort enable

#### **Cache Bypass**

This field specifies whether or not to use the TSP cache bypass function.

Setting	Meaning
0	Use cache (default)
1	Bypass Cache

#### **Offset Sort enable**

#### **Span Sort enable**

This field specifies whether or not to use the Span Sort function.

Offset	Span	Meaning
0	0	Do not use the Span Sort function. (default)
0	1	Group those items that have the same offset value.
1	0	Setting prohibited.
1	1	First group by tag values, and then group by offset values. This setting is normally recommended. Span sort minimizes TS preprocessing, and offset sort optimizes TSP parameter fetching and reduces the bus bandwidth.

### VO\_BORDER\_COL

Address : 0x005F 8040

bit 31-25	24	23-16	15-8	7-0
Reserved	Chroma	Red	Green	Blue

This register specifies the color that is displayed in the border area. (default = 0x000 0000)



**FB\_R\_CTRL**

Address : 005F 8044h

bit 31-24	23	22	21-16	15-8	7	6-4	3-2	1	0
Reserved	vlck_ div	fb_ strip_ buf_en	fb_ stripsize	fb_ chroma_ threshold	R	fb_ concat	fb_ depth	fb_ line_ double	fb_ enable

This register makes settings for frame buffer reads.

**vlck\_div**

This field specifies the clock that is output on the PCLK pin.

Setting	Meaning	Supplement
0	PCLK = VLCK/2 (default)	For NTSC/PAL
1	PCLK = VLCK	For VGA

**fb\_strip\_buf\_en**

Set this bit to "1" when using a strip buffer. (default = 0)

**fb\_stripsize**

This field specifies the size of the strip buffer in units of 32 lines (default = 0x00). "0" must be specified for the LSB (bit 16). For example, specify 0x02 for 32 lines, and 0x04 for 64 lines. Furthermore, the strip buffer size must yield an even number when the number of lines on the display screen is divided by the strip buffer size.

**fb\_chroma\_threshold**

When the frame buffer data format is ARGB8888, this field sets the comparison value that is used in order to determine the CHROMA pin output value. (default = 0x00)

When pixel alpha < fb\_chroma\_threshold, a "0" is output on the CHROMA pin.

**fb\_concat**

This field specifies the value that is added to the lower end of 5-bit or 6-bit frame buffer color data in order to output 8 bits. (default = 0x0)

**fb\_depth**

This field specifies the bit configuration of the pixel data that is read from the frame buffer.

Setting	Meaning	Supplement
0x0	0555 RGB 16 bit (default)	The lower 3 bits are the value in fb_concat.
0x1	565 RGB 16 bit	The lower 3 bits of R and G are the value in fb_concat; the lower 2 bits in G are the value in fb_concat[1:0].
0x2	888 RGB 24 bit packed	
0x3	0888 RGB 32 bit	

**fb\_line\_double**

This field specifies the read operation for each line of frame buffer data.

Setting	Meaning
0	Reads each line once. (default)
1	Reads each line twice.

**fb\_enable**

This field enables or disables frame buffer data reads.

Setting	Meaning
0	disable (default)
1	enable

### FB\_W\_CTRL

Address : 0x005F 8048

bit 31-24	23-16	15-8	7-4	3	2-0
Reserved	fb_alpha_threshold	fb_kval	Reserved	fb_dither	fb_packmode

This register contains settings concerning frame buffer writes.

#### ***fb\_alpha\_threshold***

This field sets the comparison value that is used to determine the alpha value when the data that is written to the frame buffer is ARGB1555 format data. (default = 0x00)

When pixel alpha  $\geq$  fb\_alpha\_threshold, a "1" is written in the alpha value.

#### ***fb\_kval***

This field sets the K value for writing to the frame buffer. (default = 0x00)

#### ***fb\_dither***

This field specifies whether dithering is applied upon writing frame buffer data that consists of 16 bits /pixel.

Setting	Meaning
0	Discard the lower bits. (default)
1	Perform dithering

#### ***fb\_packmode***

This field specifies the bit configuration of the pixel data that is written to the frame buffer.

Setting	Meaning	Supplement
0x0	0555 KRGB 16 bit (default)	Bit 15 is the value of fb_kval[7].
0x1	565 RGB 16 bit	
0x2	4444 ARGB 16 bit	
0x3	1555 ARGB 16 bit	The alpha value is determined by comparison with the value of fb_alpha_threshold.
0x4	888 RGB 24 bit packed	
0x5	0888 KRGB 32 bit	K is the value of fb_kval.
0x6	8888 ARGB 32 bit	
0x7	Reserved	Setting prohibited.

### FB\_W\_LINESTRIDE

Address : 0x005F 804C

bit 31-9	8-0
Reserved	FB line stride

This register specifies the line width, in 64-bit units, when writing to the frame buffer. (default = 0x000)

<b>FB_R_SOF1</b>		Address : 0x005F 8050
bit 31-24	23-2	1-0
Reserved	Frame Buffer Read Address Frame 1	00

This register specifies the starting address, in 32-bit units, for reads from the field-1 frame buffer. (default = 0x000000) The setting becomes valid starting from the next field.

<b>FB_R_SOF2</b>		Address : 0x005F 8054
bit 31-24	23-2	1-0
Reserved	Frame Buffer Read Address Frame 1	00

If interlace mode is specified, this register specifies the starting address, in 32-bit units, for reads from the field-2 frame buffer. (default = 0x000000)

**FB\_R\_SIZE**
Address : 0x005F 805C

bit 31-30	29-20	19-10	9-0
Reserved	FB modulus	FB y size	FB x size

This register specifies the frame buffer size when reading from the frame buffer.

#### **FB modulus**

This field specifies the amount of data between the last pixel on a line and the first pixel on the next line, in 32-bit units (default = 0x0000). Set this value to 0x0001 in order to link the last pixel data on a line with the first pixel data on a line.

#### **FB y size**

This field specifies the number of display lines - 1. (default = 0x0000)

#### **FB x size**

This field specifies the number of display pixels on each line - 1, in 32-bit units. (default = 0x0000)

<b>FB_W_SOF1</b>		Address : 0x005F 8060
bit 31-25	24-2	1-0
Reserved	Frame Buffer Write start of Frame Address field1/strip1	00

This register specifies, in 32-bit units, the starting address for writes to the field-1 or strip-1 frame buffer. (default = 0x000000)

In the texture map, 0x00000000 to 0x0FFFFFFFC is a 32-bit access area and 0x10000000 to 0x1FFFFFFFC is a 64-bit access area.

<b>FB_W_SOF2</b>		Address : 0x005F 8064
bit 31-25	24-2	1-0
Reserved	Frame Buffer Write start of Frame Address field2/strip2	00

This register specifies, in 32-bit units, the starting address for writes to the field-1 or strip-2 frame buffer. (default = 0x000000)

In the texture map, 0x00000000 to 0x0FFFFFFFC is a 32-bit access area (frame/strip buffer) and 0x10000000 to 0x1FFFFFFFC is a 64-bit access area.

### FB\_X\_CLIP

Address : 0x005F 8068

bit 31-27	26-16	15-11	10-0
Reserved	FB x clipping max	Reserved	FB x clipping min

This register specifies the clipping values for the X direction when writing to the frame buffer. When using a strip buffer, this register must be specified in accordance with the display screen size. For example, if the size of the display screen in the horizontal direction is 640 pixels, then specify Max. = 639 and Min. = 0.

#### **FB x clipping max**

Pixels with an X coordinate that is greater than the value specified in this field are not written to the frame buffer. (default = 0x000)

#### **FB x clipping min**

Pixels with an X coordinate that is smaller than the value specified in this field are not written to the frame buffer. (default = 0x000)

### FB\_Y\_CLIP

Address : 0x005F 806C

bit 31-26	25-16	15-10	9-0
Reserved	FB y clipping max	Reserved	FB y clipping min

This register specifies the clipping values for the Y direction when writing to the frame buffer.

#### **FB y clipping max**

Pixels with a Y coordinate that is greater than the value specified in this field are not written to the frame buffer. (default = 0x000)

#### **FB y clipping min**

Pixels with a Y coordinate that is smaller than the value specified in this field are not written to the frame buffer. (default = 0x000)

### FPU\_SHAD\_SCALE

Address : 0x05F 8074

bit 31-9	8	7-0
Reserved	Intensity Shadow Enable	Scale factor for shadows

This register sets the inclusion/exclusion volume mode.

#### **Simple Shadow Enable**

This field specifies the volume mode.

Setting	Meaning
0	Parameter Selection Volume Mode (default)
1	Intensity Volume Mode

#### **Scale factor for shadows**

When using Intensity Shadow Mode, this field specifies the intensity value that is multiplied with the Shading Color data (Base Color, Offset Color). (default = 0x00) The Base Color and Offset Color are both multiplied by the same value; that value is [Scale Factor]/256.

---

**FPU\_CULL\_VAL**

Address : 0x005F 8078

bit 31	30-0
Reserved	IEEE floating point value for culling compare

This register specifies the comparison value for the culling operation. (default = 0x0000 0000)

**FPU\_PARAM\_CFG**

Address : 0x005F 807C

bit 31-22*	21	20	19-14	13-8	7-4	3-0
Reserved	Region Header Type	R	TSP parameter burst trigger threshold	ISP parameter burst trigger threshold	pointer burst size	pointer first burst size

This register makes settings for parameter reads.

~~\* This depends on the HOLLY version; in HOLLY2, bit 21 is added; in HOLLY1, this bit is reserved (0x0).~~

**Region Header Type**

This bit specifies the Region Array data type.

Setting	Meaning
0	5 × 32bit/Tile : Type 1 (default) The Translucent polygon sort mode is specified by the ISP_FEED_CFG register.
1	6 × 32bit/Tile : Type 2 The Translucent polygon sort mode is specified by the pre-sort bit within the Region Array data.

**TSP parameter burst trigger threshold**

When the free space in the parameter FIFO is greater than or equal to this value, a parameter read request is generated. (default = 0x1F) Setting a large value increases the burst length, but also causes numerous page breaks and forces other data read requests to wait. Never set a value that is smaller than 0x4.

**ISP parameter burst trigger threshold**

This is similar to the above TSP parameter. (default = 0x1F) Never set a value that is smaller than 0x4.

**pointer burst size**

The pointer (Object List data) is requested to be read with a burst length of this value. (default = 0x7) Specify a value that is smaller than or equal to the Object Pointer Block size that was specified by the TA\_ALLOC\_CTRL register. Specifying a large value causes numerous page breaks.

**pointer first burst size**

Specify half the value that was set in Pointer Burst Size. (default = 0x7)

### HALF\_OFFSET

Address : 0x005F 8080

bit 31-3	2	1	0
Reserved	TSP texel sampling position	TSP pixel sampling position	FPU pixel sampling position

This register specifies the sampling position in the ISP and the TSP. Normally, set identically to TSP Texel Sampling Position and TSP Pixel Sampling Position.

#### *TSP texel sampling position*

Setting	Meaning
0	(0,0)
1	(0.5,0.5) (default)

#### *TSP pixel sampling position*

Setting	Meaning
0	(0,0)
1	(0.5,0.5) (default)

#### *FPU pixel sampling position*

Setting	Meaning
0	(0,0)
1	(0.5,0.5) (default)

### FPU\_PERP\_VAL

Address : 0x005F 8084

bit 31	30-0
Reserved	IEEE floating point value for perpendicular triangle compare

This register specifies the comparison value for perpendicular polygons. (default = 0x0000 0000)

### ISP\_BACKGND\_D

Address : 0x005F 8088

bit 31-4	3-0
Background Plane depth parameter	Reserved

This register specifies the depth value for the background plane. (default = 0x0000 0000)  
Set an IEEE 32-bit floating point value with the lower four bits truncated.

---

**ISP\_BACKGND\_T**

Address : 0x005F 808C

bit 31-29	28	27	26-24	23-3	2-0
Reserved	cache bypass	shadow	skip	tag address	tag offset

This register specifies the tag parameters for the background plane. This register must set correctly before the start of drawing.

**cache bypass**

This field specifies whether or not to use the TSP parameter cache. When "0" is set (the default setting), the cache is used; when "1" is set, the cache is not used.

**shadow**

This field specifies whether or not to enable a Modifier Volume for the background plane.

Setting	Meaning
0	Disable volume. (default)
1	Enable volume.

**skip**

This field specifies the data size (\* 32 bits) for one vertex in the ISP/TSP Parameters. Normally, the actual data size is "skip + 3," but if Parameter Selection Volume Mode is in effect and the above shadow bit is "1," then the actual data size is "Skip \* 2 + 3." (default = 0x0)

ISP/TSP Instruction WordSetting			skipSetting
Texture	Offset	16bit UV	
0	Disabled	Disabled	001
1	0	0	011
1	0	1	010
1	1	0	100
1	1	1	011

**tag address**

This field specifies, in 32-bit units, the starting address of the ISP/TSP Parameters for the background plane (default = 0x000000). The absolute address of the ISP/TSP Parameter that is read is the sum of this value and the base address that is specified in the PARAM\_BASE register.

**tag offset**

This field specifies the strip number (the position within a strip of triangles) of the background plane. (default = 0x0)

## ISP\_FEED\_CFG

Address : 0x005F 8098

bit 31-24	23-14	13-4	3	2-1	0
Reserved	Cache Size for translucency	Punch Through chunk size	Discard Mode	R	pre-sort mode

This register is set when sorting polygons through the hardware. ~~Bits 31 to 1 (those marked with an asterisk) have a different configuration in HOLLY2; in HOLLY1, they are reserved bits (0x0).~~

### Cache Size for translucency

This field specifies the ISP cache size for a Translucent polygon list in terms of the number of vertices stored. (default = 0x100) In the case of Triangle polygons, the number of polygons stored is this setting divided by 3; in the case of Quad polygons, the number of polygons stored is this setting divided by 4. For example, in order to store 30 triangles,  $30 \times 3 = 90$  (0x05A) must be specified in this field.

Normally, 0x200 is specified in order to achieve maximum performance, ~~but 0x100 may be specified in order to set up the same state as CLX1.~~ However, the value must be specified so that the relationship (Cache size for translucency)  $\geq$  (Punch Through chunk size) is true. In addition, the value that is specified must be in the range from 0x020 to 0x200 (32 to 512). Do not specify a value outside the range of 0x020 to 0x200 (32 to 512).

### Punch Through chunk size

This field specifies the ISP cache size for a Punch Through polygon list in terms of the number of vertices stored. (default = 0x200) In the case of Triangle polygons, the number of polygons stored is this setting divided by 3; in the case of Quad polygons, the number of polygons stored is this setting divided by 4. For example, in order to store 30 quads,  $30 \times 4 = 120$  (0x078) must be specified in this field.

The value that is set in order to achieve maximum performance differs according to the state of the screen, but normally a value from 0x040 to 0x080 may be specified. (The recommended value is 0x040.) However, the value must be specified so that the relationship (Punch Through chunk size)  $\geq$  (Cache size for translucency) is true. In addition, the value that is specified must be in the range from 0x020 to 0x200 (32 to 512). Do not specify a value outside the range of 0x020 to 0x200 (32 to 512).

### Discard Mode

This field specifies whether to perform discard processing or not when processing Punch Through polygons and Translucent polygons.

Setting	Meaning
0	Do not perform discard processing (default) <del>This results in operation that is similar to HOLLY1.</del>
1	Perform discard processing This improves drawing processing performance.

### pre-sort mode

This field specifies the Translucent polygon sort mode.

~~In HOLLY2,~~ this field is valid only when the region header type bit (bit 21) in the FPU\_PARAM\_CFG register is "0".

Setting	Meaning
0	Auto sort mode (default)
1	Pre-sort mode



### SDRAM\_REFRESH

Address : 0x005F 80A0

bit 31-8	7-0
Reserved	Refresh counter value

This field specifies the texture memory refresh counter value. (default = 0x20) Specify the number of clock cycles as the refresh time divided by 48. (The requested refresh time is 15.625μs.) For example, for 100MHz operation, the value is 15,625nsec/10nsec/48 = 32.6, so the setting in this register for the refresh counter value should be 0x20.

Set this register before releasing the reset condition through the SOFTRESET register.

### SDRAM\_ARB\_CFG

Address : 0x005F 80A4

bit 31-22	21-18	17-16	15-8	7-0
Reserved	Override value	Arbiter priority control	Arbiter crt page break latency count value	Arbiter page beak latency count value

This register contains the settings for the texture memory Arbiter. Normally, the priority ranking of each request is as follows:

- (1) CRT Controller
- (2) ISP parameters
- (3) ISP pointer data
- (4) ISP region data
- (5) TSP parameters
- (6) SH4 ports
- (7) Tile Accelerator pointers
- (8) Tile Accelerator ISP/TSP data
- (9) Texture normal data & VQ codebook
- (10) Texture VQ index
- (11) Render ports

#### Override value

When override mode is specified for the Arbiter, this field specifies the device that is given the highest priority.

Setting	Meaning
0x0	priority only (default)
0x1	Rendered data
0x2	Texture VQ index
0x3	Texture normal data & VQ codebook
0x4	Tile accelerator ISP/TSP data
0x5	Tile accelerator pointers
0x6	SH4
0x7	TSP parameters
0x8	TSP region data
0x9	ISP pointer data
0xA	ISP parameters
0xB	CRT controller
0xC-0xF	priority only

### **Arbiter priority control**

This field sets the Arbiter.

Setting	Meaning
0x0	Priority arbitration only (default)
0x1	The device specified in the "override value" field is given the highest priority.
0x2-0x3	When there is a device request with the same number as the round robin counter, that device is given the highest priority. If there is no such request, normal priority arbitration occurs.

### **Arbiter crt page break latency count value**

Setting this field forces a page break if there is a request from the CRT Controller immediately after the specified counter. (default = 0x00)

### **Arbiter page break latency count value**

Setting this field forces a page break if there is a request immediately after the specified counter. (default = 0x1F) Forcing a page break causes the Arbiter to function again, so that one type of access does not occupy memory.

## **SDRAM\_CFG**

Address : 0x005F 80A8

bit 31-29	28-0
Reserved	SDRAM Configuration

This register contains the settings for texture memory. (The default value ~~in HOLLY1 is 0x0DF28997, and in HOLLY2 is 0x15F28997.~~) Do not change these settings to other than the specified values.

Set this register before releasing the reset condition through the SOFTRESET register. Note that the value that is set depends on the HOLLY version.

#### **bit [28:26] = Read command to returned data delay**

Specify the number of clock cycles - 1. (The default value ~~in HOLLY1 is 0x3, and in HOLLY2 is 0x5.~~)

#### **bit [25:23] = CAS Latency value for mode register in SDRAM**

Specify the number of clock cycles. (default = 0x3) This value is fixed at 0x3 for 100MHz operation.

#### **bit [22:21] = Activate to Activate period**

Specify the number of clock cycles - 1. (default = 0x3)

#### **bit [20:18] = Read to Write period**

Specify the number of clock cycles - 1. (default = 0x4)

Normally, the number of clock cycles from read to write is the CAS latency + 2.

#### **bit [17:14] = Refresh to Activate period**

Specify the number of clock cycles - 2. (default = 0xA)

#### **bit [13:12] = Reserved**

"0" (zero) must be set.

#### **bit [11:10] = Pre-charge to Activate period**

Specify the number of clock cycles - 1. (default = 0x2)

#### **bit [9:6] = Activate to Pre-charge period**

Specify the number of clock cycles - 1. (default = 0x6)

#### **bit [5:4] = Activate to Read/Write command period**

Specify the number of clock cycles - 2. (default = 0x1) The minimum value is 0x1.

**bit [3:2] = Write to Pre-charge period**

Specify the number of clock cycles - 1. (default = 0x1)

**bit [1:0] = Read to Pre-charge period**

Specify the number of clock cycles - 1. (default = 0x3)

The following table shows the settings for HOLLY1.

Bit No	Contents of setting	Setting by SDRAM (NEC)	
		uPD4516161-10	uPD4516161A-10
<del>28:26</del>	<del>Read command to returned data delay</del>	<del>0x3</del>	<del>0x3</del>
<del>25:23</del>	<del>CAS Latency value mode register in SDRAM</del>	<del>0x3</del>	<del>0x3</del>
<del>22:21</del>	<del>Activate to Activate period</del>	<del>0x3</del>	<del>0x2</del>
<del>20:18</del>	<del>Read to Write period</del>	<del>0x4</del>	<del>0x4</del>
<del>17:14</del>	<del>Refresh to Activate period</del>	<del>0x8</del>	<del>0x6</del>
<del>11:10</del>	<del>Pre-charge to Activate period</del>	<del>0x2</del>	<del>0x2</del>
<del>9:6</del>	<del>Activate to Pre-charge period</del>	<del>0x6</del>	<del>0x5</del>
<del>5:4</del>	<del>Activate to Read/Write command period</del>	<del>0x1</del>	<del>0x1</del>
<del>3:2</del>	<del>Write to Pre-charge period</del>	<del>0x1</del>	<del>0x1</del>
<del>1:0</del>	<del>Read to Pre-charge period</del>	<del>0x3</del>	<del>0x3</del>
<del>31:0</del>	<del>SDRAM_CFG Register setting</del>	<del>0x0DF20997</del>	<del>0x0DD18957</del>

The following table shows the settings for HOLLY2.

Bit No	Contents of setting	Setting by SDRAM		
		NEC uPD4516161A-10 or NEC uPD4516161A-10B Rev.B,P	NEC uPD4516161-10	Samsung KM416S1020CT-L or NEC uPD4516161-A10 Rev.B,P
28:26	Read command to returned data delay	0x5	0x5	0x5
25:23	CAS Latency value mode register in SDRAM	0x3	0x3	0x3
22:21	Activate to Activate period	0x2	0x3	0x2
20:18	Read to Write period	0x4	0x4	0x4
17:14	Refresh to Activate period	0x7	0x8	0x5
13:12	Reserved	0x0	0x0	0x0
11:10	Pre-charge to Activate period	0x2	0x2	0x1
9:6	Activate to Pre-charge period	0x5	0x6	0x4
5:4	Activate to Read/Write command period	0x1	0x1	0x1
3:2	Write to Pre-charge period	0x0	0x0	0x0
1:0	Read to Pre-charge period	0x1	0x1	0x1
31:0	SDRAM_CFG Register setting value	0x15D1C951	0x15F20991	0x15D14511

---

**FOG\_COL\_RAM**

Address : 0x005F 80B0

bit 31-24	23-16	15-8	7-0
Reserved	Red	Green	Blue

This register specifies the Fog Color for Look-up Table mode. (default = 0x000000)

**FOG\_COL\_VERT**

Address : 0x005F 80B4

bit 31-24	23-16	15-8	7-0
Reserved	Red	Green	Blue

This register specifies the Fog Color for Per Vertex mode. (default = 0x000000)

**FOG\_DENSITY**

Address : 0x005F 80B8

bit 31-16	15-8	7-0
Reserved	Fog scale mantissa	Fog scale exponent

This register specifies the Fog scale value for Look-up Table mode.

***Fog scale mantissa***

This field specifies the mantissa where bit 15 is the 1.0 bit. (default = 0x00) For example, to specify 255.0 as the Fog scale value, specify 0xFF.

***Fog scale exponent***

This field specifies the exponent in two's complement format. (default = 0x00) For example, to specify 255.0 as the Fog scale value, specify 0x07.

**FOG\_CLAMP\_MAX**

Address : 0x005F 80BC

bit 31-24	23-16	15-8	7-0
Alpha	Red	Green	Blue

This register specifies the maximum value for color clamping. (default = 0x0000 0000)

**FOG\_CLAMP\_MIN**

Address : 0x005F 80C0

bit 31-24	23-16	15-8	7-0
Alpha	Red	Green	Blue

This register specifies the minimum value for color clamping. (default = 0x0000 0000)

---

**SPG\_TRIGGER\_POS (Read Only)**

Address : 0x005F 80C4

bit 31-26	25-16	15-10	9-0
Reserved	trigger v count	Reserved	trigger h count

This register indicates the HV counter value that was latched at the falling edge of the external trigger signal.

For details on the external trigger signal and the HV counter value, refer to section 5, "Peripheral Interface."

**SPG\_HBLANK\_INT**

Address : 0x005F 80C8

bit 31-26	25-16	15-14	13-12	11-10	9-0
Reserved	hblank_ in_interrupt	Reserved	hblank_ int_mod e	Reserved	line_comp_val

***hblank\_in\_interrupt***

This field specifies the horizontal position at which the H Blank interrupt is output. (default = 0x31D)

***hblank\_int\_mode***

This field specifies the H Blank interrupt mode.

Setting	Meaning
0x0	Output when the display line is the value indicated by line_comp_val. (default)
0x1	Output every line_comp_val lines.
0x2	Output every line.
0x3	Reserved

***line\_comp\_val***

This field specifies the value that is compared to the display line. (default = 0x000)

**SPG\_VBLANK\_INT**

Address : 0x005F 80CC

bit 31-26	25-16	15-10	9-0
Reserved	vblank out interrupt line number	Reserved	vblank in interrupt line number

***vblank out interrupt line number***

This field specifies the position at which the V Blank Out interrupt is output. (default = 0x150)

The recommended value is 0x015.

***vblank in interrupt line number***

This field specifies the position at which the V Blank In interrupt is output. (default = 0x104)

---

## SPG\_CONTROL

Address : 0x005F 80Do

bit 31-10	9	8	7	6	5	4	3	2	1	0
Reserved	csync_on_h	sync_direction	PAL	NTSC	force_field2	interlace	spg_lock	mcsync_pol	mvsync_pol	mhsync_pol

### ***csync\_on\_h***

This field specifies the sync signal that is output on the HSYNC pin.

Setting	Meaning
0	HSYNC (default)
1	CSYNC

### ***sync\_direction***

This field specifies the sync signal pin as either an input or an output.

Setting	Meaning
0	Input: Use an external sync signal. (default)
1	Output: Use the internal sync signal.

### ***PAL***

Specify "1" for PAL mode (default = 0). Specify "0" in VGA mode.

### ***NTSC***

Specify "1" for NTSC mode (default = 1). Specify "0" in VGA mode.

### ***force\_field2***

This field specifies whether or not to force display in field 2.

Setting	Meaning
0	Do not display in field 2. (default)
1	Display in field 2.

### ***interlace***

This field specifies whether to use interlacing or not.

Setting	Meaning
0	Non-interlace (default)
1	Interlace

### ***spg\_lock***

This field specifies whether to synchronize the internal circuitry with VSYNC input from an external source.

Setting	Meaning
0	During normal operation (default)
1	Set to '1' for only one frame upon extend synchronization.

### ***mcsync\_pol***

### ***mvsync\_pol***

### ***mhsync\_pol***

This field specifies the polarity of CSYNC, VSYNC, and HSYNC.

Setting	Meaning
0	active low (default)
1	active high

---

**SPG\_HBLANK**

Address : 0x005F 80D4

bit 31-26	25-16	15-10	9-0
Reserved	hbend	Reserved	hbstart

**hbend**

Specify the H Blank ending position. (default = 0x07E)

**hbstart**

Specify the H Blank starting position. (default = 0x345)

**SPG\_LOAD**

Address : 0x005F 80D8

bit 31-26	25-16	15-10	9-0
Reserved	vcount	Reserved	hcount

**vcount**

Specify "number of lines per field - 1" for the CRT; in interlace mode, specify "number of lines per field/2 - 1." (default = 0x106)

**hcount**

Specify "number of video clock cycles per line - 1" for the CRT. (default = 0x359)

**SPG\_VBLANK**

Address : 0x005F 80DC

bit 31-26	25-16	15-10	9-0
Reserved	vbend	Reserved	vbstart

**vbend**

Specify the V Blank ending position. (default = 0x150) The recommended value is 0x015.

**vbstart**

Specify the V Blank starting position. (default = 0x104)

**SPG\_WIDTH**

Address : 0x005F 80E0

31-22	21-12	11-8	7	6-0
eqwidth	bpwidth	vswidth	R	hswidth

**eqwidth**

Specify the equivalent pulse width as "number of video clock cycles - 1". (default = 0x01F)

**bpwidth**

Specify the broad pulse width as "number of video clock cycles - 1". (default = 0x319)

**vswidth**

Specify the VSYNC width in terms of the number of lines. (default = 0x3)

**hswidth**

Specify the HSYNC width as "number of video clock cycles - 1". (default = 0x3F)

---

---

**TEXT\_CONTROL**

Address : 0x005F 80E4

31-18	17	16	15-13	12-8	7-5	4-0
Reserved	Code book endian_re g	Index endian_re g	Reserved	bank bit	Reserved	stride

***Code book endian\_reg******Index endian\_reg***

This field makes the Endian specification for the code book and index.

Setting	Meaning
0	Little Endian (default)
1	Big Endian

***bank bit***

This field specifies the position of the bank bit when accessing texture memory (default = 0x00). Normally, set 0x00.

***stride***

This field specifies the U size of the stride texture. The U size is the stride value  $\times$  32.

Setting	Meaning
0x00	invalid (default)
0x01	32
0x02	64
0x03	96
0x04	128
. . .	. . . . .
. . .	
0x1C	896
0x1D	928
0x1E	960
0x1F	992



## VO\_CONTROL

Address : 0x005F 80E8

bit 31-22	21-16	15-9	8	7-4	3	2	1	0
Reserved	pclk_delay	Reserved	pixel_double	Field_mode	blank_video	blank_pol	vsync_pol	hsync_pol

This register contains the video output settings. (default = 0x00 0108)

### ***pclk\_delay***

This field specifies the delay for the PCLK signal to the DAC.

bit21 : Reset delay value

bit20~16 : Controls delay value

### ***pixel\_double***

This field specifies whether to output the same pixel or not for two pixels in the horizontal direction.

Setting	Meaning
0	not pixel double
1	pixel double (default)

### ***field\_mode***

This field specifies the video field control method.

Setting	Meaning
0x0	Use field flag from SPG. (default)
0x1	Use inverse of field flag from SPG.
0x2	Field 1 fixed.
0x3	Field 2 fixed.
0x4	Field 1 when the active edges of HSYNC and VSYNC match.
0x5	Field 2 when the active edges of HSYNC and VSYNC match.
0x6	Field 1 when HSYNC becomes active in the middle of the VSYNC active edge.
0x7	Field 2 when HSYNC becomes active in the middle of the VSYNC active edge.
0x8	Inverted at the active edge of VSYNC.
0x9-0xF	Reserved

### ***blank\_video***

This field specifies whether to display the screen or not.

Setting	Meaning
0	Display the screen.
1	Do not display the screen. (Display the border color.) (default)

### ***blank\_pol***

### ***vsync\_pol***

### ***hsync\_pol***

This field specifies the polarity of BLANK, VSYNC, and HSYNC.

Setting	Meaning
0	active low (default)
1	active high

### VO\_STARTX

Address : 0x005F 80EC

bit 31-10	9-0
Reserved	Horizontal start position

This register specifies the display starting position in the horizontal direction for HSYNC.  
(default = 0x09D)

### VO\_STARTY

Address : 0x005F 80F0

bit 31-26	25-16	15-10	9-0
Reserved	Vertical start position on field 2	Reserved	Vertical start position on field 1

This register specifies the display start position in the vertical direction for field-1/field-2  
versus VSYNC. (The default for both is 0x015.)

### SCALER\_CTL

Address : 0x005F 80F4

bit 31-19	18	17	16	15-0
Reserved	Field Select	Interlace	Horizontal scaling enable	Vertical scale factor

#### **Field Select**

This register specifies the field that is to be stored in the frame buffer in flicker-free  
interlace mode type B.

Setting	Meaning
0	field 1 (default)
1	field 2

#### **Interlace**

This register specifies whether or not to use flicker-free interlace mode type B.

Setting	Meaning
0	off (default)
1	on

#### **Horizontal scaling enable**

This field specifies whether or not to use the horizontal direction 1/2 scaler.

Setting	Meaning
0	disable (default)
1	enable

#### **Vertical scale factor**

This field specifies the scale factor in the vertical direction. (default = 0x0400)

This value consists of a 6-bit integer portion and a 10-bit decimal portion, and expands or  
reduces the screen in the vertical direction by "1/scale factor." When using flicker-free  
interlace mode type B, specify 0x0800.

<Example of setting>

× 2:      0x0200  
× 1:      0x0400  
× 0.5:    0x0800

### PAL\_RAM\_CTRL

Address : 0x005F 8108

bit 31-2	1-0
Reserved	pixel format

This register specifies the palette color format. This register must be set before storing any data in palette RAM.

Setting	Meaning
0x0	ARGB1555 (default)
0x1	RGB565
0x2	ARGB4444
0x3	ARGB8888

#### SPG\_STATUS (Read Only)

Address : 0x005F 810C

bit 31-14	13	12	11	10	9-0
Reserved	vsync	hsync	blank	fieldnum	scanline

This register indicates the current status of the internal sync circuit (SPG). (Value after reset = 0x0000)

##### ***vsync***

This field indicates the status of the VSYNC signal.

##### ***hsync***

This field indicates the status of the HSYNC signal.

##### ***blank***

This field indicates the status of the BLANK signal.

##### ***fieldnum***

This field indicates the field number.

Setting	Meaning
0	field 1
1	field 2

##### ***scanline***

This field indicates the display line number.

#### FB\_BURSTCTRL

Address : 0x005F 8110

bit 31-20	19-16	14-8	7-6	5-0
Reserved	wr_burst	vid_lat	Reserved	vid_burst

##### ***wr\_burst***

Specify the frame buffer burst write size - 1. (default = 0x09)

##### ***vid\_lat***

This field specifies the amount of data remaining in the read data FIFO when making a frame buffer read request. (default = 0x06) Set a value such that (vid\_lat) < 0x80 - (vid\_burst). The recommended value is 0x3F.

##### ***vid\_burst***

This field specifies the burst read size from the frame buffer. (default = 0x39)

### FB\_C\_SOF (Read Only)

Address : 0x005F 8114

bit 31-24	23-2	1-0
Reserved	Frame Buffer Current Read Address	Reserved

Specify the starting address, in 32-bit units, for the frame that is currently being sent to the DAC. (default = 0x000000)

### Y\_COEFF

Address : 0x005F 8118

bit 31-16	15-8	7-0
Reserved	Coefficient 1	Coefficient 0/2

Scaling in the vertical direction is filtered with a three-line buffer. This register specifies an unsigned 8-bit value as the filtering co-efficient for each line when scaling down.

Coefficient 0/2: Coefficient for line 0/2 (default = 0x00)

Coefficient 1: Coefficient for line 1 (center) (default = 0x00)

<Normal setting example>

Coefficient 0/2 = 0x40 (Coefficient:  $\times 0.25$ )

Coefficient 1 = 0x80 (Coefficient:  $\times 0.5$ )

### PT\_ALPHA\_REF

Address : 0x005F 811C

bit 31-8	7-0
Reserved	Alpha reference for punch through

~~<Additional register in HOLLY>~~

This register specifies the alpha value that is used for comparison when drawing Punch Through polygons. (default = 0xFF) Only those pixels for which [(pixel  $\alpha$  value)  $\geq$  (register setting)] is true are drawn.

### FOG\_TABLE

Address : 0x005F 8200 ~ 0x005F 83FD

bit 31-16	15-0
Reserved	Fog table data

This register specifies the Fog data (128 tables) for Look-up Table mode.

### PALETTE\_RAM

Address : 0x005F 9000 ~ 0x005F 9FFF

bit 31-0
Palette data

This register specifies the color data (1024 colors) for the palette texture. The color format is specified by the PAL\_RAM\_CTRL register. In the case of the 16-bit color format, only the lower 16 bits (bits 15 through 0) are valid. The PAL\_RAM\_CTRL register must be set before any color data is set.

### § 8.4.3 Tile Accelerator Registers

#### TA\_OL\_BASE

Address : 0x005F 8124

bit 31-24	23-5	4-0
Reserved	Base Address	0 0000

This register specifies (in  $8 \times 32$ -bit units) the starting address for storing Object Lists as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x00000)

#### TA\_ISP\_BASE

Address : 0x005F 8128

bit 31-24	23-2	1-0
Reserved	Base Address	00

This register specifies (in 32-bit units) the starting address for storing the ISP/TSP Parameters as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x000000)

#### TA\_OL\_LIMIT

Address : 0x005F 812C

bit 31-24	23-5	4-0
Reserved	Limit Address	0 0000

This register specifies (in  $8 \times 32$ -bit units) the limit address for storing Object Lists as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x00000) Because the TA may automatically store data in the address that is specified by this register, it must not be used for other data. For example, the address specified here must not be the same as the address in the TA\_ISP\_BASE register.

If the Object List storage address exceeds this address, the data is not stored and an interrupt is generated. Because the Object List will not be stored as a data structure correctly when this interrupt is generated, the Object List can be used for drawing, but will not produce the expected image.

#### TA\_ISP\_LIMIT

Address : 0x005F 8130

bit 31-24	23-2	1-0
Reserved	Limit Address	00

This register specifies (in 32-bit units) the limit address for storing ISP/TSP Parameters as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x00000) If the ISP/TSP Parameter storage address exceeds this address, an interrupt is generated and the Object List is not stored. Because the ISP/TSP Parameters are not stored correctly when this interrupt is generated, the parameters cannot be used for drawing.

### TA\_NEXT\_OPB (Read Only)

Address : 0x005F 8134

bit 31-24	23-5	4-0
Reserved	Address	0 0000

This register indicates (in  $8 \times 32$ -bit units) the starting address for the Object Pointer Block that the TA will use next as a relative address, assuming the start of texture memory (32-bit area) as "0." This address is not finalized until it is initialized by the TA\_LIST\_INIT register.

~~In HOLLY1, when this register is initialized by the TA\_LIST\_INIT register, the following start address is set:~~

~~OPB\_Mode = 0 (TA\_ALLOC\_CTRL)  
Start address = TA\_OL\_BASE  
+ ([OPB ( = Object Pointer Block ) size of Opaque]  
+ [OPB size of Opaque Modifier Volume]  
+ [OPB size of Translucent]  
+ [OPB size of Translucent Modifier Volume] )  
\* ([Tile\_X\_Num of TA\_GLOB\_TILE\_CLIP] + 1)  
\* ([Tile\_Y\_Num of TA\_GLOB\_TILE\_CLIP] + 1)  
\* 4~~

~~OPB\_Mode = 1 (TA\_ALLOC\_CTRL)  
Start address = TA\_OL\_BASE~~

~~In HOLLY2, When this register is initialized by the TA\_LIST\_INIT register, the value in the TA\_NEXT\_OPB\_INIT register is loaded into this register. This register is not initialized by the TA\_LIST\_CONT register.~~

### TA\_ITP\_CURRENT (Read Only)

Address : 0x005F 8138

bit 31-24	23-2	1-0
Reserved	Address	00

This register specifies (in 32-bit units) the starting address where the next ISP/TSP Parameters are stored as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x00 0000)

~~In HOLLY2, when this register is initialized by the TA\_LIST\_INIT register, the value in the TA\_ISP\_BASE register is loaded into this register. This register is not initialized by the TA\_LIST\_CONT register.~~

### TA\_GLOB\_TILE\_CLIP

Address : 0x005F 813C

bit 31-20	19-16	15-6	5-0
Reserved	Tile_Y_Num	Reserved	Tile_X_Num

This register specifies the Global Tile Clip values. Only those objects that correspond to Tiles in the Global Tile Clipping area are stored in texture memory. This register must be set before the list is initialized by the TA\_LIST\_INIT register.

#### Tile\_Y\_Num

This field specifies the Tile number in the Y direction (0 to 14) for the lower right corner of the Global Tile Clip. (default = 0x0) Set [the number of Tiles in the Y direction in the valid area] - 1. "15" (0xF) must not be specified.

#### Tile\_X\_Num

This field specifies the Tile number in the X direction (0 to 39) for the lower right corner of the Global Tile Clip. (default = 0x00) Set [the number of Tiles in the X direction in the valid area] - 1. "40" (0x28) through "63" (0x3F) must not be specified.

### TA\_ALLOC\_CTRL

Address : 0x005F 8140

bit 31-17	20	19-18	17-16	15-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0
R	OPB_Mode	R	PT_OPB	R	TM_OPB	R	T_OPB	R	OM_OPB	R	O_OPB

### **OPB\_Mode**

This field specifies the address direction when storing the next Object Pointer Block (OPB) in texture memory, in the event that the specified Object Pointer Block size has been exceeded.

~~\*The position differs according to the HOLLY version; bit 16 in HOLLY1 and bit 20 in HOLLY2.~~

Setting	OPB storage method
0	Store in the direction of increasing addresses (default)
1	Store in the direction of decreasing addresses

### **PT\_OPB**

This field specifies the unit size for the Object Pointer Block of the Punch Through list ~~in HOLLY2.~~

### **TM\_OPB**

This field specifies the unit size of an Object Pointer Block for a Translucent Modifier Volume list.

### **T\_OPB**

This field specifies the unit size of an Object Pointer Block for a Translucent list.

### **OM\_OPB**

This field specifies the unit size of an Object Pointer Block for an Opaque Modifier Volume list.

### **O\_OPB**

This field specifies the unit size of an Object Pointer Block for an Opaque list.

These fields specify the Object Pointer Block unit size for each type of list (Opaque, etc.) Specify "No List" for a list that is not used in the screen. For the Pointer Burst Size value in the FPU\_PARAM\_CFG register, set a value that is less than or equal to the Object Pointer Block size specified here.

This register must be set before the lists are initialized through the *TA\_LIST\_INIT* register.

Setting	Unit size
0	No List (Not used in screen)
1	8 × 32bit
2	16 × 32bit
3	32 × 32bit

### **TA\_LIST\_INIT**

Address : 0x005F 8144

bit 31	30-0
List_Init	Reserved

Setting the List\_Init bit to "1" initializes the lists. This bit must be set before the lists are created by the TA. This bit always returns a "0" when read.

Before initializing the lists through this register, the *TA\_GLOB\_TILE\_CLIP* register, the *TA\_ALLOC\_CTRL* register, and, ~~in HOLLY2,~~ the *TA\_NEXT\_OPB\_INIT* register must be set.

### **TA\_YUV\_TEX\_BASE**

Address : 0x005F 8148

bit 31-24	23-3	2-0
Reserved	Base Address	000

This register specifies (in 64-bit units) the starting address for storing YUV422-Texture data as a relative address, assuming the start of texture memory (64-bit area) as "0." (default = 0x00 0000) When this register is written, the YUV-data Converter in the TA is initialized, and then begins operation using the next data that is input as U-data.

## TA\_YUV\_TEX\_CTRL

Address : 0x005F 814C

bit 31-25	24	23-17	16	15-14	13-8	7-6	5-0
Reserved	YUV_Form	R	YUV_Tex	R	YUV_V_Size	R	YUV_U_Size

### YUV\_Form

This field specifies the format of the YUV data that is input to the TA.

Setting	YUV data format
0	YUV420 format (default)
1	YUV422 format

### YUV\_Tex

This field selects the type of YUV422-Texture that is stored in texture memory.

Setting	Texture type
0	One texture of [(YUV_U_Size + 1) * 16] pixels (H) × [(YUV_V_Size + 1) * 16] pixels (V)
1	[(YUV_U_Size + 1) * (YUV_V_Size + 1)] textures of 16 texels (H) × 16 texels (V)

### YUV\_V\_Size

This field specifies the vertical size of the YUV422-Textures that are stored in texture memory. (default = 0x00) Specify "the number of pixels in the vertical direction/16 - 1."

<Non-Twiddled texture>

Texture V size	16	32	64	128	256	512	1024
YUV_V_Size setting	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F

### YUV\_U\_Size

This field specifies the horizontal size of the YUV422 textures that are stored in texture memory. (default = 0x00) Specify "the number of pixels in the horizontal direction/16 - 1." Based on the texture sizes that the CORE Block supports, the following values can be specified:

<Non-Twiddled texture>

Texture U size	16	32	64	128	256	512	1024
YUV_U_Size setting	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F

<Non-Twiddled Stride texture>

Texture U size	32	64	96 ~ 960 (a multiple of 32)	992	1024
YUV_U_Size setting	0x01	0x03	0x05 ~ 0x3B (an odd number)	0x3D	0x3F



### TA\_YUV\_TEX\_CNT (Read Only)

Address : 0x005F 8150

bit 31-13	12-0
Reserved	YUV_Num

This register indicates the number of macroblocks (16 pixels × 16 pixels) that are currently stored in texture memory. (default = 0x0000) If the TA\_YUV\_TEX\_BASE register is written, this register is initialized to "0."

### TA\_LIST\_CONT

Address: 0x005F 8160

bit 31	30-0
List_Cont	Reserved

~~◀Additional register in HOLLY2▶~~

If the List\_Cont bit is set to "1", list continuation processing is performed. Although the TA is initialized, just as with the TA\_LIST\_INIT register, the values in the TA\_NEXT\_OPB register and in the TA\_ITP\_CURRENT register are not initialized. As a result, when the second and subsequent lists are input on a continued basis, the Object List and ISP/TSP Parameters are stored after the previous parameters. If this bit is read, it returns a "0".

The value of the TA\_OL\_BASE register must be changed before performing list continuation processing.

### TA\_NEXT\_OPB\_INIT

Address : 0x005F 8164

bit 31-24	23-5	4-0
Reserved	Address	0 0000

~~◀Additional register in HOLLY2▶~~

This register indicates (in 32-bit units) the address for storing additional OPBs during list initialization as a relative address, assuming the start of texture memory (32-bit area) as "0." (default = 0x0 0000) When setting this register, it is necessary to consider the total number of OPBs for which area will need to be allocated in texture memory for the entire list that is being input (in several pieces) to the TA.

This register must be set before initializing lists through the TA\_LIST\_INIT register.

### TA\_OL\_POINTERS (Read Only)

Address : 0x005F 8600 ~ 0x005F 8F5C

bit 31	30	29	28-25	24	23-2	1-0
Entry	Sprite flag	Triangle flag	Number of Triangles/Quad s	Shadow	Pointer Address	Skip[1:0]

There are enough of these registers for 600 Tiles: 40 Tiles in the X direction × 15 Tiles in the Y direction. These registers cannot be accessed while the TA is in operation.

#### **Entry**

This field indicates whether the current object type has been registered at least once in the Tile in question. If the lists are initialized through the TA\_LIST\_INIT register or the End Of List Control Parameter is input, this field is cleared to "0."

#### **Sprite flag**

This field indicates that the previous polygon that was registered in the Tile in question was a Quad polygon.

#### **Triangle flag**

This field indicates that the previous polygon that was registered in the Tile in question was a Triangle polygon.

#### **Number of Triangle/Quad**

This field indicates the number of consecutive previous polygons that were registered in the Tile in question.

**Shadow**

This field indicates the shadow value for the previous polygon that was registered in the Tile in question.

**Pointer Address**

This field indicates the storage address of the next Object List for the Tile in question. The address that is indicated is a 32-bit aligned relative address, assuming the start of texture memory (32-bit area) as "0."

**Skip[1:0]**

This field indicates the lower 2 bits of the skip value for the previous polygon that was registered in the Tile in question.

## § 8.4.4 GD-ROM Registers

The GD-ROM device is positioned as an ATA device, and the registers are designed accordingly. Note that in some cases, different registers are indicated for reading as opposed to writing.

*(The Control Block Registers are described below.)*

**Alternate Status(Read) / Device Control(Write)**

Address : 0x005F 7018

bit 31-8	7-0
Reserved	Alternate Status / Device Control

- Alternate Status (Read)

The contents of this register are identical to those of the 0x005F 709C status register; refer to that description for details on the function of each bit. Note also that interrupt and DMA status information is not cleared even if this register is read.

7	6	5	4	3	2	1	0
BSY	DRDY	DF	DSC	DRQ	CORR	Reserved	CHECK

- Device Control (Write)

7	6	5	4	3	2	1	0
Reserved				1	SRST	nIEN	0

**SRST**

This is the bit that the SH4 (i.e., the "system" or the "host") sets in order to initiate a software reset. This protocol is not used, however. To initiate a software reset, use the software reset defined by ATAPI.

**nIEN**

This bit sets interrupts to the host. When "0," the interrupt is enabled; when "1," the interrupt is disabled.

***(The Command Block Registers are described below.)***

**Data (Read/Write)**

Address : 0x005F 7080

bit 31-16	15-0
Reserved	Data / Data

- Data (Read/Write)  
This register is used for data transfers with the host, and can switch between 8 bits and 16 bits.

**Error(Read) / Features(Write)**

Address : 0x005F 7084

bit 31-8	7-0
Reserved	Error / Features

- Error (Read)  
This register can be used to read the end status of the command that was executed last. This register is also set when device diagnostics are terminated. If bit 0 of the status register is "1," it indicates that an error occurred. In that event, the details of the error are reflected in this register.

7	6	5	4	3	2	1	0
Sense Key				MCR	ABRT	EOMF	ILI

***Sense Key***

The contents of this field are explained below.

Code	Meaning
0	NO SENSE. This sense key code indicates that there is no specific sense key information that should be reported. This sense key code is also used when the command was executed successfully.
1	RECOVERED ERROR. This sense key code indicates that the last command was executed successfully after some error recovery processing by the device. Further details can be found by checking the supplemental sense byte and information field. If multiple error recoveries occurred during the execution of one command, this device reports the error that was recovered from last.
2	NOT READY. This sense key code indicates that this device cannot be accessed.
3	MEDIUM ERROR. This sense key code indicates that the command terminated with a nonrecoverable error due to a defect on the recording medium or an error that occurred during recording or reading. This sense key code is also returned when this device cannot determine whether the problem was a medium defect or a hardware error (sense key code 4).
4	HARDWARE ERROR. This sense key code indicates that a nonrecoverable hardware error (for example, a controller failure, a device failure, a parity error, etc.) occurred while this device was executing a command or running self-diagnostics.
5	ILLEGAL REQUEST. This sense key code indicates either that there was an illegal parameter in a command packet, or that there was an illegal parameter in additional parameters that were added as data for a command. When this device detects an illegal parameter in a command packet, it terminates the command without making any changes to the medium. When this device detects an illegal parameter in additional parameters that were added as data for a command, it is possible that the device has already made changes to the medium. When this sense key code is reported, the command has not yet been executed.
6	UNIT ATTENTION. This sense key code indicates either that a removable media has been switched, or that this device was reset.
7	DATA PROTECT. This sense key code indicates that an attempt was made to write to a block that is write-protected.
8-0xA	Reserved
0xB	ABORTED COMMAND. This sense key code indicates that the device aborted the command. Recovery may be possible by re-executing the command from the host system.
0xC-0xF	Reserved

***MCR***

This field indicates that there was a media change request and the media was ejected (ATA level).

***ABRT***

This field indicates that the command was invalidated because the drive is not ready (ATA level).

***EOMF***

This field indicates that the end of the media was detected (option).

***ILI***

This field indicates that the command has an illegal length (option).

- Features (Write)

This register is normally used to specify the data transfer method, but is also used to specify the Set Features parameters among the SATA commands (commands that correspond to the ATA command within the protocol).

When using this register to specify the data transfer method

7	6	5	4	3	2	1	0
Reserved							DMA

***DMA***

This field indicates that the transfer of the data to a command is to be performed in DMA mode.

When using this register to the parameter of Set Features command.

7	6	5	4	3	2	1	0
Set(1)/ Clear(0) Feature	Feature Number(= "3")						

***Feature Number***

This field is the transfer mode setting. The transfer mode that was set in the Sector Count register can be set by writing a "3" in Feature Number and then receiving the Set Feature command.

The actual transfer mode is specified by using the *Sector Count* register.

**Interrupt reason(Read) / Sector Count(Write)**

Address : 0x005F 7088

bit 31-8	7-0
Reserved	Interrupt reason / Sector Count

- Interrupt reason (Read)

7	6	5	4	3	2	1	0
Reserved						IO	CoD

**IO**

When "0," this field indicates the direction of transfer is from the host to the device; when "1," this field indicates the direction of transfer is from the device to the host.

**CoD**

When "0," this field indicates data; when "1," this field indicates a command.

IO	DRQ	CoD	Meaning
0	1	1	Ready to receive command packet.
1	1	1	Ready to send message from device to host.
1	1	0	Ready to send data to the host.
0	1	0	Ready to receive data from the host.
1	0	1	The "completed" status is in the status register.

- Sector Count (Write)

7	6	5	4	3	2	1	0
Transfer Mode					Mode value		

Transfer mode according to the sector count register value

Register value	Transfer mode
00000 00x	PIO Default Transfer Mode
00001 xxx	PIO Flow Control Transfer Mode x
00010 xxx	Single Word DMA Mode x
00100 xxx	Multi-Word DMA
00011 xxx	Reserved(for Pseudo DMA Mode)

This register is used in combination with the Set Features command, a SATA command.

### Sector Number

(Read/Write)

Address : 0x005F 708C

bit 31-8	7-0
Reserved	Sector Number / Sector Number

- Sector Number (Read/Write)

The information that is obtained in this register is identical to the value of the REQ\_STAT command. For details, refer to the explanation of REQ\_STAT. The operation of this register does not conform with the ATA standard.

7	6	5	4	3	2	1	0
Disc Format				Status			

### Byte Count Low (Read/Write)

Address : 0x005F 7090

bit 31-8	7-0
Reserved	Byte Count LSB / Byte Count LSB

### Byte Count High (Read/Write)

Address : 0x005F 7094

bit 31-8	7-0
Reserved	Byte Count MSB / Byte Count MSB

These two registers (*Byte Count Low* and *Byte Count High*) are used to control the number of bytes that the host sends in response to each DRQ.

These register show the LSB (*Byte Count Low*) and the MSB (*Byte Count High*), respectively, for the byte count.

These registers are used in PIO transfer mode only. In DMA mode, this byte count is ignored. This count is set before the packet command is issued. This count stipulates the total transfer length for commands that transfer data groups (MODE SELECT/SENSE, INQUIRY, etc.).

For commands that request multiple DRQ interrupts, such as read and write instructions, the expected transfer length is set in this count.

Whenever data is transferred, this device sets the number of data bytes that the host transfers in this byte count, and then generates a DRQ interrupt. The contents of this register do not change while DRQ is "1."

### Drive Select (Read/Write)

Address : 0x005F 7098

bit 31-8	7-0
Reserved	Drive Select / Drive Select

- Drive Sector (Read/Write)

7	6	5	4	3	2	1	0
1	Reserved	1	0	LUN			

#### LUN

This field specifies the logical unit that executes the command.  
This parameter is optional, and is reserved for future use.

---

---

**Status(Read) / Command (Write)**

Address : 0x005F 709C

bit 31-8	7-0
Reserved	Drive Select / Drive Select

- **Status (Read)**

This register indicates the drive status. If this register is read, the interrupt signal that was pending is cleared.

When bit 7 (BSY) is "0," the other bits are also valid, and access to the command block is possible. When bit 7 (BSY) is "1," the other bits are invalid, and access to the command block is not possible.

Bit 7 (BSY) becomes valid 400nsec after the command is accepted.

7	6	5	4	3	2	1	0
BSY	DRDY	DF	DSC	DRQ	CORR	Reserved	CHECK

**BSY**

This field is set to "1" when a command is accepted.

**DRDY**

This field is set to "1" when response to an ATA command is possible.

**DF**

This field returns the Drive Fault information.

**DSC**

This field indicates that seek processing is complete.

**DRQ**

This field is set to "1" when data transfer with the host is possible.

**CORR**

This field indicates that a correctable error occurred.

**CHECK**

If an error occurs, this bit is set to "1."

- **Command (Write)**

The host sets commands in this register. The command is loaded into the appropriate register in the command block along with the necessary parameters, and becomes valid when the command code is written to the *Command* register (0x005F 70C9).

When the GD-ROM device receives a command, it sets BSY within 400nsec.

The following commands that are part of the ATA standard specifications are supported by this system.

Command	Code
NOP	0x00
Soft Reset	0x08
Execute Device Diagnostic	0x90
Packet Command	0xA0
Identify Device	0xA1
Set Features	0xEF

The contents of the commands are described below.

***NOP (0x00)***

This command enables access to the device status for hosts for which only 16-bit register access is valid. The device executes this command as a response to commands that are not recognized by doing the following:

- Setting "abort" in the error register
- Setting "error" in the status register
- Clearing BUSY in the status register
- Asserting the INTRQ signal

***Soft Reset (0x08)***

This command executes a software reset.

If the GD-ROM device receives a "Soft Reset" command, it initializes the hardware and sets the default parameters. When the device is stopped, the disc motor begins to rotate and the device becomes ready to operate.

***Execute Drive Diagnostic (0x90)***

This command executes the device's internal diagnostics.

- (a) The device reports the results of its own diagnostics.
- (b) The device clears the BSY bit and initiates an interrupt.

The diagnostics codes that are written to the error register are 8-bit codes such as those shown in the table below.

Error Code	Meaning
0x00	Normal
0x03	Data buffer error
0x04	ODC error
0x05	CPU error
0x06	DSC error
0x07	Other error

***Packet Command (0xA0)***

For details, refer to the GD-ROM Protocol SPI (Sega Packet Interface) Specifications.

***Identify Device (0xA1)***

This command requests information on the drive (device) that is connected.

The host can get information from the device by using the ATAPI IDENTIFY DEVICE command.

Byte	Meaning
0x00	Manufacturer's ID
0x01	Model ID
0x02	Version ID
0x03~0x0F	Reserved
0x10~0x1F	Manufacturer's name (16 ASCII characters)
0x20~0x2F	Model name (16 ASCII characters)
0x30~0x3F	Firmware version (16 ASCII characters)
0x40~0x4F	Reserved

***Set Features (0xEF)***

This command makes settings concerning the timing and protocol for the interface with the device. A device can only make settings that concern the transfer mode.

1. Set "3" in the *Feature* register Set bit and the Feature Number.
2. Specify the transfer method in the upper five bits of the *Sector Count* register, and the mode number in the lower three bits.
3. Issue the Set Features command.

These settings can be made independently for DMA mode and PIO mode.

## § 8.4.5 AICA Register

The contents and function of each register for the AICA audio chip are explained below.



The addresses (in the "ADDRESS" column in the table that follows and in the descriptions) are the same for accesses that are internal and external to the AICA. In addition, register accesses by the SH4 are 4-byte accesses only, and only the lower 16 bits are valid.

### • Channel Data

AICA ADDR.	G2 ADDR.	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
0x00800000 0	0x00700000	KX	KB	--			SS	LP	PCMS		SA[22:16]							KX:KYONEX
0x00800004 4	0x00700004	SA[15:0]																LP:LPCTL
0x00800008 8	0x00700008	LSA[15:0]																KB:KYONB
0x0080000C C	0x0070000C	LEA[15:0]																SS:SSCTL
0x00800010	0x00700010	D2R[4:0]					D1R[4:0]					--	AR[4:0]					
0x00800014	0x00700014	--	LS	KRS[3:0]				DL[4:0]				RR[4:0]				LS:LPSLNK		
0x00800018	0x00700018	--	OCT[3:0]				--	FNS[9:0]										
0x0080001C	0x0070001C	RE	LFOF[4:0]					PLFOWS		PLFOS[2:0]		ALFOWS		ALFOS[2:0]		RE:LFORE		
0x00800020 0	0x00700020	--								IMXL[3:0]			ISEL[3:0]					
0x00800024	0x00700024	--				DISDL[3:0]				--			DIPAN[4:0]					
0x00800028 8	0x00700028	TL[7:0]								--			Q[4:0]					
0x0080002C C	0x0070002C	--			FLVo[12:0]													
0x00800030 0	0x00700030	--			FLV1[12:0]													
0x00800034	0x00700034	--			FLV2[12:0]													
0x00800038 8	0x00700038	--			FLV3[12:0]													
0x0080003C C	0x0070003C	--			FLV4[12:0]													
0x00800040 0	0x00700040	--			FAR[4:0]					--			FD1R[4:0]					
0x00800044	0x00700044	--			FD2R[4:0]					--			FRR[4:0]					
0x00800080 0   0x008000C4 4	0x00700080   0x007000C4	SLOT 1 CONTROL REGISTER																
: :																		
0x00801F80   0x00801FC4	0x00701F80   0x00701FC4	SLOT 63 CONTROL REGISTER																
0x00802000 0	0x00702000	--				EFSDL[3:0]					--			EFPAN[4:0]				DSP_OUT_1

---

---

:	:		:			
0x00802044	0x00702044	--	EFSDL[3:0]	--	EFPAN[4:0]	DSP_OUT_18

Table 8-21 Channel Data

### Register Descriptions (Channel Data)

The various registers that comprise the channel data are described below. (All registers are read/write registers, unless indicated otherwise.)

#### **KYONEX**(-/w)

Writing "1" to this register executes KEY\_ON, OFF for all slots. Writing "0" is invalid.

#### **KYONB**

This register registers KEY\_ON, OFF.

(If KEY\_ON is to be registered simultaneously, set this bit to "1" for all slots to be turned ON, and then write a "1" to **KEYONEX** for one of the slots.)

#### **SSCTL**

- 0: Use the data in external memory (SDRAM) as sound input data.
- 1: Use noise as sound input data.

#### **LPCTL**

- 0: Loop OFF (The **LSA** and **LEA** settings are required; once LEA is reached, processing ends.)
- 1: Forward loop.

#### **PCMS**[1:0]

(Cannot be changed during ADPCM playback.)

- 0: 16-bit PCM (two's complement format)
- 1: 8-bit PCM (two's complement format)
- 2: 4-bit ADPCM (Yamaha format)
- 3: 4-bit ADPCM long stream

#### **SA**[22:0]

This register specifies the starting address for the sound data in terms of the byte address. However,

when **PCMS** = 0, the LSB of SA must be "0."

**PCMS** = "2" or "3", LSB two bits of SA must be "00".

#### **LSA**[15:0]

This register specifies the loop starting address for the sound data in terms of the number of samples from SA.

The number of samples indicates the number of bytes in 8-bit PCM, the number of pairs of bytes (16 bits) in the case of 16-bit PCM, and the number of half-bytes in the case of ADPCM. The minimum values that can be set are limited by the pitch and the loop mode. Because the actual value is not approximated at values near SA due to the specifications for ADPCM, as large a value as possible must be used for **LSA** ( $LSA > 0x8$ ). (When in a loop) When using long stream, the lowest two bits of LSA must be "00".

#### **LEA**[15:0]

This register specifies the loop ending address for the sound data in terms of the number of samples from SA.

The minimum value that can be set is limited by the pitch and the loop mode.

Specify so that  $SA \leq LSA \leq LEA$ . When using long stream, the lowest two bits of LEA must be "00".

Refer to section 8.1.1.1, "Loop Control."

#### **AR**[4:0]

This register specifies the rate of change in the EG in the attack state. (The volume increases.)

#### **D1R**[4:0]

This register specifies the rate of change in the EG in the decay 1 state. (The volume decreases.)

#### **D2R**[4:0]

This register specifies the rate of change in the EG in the decay 2 state. (The volume decreases.)

#### **RR**[4:0]

This register specifies the rate of change in the EG in the release state. (The volume decreases.)

#### ***DL[4:0]***

This register specifies the EG level at which the transition is made from decay 1 to decay 2, making the specification through the upper 5 bits of the EG code.

#### ***KRS[3:0]***

This register specifies the EG key rate scaling rate (as a positive number).

0x0: Minimum scaling  
:  
0xE: Maximum scaling  
0xF: Scaling off

#### ***LPSLNK***

Loop start link function: when the sound slot input data address that is read exceeds the loop start address, the EG makes the transition to decay 1.

(When EG = 000, the transition is not made.) In this case, the transition to decay 2 may not be made, depending on the DL setting.

(Refer to section 8.1.1.3, "AEG.")

#### ***OCT[3:0]***

This register specifies the octave in two's complement format. The values that appear in parentheses in the table below could generate noise in the ADPCM, so they should be used with caution. (A maximum of "2" (when *FNS* = 0) is valid.)

OCT	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
Interval	-8	-7	-6	-5	-4	-3	-2	-1	0	+1	(+2)	(+3)	(+4)	(+5)	(+6)	(+7)

Table 8-22 Octave Specification

#### ***FNS[9:0]***

The pitch is set along with OCT by setting the F number.

Pitch:  $P[\text{CENT}] = 1200 \times \log_2((2^{10} + FNS)/2^{10})$

When *FNS* = 0 (and *OCT* = 0), the interval matches the sampling source. The pitch error (pitch precision) that is equivalent to the LSB of the *FNS* is 1.69.

(Refer to section 8.1.1.4 "PG.")

#### ***LFORE***

This register specifies whether or not to put the LFO into the initial state. (If noise was selected, the setting is invalid.)

0: Do not put the LFO in the reset state.  
1: Put the LFO in the reset state.

#### ***LFOF[4:0]***

This register specifies the LFO oscillating frequency. (If noise was selected, the setting is invalid.)

#### ***ALFOWS[1:0]***

This register specifies the shape of the ALFO waveform.

#### ***PLFOWS[1:0]***

This register specifies the shape of the PLFO waveform.

#### ***ALFOS[2:0]***

This register specifies the degree of mixing of the LFO to the EG.

#### ***PLFOS[2:0]***

This register specifies the degree of the LFO on the pitch.  
(Refer to section 8.1.1.5, "LFO.")

#### ***ISEL[3:0]***

This register specifies the *MIXS* register address for each slot when inputting sound slot output data to the DSP's *MIXS* register.

(Supplement) *MIXS* determines the sum of the inputs for all slots and handles the result as the DSP input. *MIXS* has an area for adding the input on each slot, and an area for storing the interval and value of one sample. These areas are allocated in alternation. As a result, reads on the DSP side are possible at

(Caution) any step.  
Make the settings so that the sum of the inputs to the *MIXS* does not exceed 0dB. (There is no overflow protect function.)

***TL[7:0]***

Total level: This register specifies the actual attenuation, which is derived by multiplying the EG value by this value which indicates the attenuation.

***DIPAN[4:0]***

This register specifies the orientation for each slot when sending direct data.

***EFPAN[4:0]***

This register specifies the orientation for each slot of effect data and external input data.

***IMXL[3:0]***

This register specifies the level for each slot when inputting sound slot output data to the DSP MIXS register. (Refer to Table 8-23 below.)

***DISDL[3:0]***

This register specifies the send level for each slot when outputting direct data to the DAC. (Refer to the table below.)

***EFSDL[3:0]***

This register specifies the send level for each slot when outputting of effect data and external input data to the DAC.

Register value	Volume
0	-MAXdB
1	-42dB
2	-39dB
⋮	⋮
0xD	-6dB
0xE	-3dB
0xF	0dB

Table 8-23 Send Level

(Refer to section 8.1.1.6, "MIXER.")

***Q[4:0]***

This register contains resonance data, and sets the Q value for the FEG filter. A gain range from -3.00 to 20.25dB can be specified. The relationship between the bit settings and the gain is illustrated in the following table. ( $Q[\text{dB}] = 0.75 \times \text{register value} - 3$ )

DATA	GAIN[dB]	DATA	GAIN[dB]
11111	20.25	00110	1.50
11100	18.00	00100	0.00
11000	15.00	00011	-0.75
10000	9.00	00010	-1.50
01100	6.00	00001	-2.25
01000	3.00	00000	-3.00

Table 8-24 Resonance Data Setting Values

The definition of Q is illustrated in the following graph.

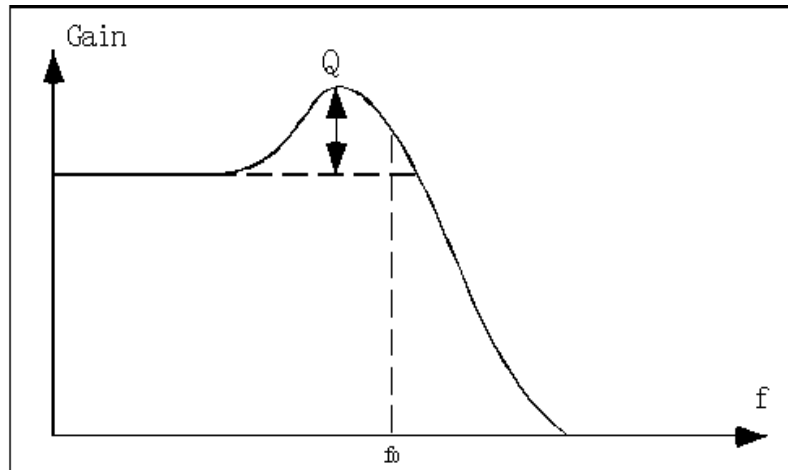


Fig. 8-13 Definition of Q

***FLV0[12:0]***

This is the cutoff frequency at attack start.

***FLV1[12:0]***

This is the cutoff frequency at attack end (decay start).

***FLV2[12:0]***

This is the cutoff frequency at decay end (sustain start).

***FLV3[12:0]***

This is the cutoff frequency at KOFF.

***FLV4[12:0]***

This is the cutoff frequency after release.

***FAR[4:0]***

However, only values ranging from 0x0008 to 0x1FF8 can be used for *FLV0* through 4. Playback may not be possible if any other values are used.

The following graph summarizes the function of each register.

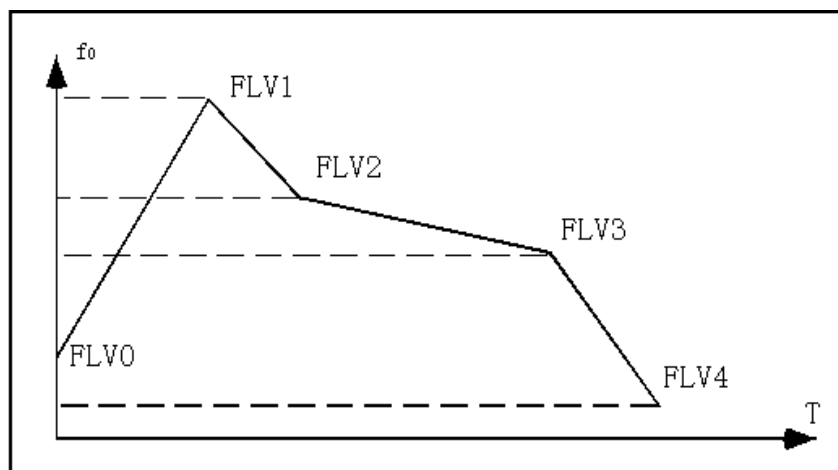


Fig. 8-14 Function of Each Register

The following graph roughly shows the correspondence between the filter cutoff frequency and the registers.

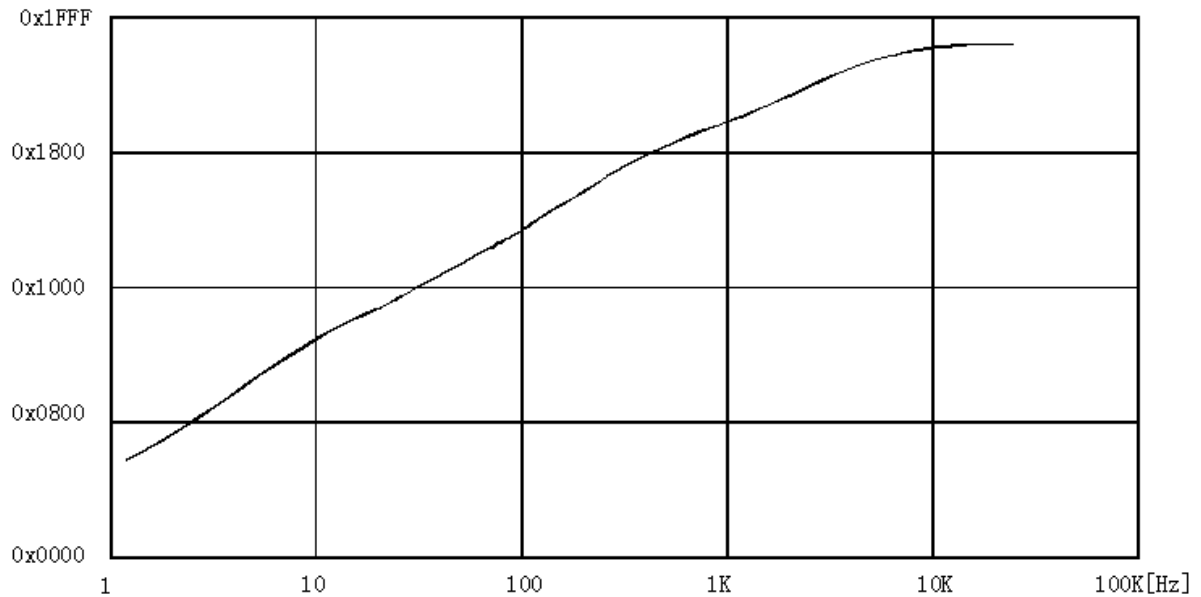


Fig. 8-15 Filter Cutoff Frequency

\* To set the filter to pass signals through, set **Q** to 4h and **FLV** to 0x1FF8.

***FAR[4:0]***

This register specifies the rate of change in the FEG in the attack state.

***FD1R[4:0]***

This register specifies the rate of change in the FEG in the decay 1 state.

***FD2R[4:0]***

This register specifies the rate of change in the FEG in the decay 2 state.

***FRR[4:0]***

This register specifies the rate of change in the FEG in the release state.

**• Common Data (Data that Does Not Depend on the Channel)**

AICA ADDR.	G2 ADDR.	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00							
0x00802800	0x00702800	MN	--					M8	D8	VER[3:0]				MVOL[3:0]				MN:Mono D8:DAC18B M8:MEM8MB						
0x00802804	0x00702804	\$T	RBL	--		RBP[22:11]												\$T:TESTBo(IC TEST)						
0x00802808	0x00702808	--			OF	OE	IO	IF	IE	MIBUF[7:0]								IF:MIFUL IO:MIOVF OE:MOEMP OF:MOFUL IE:MIEMP						
0x0080280C	0x0070280C	--	AF	MSLC[5:0]					MOBUF[7:0]								AF:AFSET							
0x00802810	0x00702810	LP	SGC		EG[12:0]																			
0x00802814	0x00702814	CA[15:0]																						
0x00802880	0x00702880	DMEA[22:16]							--	\$TSCD[2:0]			\$T	MRWINH[3:0]				\$*** (IC TEST)						
0x00802884	0x00702884	DMEA[15:2]															--							
0x00802888	0x00702888	GA	DRGA[14:2]												--				GA:DGATE					
0x0080288C	0x0070288C	DI	DLG[14:2]												--				EX	DI:DDIR EX:DEXE				
0x00802890	0x00702890	--					TACTL[2:0]			TIMA[7:0]														
0x00802894	0x00702894	--					TBCTL[2:0]			TIMB[7:0]														
0x00802898	0x00702898	--					TCCTL[2:0]			TIMC[7:0]														
0x0080289C	0x0070289C	--					SCIEB[10:0]																	
0x008028A0	0x007028A0	--					SCIPD[10:0]																	
0x008028A4	0x007028A4	--					SCIRE[10:0]																	
0x008028A8	0x007028A8	--							SCILV0[7:0]															
0x008028AC	0x007028AC	--							SCILV1[7:0]															
0x008028B0	0x007028B0	--							SCILV2[7:0]															
0x008028B4	0x007028B4	--					MCIEB[10:0]																	
0x008028B8	0x007028B8	--					MCIPD[10:0]																	
0x008028BC	0x007028BC	--					MCIRE[10:0]																	
--	0x00702C00	--						*VREG		--								AR	AR:ARMRST					
0x00802D00	--	--							L7	L6	L5	L4	L3	L2	L1	Lo	For interruption							
0x00802D04	--	--							RP	M7	M6	M5	M4	M3	M2	M1	Mo	For interruption RP:ReadProtection						
--	0x00710000	*RTC[31:16]																						
--	0x00710004	*RTC[15:0]																						
--	0x00710008	--															*EN	EN:RTC Write Enable						

Table 8-25 Common Data



## Register Descriptions (Common Data)

The various registers that comprise the common data are described below.

### **MONO** (-/w)

- 0: Enables the panpot information.
- 1: Disables the panpot information.

(Note) If the panpot information has been disabled, a sound that is on only one channel doubles in volume, so it is necessary to lower *MVOL*.

### **MVOL[3:0]** (-/w)

This is the master volume for the digital output to the DAC.  
(Refer to section 8.1.1.6, "MIXER.")

### **DAC18B** (-/w)

- 0: Makes the digital output a 16-bit DAC interface.
- 1: Makes the digital output an 18-bit DAC interface.

### **MEM8MB** (-/w)

This register specifies the size of the memory that is used for wave memory.  
0 : 16Mbit\_SDRAM  
1 : 64Mbit\_SDRAM

The following table indicates the relationship between memory size and the memory space that is used.

ADDRESS	16Mbit SDRAM	64Mbit SDRAM
0x00FF FFFF   0x00E0 0000	Not Available	Available
0x00DF FFFF   0x00C0 0000		
0x00BF FFFF   0x00A0 0000		
0x009F FFFF   0x0080 0000		
	Available	

Table 8-26 Relationship between Memory Space and the Memory That Is Used

### **VER[3:0]** (r/-)

This register is used to read the version information for the AICA chip based on these specifications.

### **RBL[1:0]** (-/w)

This specifies the length of the ring buffer.  
0 : 8K words  
1 : 16K words  
2 : 32K words  
3 : 64K words

### **RBP[22:11]** (-/w)

This register specifies the starting address of the ring buffer. (1K word boundary)

### **MIBUF[7:0]** (r/-)

This register is the MIDI input data buffer. (4-byte FIFO buffer)

***MIOVF*** (r/-)

This register indicates that the input FIFO buffer has overflowed.

***MIEMP*** (r/-)

Indicates that the input FIFO is empty.

***MIFUL*** (r/-)

This register indicates that the input FIFO buffer is full (i.e., has no free space).

(The three flags ***MIOVF***, ***MIEMP*** and ***MIFUL*** indicate the status before reading ***MIBUF[7:0]***.)

***MOFUL*** (r/-)

This register indicates that the output FIFO buffer is full.

***MOEMP*** (r/-)

This register indicates that the output FIFO buffer is empty.

***MOBUF[7:0]*** (-/w)

This register is the MIDI output data buffer.

***AFSEL*** (-/w)

This register determines whether to use the AEG or the FEG to monitor the EG.

0: AEG monitor

1: FEG monitor

***MSLC[5:0]*** (-/w)

This register specifies the slot number for which to monitor ***SGC***, ***CA***, ***EG***, and ***LP*** below.

***SGC[1:0]*** (r/-)

This register monitors the current EG status.

0: Attack

1: Decay 1

2: Decay 2

3: Release

***CA[15:10]*** (r/-)

This register indicates the position of the sample that is currently being read from the sound source, in terms of the upper 16 bits of the relative sample number from the SA. The LSB is equivalent to one sample.

***EG[12:0]*** (r/-)

These bits monitor the upper 13 bits of the current EG value. Only the lower 10 bits are valid for AEG. When the channel is selected by ***MSLC[5:0]***, these flags can be used to check for the loop end. Performing a read while a flag is "1" clears that flag to "0."

***LP*** (r/-)

This bit is set when the sample position that is read by the sound source loops. However, this bit is undefined when monitoring FEG. When a slot for which this bit has been set is set in ***MSLC[5:0]***, the flag is cleared to "0" by reading ***LP***.

***MRWINH[3:0]*** (-/w)

By writing a "1" to each of the bits shown below, the corresponding type of wave memory access can be prohibited. (Register access cannot be prohibited.)

bit 0: Access by DSP

bit 1: Read by sound source

bit 2: Access by AICA's built-in sound processor (ARM, hereafter) (This bit cannot be written by the ARM.)

bit 3: Access by system (SH4)

***DGATE*** (r/w)

This register specifies zero clear of the destination area by DMA transfer.

0: Zero clear is not executed.

1: Zero clear is executed.

***DDIR*** (r/w)

This register specifies the DMA transfer direction.

- 0: Transfer to AICA register from wave memory.
- 1: Transfer to wave memory from AICA register.

**DEXE** (r/w)

This register specifies DMA start. (The value goes to "0" at DMA end.)  
Writing a "1" to this register starts DMA. (Writing "0" is invalid.)

**DMEA[22:2]** (-/w)

This register specifies, as a word address, the wave memory address where DMA is to start.

**DRGA[14:2]** (-/w)

This register specifies, as a word address, the internal register address where DMA is to start.

**DLG[14:2]** (-/w)

This register specifies the number of words to be transferred by DMA.

(Note) The source and destination areas must not exceed the memory area and the internal register area. DMA-related registers must not be changed while DMA transfer is in progress.

(Supplement) Registers are allocated to the memory space [AICA:0x00800000-0x008045C7],[G2:0x00700000-0x007045C7]. The transfer address always changes in the increasing direction. DMA to an RTC register is not possible. The offset address from the start of each area is input in the *DMEA* and *DRGA* registers.

**TACTL[2:0]** (-/w)

This register specifies the cycle for incrementing timer A.

- 0: Increment once every sample
- 1: Increment once every 2 samples
- 2: Increment once every 4 samples
- 3: Increment once every 8 samples
- 4: Increment once every 16 samples
- 5: Increment once every 32 samples
- 6: Increment once every 64 samples
- 7: Increment once every 128 samples

**TIMA[7:0]** (-/w)

Timer A (An interrupt request is generated each time that the UP counter changes from All "1" to All "0".)

**TBCTL[2:0]** (-/w)

This register specifies the increment cycle for timer B. (The codes are the same as for timer A.)

**TIMB[7:0]** (-/w)

Timer B (Interrupt generation is the same as for timer A.)

**TCCTL[2:0]** (-/w)

This register specifies the increment cycle for timer C. (The codes are the same as for timer A.)

**TIMC[7:0]** (-/w)

Timer C (Interrupt generation is the same as for timer A.)

**SCIPD[10:0]**

This register stores interrupt requests to the ARM. (The bit correspondence is as shown below.)

- bit 0(r): Interrupt request to external interrupt input pin **INTN** (SCSI)
- bit 1(r): Reserved
- bit 2(r): Reserved
- bit 3(r): This is the MIDI input interrupt request; an interrupt request is generated when valid data is loaded into the input FIFO buffer. Therefore, when reading the FIFO buffer, it is necessary to read out the entire buffer in one operation, so that the FIFO buffer is then empty. This interrupt request is automatically cleared when the FIFO buffer is emptied.

bit 4(r): DMA end interrupt request

bit 5(r/w): This interrupt request to the ARM is written by the CPU; only a "1" can be written to this bit. (If a "0" is written, it is invalid.) This flag can be set by the system (SH4) or by the ARM.

bit 6(r): Timer A interrupt request

bit 7(r): Timer B interrupt request

bit 8(r): Timer C interrupt request

bit 9(r): This is the MIDI output interrupt request. This interrupt request is generated when the output FIFO buffer becomes empty. This interrupt request is automatically cleared when a write to the output FIFO buffer causes it to no longer be empty.

bit 10(r): Sample interval interrupt request

**SCIEB[10:0]** (r/w)

This register enables interrupts to the ARM. If a bit is set to "1," the interrupt that corresponds to that bit is enabled.

**SCIRE[10:0]** (-/w)

Writing a "1" to a bit in this register resets the interrupt request that corresponds to that bit.

**SCILV0[7:0]** (-/w)

This register specifies bit 0 of the level codes for the interrupts to the ARM that are defined by the corresponding bits.

**SCILV1[7:0]** (-/w)

This register specifies bit 1 of the level codes for the interrupts to the ARM that are defined by the corresponding bits.

**SCILV2[7:0]** (-/w)

This register specifies bit 2 of the level codes for the interrupts to the ARM that are defined by the corresponding bits. (For details on the bits, refer to the description of SCIPD.)

(Supplement) The level of bits 7, 8, 9, and 10 of an interrupt request can be specified as a group through bit 7.

#### **MCIPD[10:0]**

This register stores interrupt requests to the system (SH4).

bit 0(r): Interrupt request to external interrupt input pin **INTN** (SCSI)

bit 1(r): Reserved

bit 2(r): Reserved

bit 3(r): This is the MIDI input interrupt request; an interrupt request is generated when valid data is loaded into the input FIFO buffer. Therefore, when reading the FIFO buffer, it is necessary to read out the entire buffer in one operation, so that the FIFO buffer is then empty. This interrupt request is automatically cleared when the FIFO buffer is emptied.

bit 4(r): DMA end interrupt request

bit 5(r/w): This interrupt request to the system (SH4) is written by the CPU; only a "1" can be written to this bit. (If a "0" is written, it is invalid.) This flag can be set by the system (SH4) or by the ARM.

bit 6(r): Timer A interrupt request

bit 7(r): Timer B interrupt request

bit 8(r): Timer C interrupt request

bit 9(r): This is the MIDI output interrupt request. This interrupt request is generated when the output FIFO buffer becomes empty. This interrupt request is automatically cleared when a write to the output FIFO buffer causes it to no longer be empty.

bit 10(r): Sample interval interrupt request

#### **MCIEB[10:0] (r/w)**

This register enables interrupts to the system (SH4). If a bit is set to "1," the interrupt that corresponds to that bit is enabled.

#### **MCIRE[10:0] (-/w)**

Writing a "1" to a bit in this register resets the interrupt request that corresponds to that bit.

(Supplement) The **MCINTN** interrupt signal to the system (SH4) is regarded to indicate the start of the above interrupt request, and generates a negative pulse that corresponds to one clock cycle on "MCKK". Interrupt levels cannot be specified for interrupts to the system (SH4).

#### **ARMRST (r/w)**

This register resets the ARM.

0: Reset clear

1: Reset

(Note) This register can only be controlled by the system (SH4).

#### **RP (-/w)**

This register sets control of the SDRAM (wave memory) from the system side (SH4) to the "write only" state.

0: The system (SH4) can read/write SDRAM.

1: The system (SH4) can only write SDRAM.

(Note) This register can only be controlled by the ARM.

#### **L[7:0] (r/-)**

This register indicates the number of the interrupt input to the ARM. Note that using **L[7:3]** is prohibited.

(Note) This register can only be controlled by the ARM.

#### **M[7:0] (-/w)**

When the ARM has completed interrupt processing, it indicates the end of interrupt processing by setting this bit to "1". Note that using **M[7:1]** is prohibited.

(Note) This register can only be controlled by the ARM.

The following registers are inside the AICA, but are not related to the sound system. These registers can only be controlled from the SH4 side.

\* **VREG[1:0]** (r/w)

This register sets the operation and output mode of the DVE (Digital Video Encoder). (Refer to section 6.1, "DVE.") The relationship between the setting in this register and the output mode is shown in the table below.

VREG1	VREG0	DVE output mode
0	0	VGA ( RGB )
0	1	VGA ( RGB )
1	0	NTSC/PAL ( RGB )
1	1	NTSC/PAL ( VBS / Y+S / C )

Table 8-27 Video Mode Setting

\* **RTC[31:0]** (r/w)

This is the setting register for the AICA's internal real time clock. This register indicates the status of the counter that is incremented by one each second. This register permits both read and write access. This register can count for up to 136 years. For details on usage, refer to section 4.2.3, "RTC."

\* **EN** (-/w)

Setting this bit to "1" enables writes to the *RTC*. For details, refer to section 4.2.3, "RTC."

## • DSP Data

The register configuration of the DSP section in the chip is shown below.

AICA ADDR.	G2 ADDR.	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
0x00803000   0x008031FF	0x00703000   0x007031FF	COEF REG "COEF[12:0]"													0	0	0	00~127
0x00803200   0x008032FF	0x00703200   0x007032FF	MEMORY ADDRESS REG "MADRS[16:1]"															00~63	
0x00803400   0x00803404	0x00703400   0x00703404	DSP MICRO PROGRAM "MPRO[63:48]"															STEP_0	
0x00803408   0x0080340C	0x00703408   0x0070340C	DSP MICROPROGRAM "MPRO[47:32]"																
0x00803410   0x00803BEC	0x00703410   0x00703BEC	DSP MICROPROGRAM "MPRO[31:16]"																
0x00803410   0x00803BEC	0x00703410   0x00703BEC	DSP MICROPROGRAM "MPRO[15:0]"																
0x00803BF0   0x00803BF4	0x00703BF0   0x00703BF4	DSP MICRO PROGRAM "MPRO[63:48]"															STEP_127	
0x00803BF8   0x00803BFC	0x00703BF8   0x00703BFC	DSP MICROPROGRAM "MPRO[47:32]"																
0x00804000   0x008043FF	0x00704000   0x007043FF	DSP MICROPROGRAM "MPRO[31:16]"																
0x00804000   0x008043FF	0x00704000   0x007043FF	DSP MICROPROGRAM "MPRO[15:0]"																
0x00804400   0x008044FF	0x00704400   0x007044FF	--									LOW "TEMP[7:0]"						00~127	
0x00804400   0x008044FF	0x00704400   0x007044FF	TEMPBUFFER HIGH "TEMP[23:8]"																
0x00804500   0x0080457F	0x00704500   0x0070457F	--									LOW "MEMS[7:0]"						00~31	
0x00804500   0x0080457F	0x00704500   0x0070457F	SOUND MEMORY DATA HIGH "MEMS[23:8]"																
0x00804580   0x008045BF	0x00704580   0x007045BF	--												"MIXS[3:0]"			00~15	
0x00804580   0x008045BF	0x00704580   0x007045BF	MIXSOUND SLOT DATA STACK "MIXS[19:4]"																
0x008045C0   0x008045C7	0x007045C0   0x007045C7	EFCTED DATA OUTPUT "EFREG[15:0]"															00~15	
0x008045C0   0x008045C7	0x007045C0   0x007045C7	EXTERNAL INPUT DATA STACK "EXTS"															00~01	

Table 8-28 DSP Data

## Register Descriptions

The various registers that comprise the DSP data are described below. (*EXTS* is read-only; all of the other registers are read/write.)

### ***COEF[12:0]***

This is the DSP coefficient buffer. (Number of data items: 128)

(Note) In order to maintain compatibility in the event of a future expansion of the data width to 16 bits, a "0" should be written to the lower three bits that are undefined in the register map.

### ***MADRS[16:1]***

This is the DSP address buffer. (Number of data items: 64)

### ***MPRO[63:0]***

This is the DSP microprogram buffer. (Number of data items: 128)

### ***TEMP[23:0]***

This is the DSP work buffer. (Number of data items: 128)

This buffer has a ring buffer configuration, but the pointer is decremented by "1" for each sample.

### ***MEMS[23:0]***

This is the buffer for input data from wave memory. (Number of data items: 32)

Actual writes to *MEMS[7:0]* are executed at the same time as writes to *MEMS[23:16]*.

### ***MIXS[19:0]***

This is the buffer for the sound data from the input mixture. (Number of data items: 16)

(Note) Writes to *MIXS[19:0]* are used for LSI testing.

Writes that are not made in test mode are invalid for the following reasons:

- Regardless of the register settings, data from the sound source is always being written to this register.
- Second-generation data is retained in order to integrate all slots, but it is not possible to specify a generation when accessing the register.

### ***EFREG[15:0]***

This is the DSP output buffer. (Number of data items: 16)

### ***EXTS[15:0]***

This is the digital audio input data buffer. (Number of data items: 2)



## § 8.5 List of Interrupts

### § 8.5.1 Interrupt Tree

Diagram only

## § 8.5.2 List of Interrupt Sources (System Bus-related Interrupts)

i/f Block	Internal Name (Source)	Type	Description
PVR i/f	PIDEINT (End of DMA)	Notification	When a PVR-DMA transfer has been completed normally, this interrupt is generated when completion of the write in the destination has been confirmed for a "PVR CORE → Root Bus (the HOLLY's internal bus)" transfer, and when the write is completed on the PVR side for a "Root Bus → PVR" transfer.
	PIIAINT (Illegal Address Set)	Error	This interrupt is generated when an address outside of the range indicated in Note 1 <sup>1</sup> has been set in SB_PDSTAP(0x005F7C00), SB_PDSTR(0x005F7C04) both when the address is written to the register and when an attempt is made to initiate DMA with that address in effect.
	PIORINT (DMA Over Run)	Error	This interrupt is generated when a PVR-DMA transfer that is in progress attempts to access an address outside of the range specified by Note 1.
Maple i/f	MDEINT (End of DMA)	Notification	This interrupt is generated when a Maple-DMA transfer (transmission/reception) has ended normally, when the transfer is completed at the instruction level.
	MVOINT (V-Blank Over)	Notification	This interrupt is generated when a Maple interface transmission/reception operation spans V-Blank_In.
	MIAINT (Illegal Address Set)	Error	This interrupt is generated when an address outside of the range indicated in Note 2 <sup>2</sup> has been set in SB_MDSTAR(0x005F6C04) both when the address is written to the register and when an attempt is made to initiate DMA with that address in effect.
	MORINT (DMA Over Run)	Error	This interrupt is generated when a Maple-DMA transfer that is in progress attempts to access an address outside of the range specified by Note 2.
	MFOFINT (Write FIFO Over Flow)	Error	This interrupt is generated when an attempt was made to write data from a peripheral to the FIFO buffer, and the FIFO buffer was already full. The interrupt is generated at the time of the write to the FIFO buffer.
	MICINT (Illegal Command)	Error	This interrupt is generated when the Maple interface loaded in an illegal instruction through a transmission/reception operation. The interrupt is generated at the time of the instruction fetch.
G1bus i/f	G1DEINT (End of DMA)	Notification	When a G1-DMA transfer has been completed normally, this interrupt is generated when completion of the write in the destination has been confirmed for a "G1 Bus → Root Bus" transfer and when the write is completed on the G1 side for a "Root Bus → G1 Bus" transfer.
	G1IAINT (Illegal Address Set)	Error	This interrupt is generated when an address outside of the range indicated in Note 3 <sup>3</sup> has been set in SB_GDSTAR(0x005F7404) both when the address is written to the register and when an attempt is made to initiate DMA with that address in effect.
G1bus i/f	G1ORINT (DMA Over Run)	Error	This interrupt is generated when a G1-DMA transfer that is in progress attempts to access an address outside of the range specified by Note 3.

<sup>1</sup> Note 1: Range on the Power VR side (texture memory area) from 0x0400 0000 to 0x05FF FFE0, on the Root Bus side (system memory area) from 0x0C00 0000 to 0x0FFF FFE0, or the range specified by the register at 0x005F 7C80 (PVR-DMA System Memory Area Protection).

<sup>2</sup> Note 2: Range from 0x0C00 0000 to 0x0FFF FFE0 (system memory area), or the range specified by the register at 0x005F 6C8C (Maple System Memory Area Protection).

<sup>3</sup> Note 3: Range from 0x0080 0000 to 0x00FF FFE0 (wave memory), from 0x0400 0000 to 0x05FF FFE0 (texture memory), from 0x0C00 0000 to 0x0FFF FFE0 (system memory area), from 0x0300 0000 to 0x0300 000 to 0x03FF FFE0, from 0x1400 0000 to 0x17FF FFE0 (G2-external device), or the range specified by the register at 0x005F 74B8 (GD-DMA System Memory Area Protection).

	G1ATINT (G1 Access At DMA)	Error	This interrupt is generated when an attempt is made during a G1-DMA transfer to access ROM or the GD-ROM device on the G1 Bus from the Root bus.
	G1GDINT (From GD-ROM Drive)	Status	This interrupt is generated by the GD-ROM device. (This is an asynchronous level interrupt.)
G2bus i/f	G2DEAINT(End of AICA-DMA)	Notificat ion	When a G2-DMA transfer has been completed normally, these four interrupts are generated when completion of the write in the destination has been confirmed for a "G2 Bus → Root Bus" transfer, and when the write is completed on the G2 side for a "Root Bus → G2 Bus" transfer.
	G2DE1INT(End of Ext-DMA1)		
	G2DE2INT(End of Ext-DMA2)		
	G2DEDINT(End of Dev-DMA)		
	G2IAAINT (AICA-DMA Illegal Address Set)	Error	These four interrupts are generated when an address outside of the range indicated in Note 4 <sup>4</sup> has been set in the corresponding "DMA Start Address," both when the address is written to the register and when an attempt is made to initiate DMA with that address in effect.
	G2IA1INT (Ex1-DMA Illegal Address Set)		
	G2IA2INT (Ex2-DMA Illegal Address Set)		
	G2IADINT (Dev-DMA Illegal Address Set)		
	G2ORAIINT (AICA-DMA Over Run)	Error	These four interrupts are generated if, during an attempt at access by a G2-DMA transfer, the target device does not respond within the specified period of time.
	G2OR1INT (Ex1-DMA Over Run)		
	G2OR2INT (Ex1-DMA Over Run)		
	G2ORDINT (Dev-DMA Over Run)		
	G2TOAIINT (AICA-DMA Time Out)	Error	These four interrupts are generated if, during an attempt at access by a G2-DMA transfer, the target device does not respond within the specified period of time.
	G2TO1INT (EX1-DMA Time Out)		
	G2TO2INT (EX2-DMA Time Out)		
	G2TODINT (Dev-DMA Time Out)		
	G2TOCINT (Time Out in CPU Accessing)	Error	this interrupt is generated during an access from the CPU if the target device on the G2 Bus does not respond within the specified period of time.
	G2AICINT (from AICA)	Status	These three interrupts are interrupt signals from their respective devices; the timing with which these interrupts are generated depends on the device. (These are asynchronous level interrupts.)
	G2MDMINT (from Modem)		
	G2EXTINT (from Ext. DEV)		
DDT i/f	DTDE2INT (End of ch2-DMA)	Notificat ion	This interrupt is generated at the end of a DMA transfer.
	DTDESINT (End of Sort-DMA)	Notificat ion	This interrupt is generated at the end of a DMA transfer.
	DTCESINT (Sort-DMA Command Error)	Error	When a format that Sort-DMA cannot handle is encountered in the polygon parameters, this interrupt is generated while loading the Global Parameters.
SH4 i/f	CIHINT (Accessing to Inhibited Area)	Error	This interrupt is generated when the area indicated in Note 5 is accessed.

Table 8-29

### (Drawing Core-related Interrupts)

PCEOVINT (End Of Render Video)	Notificat ion	This interrupt is generated when the last data in a frame is transferred to the frame buffer.
PCEOIINT (End Of Render ISP)	Notificat ion	This interrupt is generated when rendering of the final Tile to the ISP has been completed.

<sup>4</sup> Note 4: Range on the G2 side from 0x0080 0000 to 0x00FF (wave memory), on the Root Bus side from 0x0400 0000 to 0x05FF FFE0 (texture memory), from 0x0C00 0000 to 0x0FFF FFE0 (system memory), or the range specified by the register at 0x005F 78BC (G2-DMA System Memory Area Protection).

<sup>5</sup> Note 5: An unused area of Area 0: 0x00400000 0x005F5FFF

PCEOTINT (End Of Render TSP)	Notification	This interrupt is generated when rendering of the final Tile to the TSP has been completed.
PCVIINT (V-Blank In)	Notification	Indicates the start of the V-Blank interval. (This interrupt is generated when the raster reaches the value specified by the SPG_VBLANK_INT register.)
PCVOINT (V-Blank Out)	Notification	Indicates the end of the V-Blank interval. (This interrupt is generated when the raster reaches the value specified by the SPG_VBLANK_INT register.)
PCHIINT (H-Blank In)	Notification	Indicates the start of the H-Blank interval. This interrupt can be generated either at a specified line, after every specified number of lines, or every line; this selection is made through the SPG_HBLANK_INT register.
PCIOCINT (ISP Out of Cache)	Error	ISP parameter cache overflow.
PCHZDINT (Hazard Processing of Strip Buffer)	Error	This interrupt is generated when rendering is forcibly terminated due to strip buffer switching.

Table 8-30

### (Tile Accelerator-related Interrupts)

TAYUVINT (End Of YUV Data Strage)	Notific ation	This interrupt is generated when the number of macroblocks of YUV data set in the TA_YUV_TEX_CTRL register have been stored in texture memory.
TAEOINT (End Of Opaque List Strage)	Notific ation	This interrupt is generated when, according to the End Of List (Control Parameter), all of the data in the Opaque List has been transferred to texture memory.
TAEOMINT (End Of Opaque Modifier Volume List Strage)	Notific ation	This interrupt is generated when, according to the End Of List (Control Parameter), all of the data in the Opaque Modifier Volume List has been transferred to texture memory.
TAETINT (End Of Translucent List Strage)	Notific ation	This interrupt is generated when, according to the End Of List (Control Parameter), all of the data in the Translucent List has been transferred to texture memory.
TAETMINT (End Of Translucent Modifier Volume List Strage)	Notific ation	This interrupt is generated when, according to the End Of List, all of the data in the Translucent Modifier Volume List has been transferred to texture memory.
TAEPTIN* . <del>From HOLLYe specifications</del> (End Of Punch Through List Strage)	Notific ation	This interrupt is generated when the transfer of Punch Through List data to texture memory is complete, as indicated by End Of List.
TAPOFINT (ISP/TSP Parameter Limit Address)	Error	This interrupt is generated when the ISP/TSP Parameter storage address has exceeded the value set in the TA_ISP_LIMIT register. Because the integrity of the display list cannot be guaranteed if this interrupt has been generated, it is necessary to start over from list initialization.
TALOFINT (Object List Limit Address)	Error	This interrupt is generated when the Object List storage address has exceeded the value set in the Object List Limit register. Because the integrity of the display list cannot be guaranteed if this interrupt has been generated, it is necessary to start over from list initialization.
TAIPINT (Illegal Parameter Input)	Error	This interrupt is generated if a parameter that is not a Vertex Parameter has been input, even though the Vertex Parameter that specifies "End Of Strip" (in the Parameter Control Word) has not been input.
TAFOFINT (TA FIFO Overflow)	Error	This interrupt is generated when an Overflow has occurred in the input data FIFO buffer. Because the input data is invalid and the operation of the TA after this interrupt has been generated cannot be guaranteed, it is necessary to execute a software reset, etc.

Table 8-31

## § 8.6 List of Input Parameters

### <Triangle Polygon Input Parameters>

The parameters that are input to the TA for a Triangle polygon are determined by the polygon data settings. The configuration of the input parameter data can be determined by checking the table below and the parameter tables on the following pages.

Switching of parameters inside/outside of a volume	Shading data type	Use of texture	Use of Offset Color	Number of UV bits	Input parameter number
Does not switch	Packed Color	Non-Textured	Not used	Not applicable	(1)
				32bit	(2)
				16bit	(3)
		Textured	Used	32bit	(2)
				16bit	(3)
				32bit	(3)
	Floating Color	Non-Textured	Not used	Not applicable	(4)
				32bit	(5)
				16bit	(6)
		Textured	Not used	32bit	(5)
				16bit	(6)
				32bit	(6)
	Intensity A	Non-Textured	Not used	Not applicable	(7)
				32bit	(8)
				16bit	(9)
		Textured	Not used	32bit	(10)
				16bit	(11)
				32bit	(11)
	Intensity B	Non-Textured	Not used	Not applicable	(12)
				32bit	(13)
				16bit	(14)
		Textured	Not used	32bit	(13)
				16bit	(14)
				32bit	(14)
Switches	Packed Color	Non-Textured	Not used	Not applicable	(15)
				32bit	(16)
				16bit	(17)
		Textured	Not used	32bit	(16)
				16bit	(17)
				32bit	(17)
	Floating Color	Non-Textured	Not used	Not applicable	Not supported
				32bit	Not supported
				16bit	Not supported
		Textured	Not used	32bit	Not supported
				16bit	Not supported
				32bit	Not supported
	Intensity A	Non-Textured	Not used	Not applicable	(18)
				32bit	(19)
				16bit	(20)
		Textured	Not used	32bit	(19)
				16bit	(20)
				32bit	(20)
	Intensity B	Non-Textured	Not used	Not applicable	(21)
				32bit	(22)
				16bit	(23)
		Textured	Not used	32bit	(22)
				16bit	(23)
				32bit	(23)

Table 8-32

<Note>

"Intensity A" specifies the Face Color through the immediately preceding Global Parameter, while "Intensity B" uses the Face Color that was used for the previous object.

(1) Packed Color Non-Textured	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 0
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
(ignored)	Base Color
(ignored)	(ignored)
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)

(2) Packed Color Textured 32bit UV	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 3
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U
(ignored)	V
Data Size for Sort DMA	Base Color
Next Address for Sort DMA	Offset Color

(3) Packed Color Textured 16bit UV	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 4
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U / V
(ignored)	(ignored)
Data Size for Sort DMA	Base Color
Next Address for Sort DMA	Offset Color

(4) Floating Color Non-Textured	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 1
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
(ignored)	Base Color Alpha
(ignored)	Base Color R
Data Size for Sort DMA	Base Color G
Next Address for Sort DMA	Base Color B

(5) Floating Color Textured 32bit UV	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 5
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U
(ignored)	V
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)
	Base Color Alpha
	Base Color R
	Base Color G
	Base Color B
	Offset Color Alpha
	Offset Color R
	Offset Color G
	Offset Color B

(6) Floating Color Textured 16bit UV	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 6
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U / V
(ignored)	(ignored)
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)
	Base Color Alpha
	Base Color R
	Base Color G
	Base Color B
	Offset Color Alpha
	Offset Color R
	Offset Color G
	Offset Color B

<b>(7) Intensity A Non-Textured</b>	
Global Parameter Polygon Type 1	Vertex Parameter Polygon Type 2
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
Face Color Alpha	Base Intensity
Face Color R	(ignored)
Face Color G	(ignored)
Face Color B	(ignored)

<b>(8) Intensity A Textured no Offset Color 32bit UV</b>	
Global Parameter Polygon Type 1	Vertex Parameter Polygon Type 7
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
Face Color Alpha	U
Face Color R	V
Face Color G	Base Intensity
Face Color B	(ignored)

<b>(9) Intensity A Textured no Offset Color 16bit UV</b>	
Global Parameter Polygon Type 1	Vertex Parameter Polygon Type 8
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
Face Color Alpha	U / V
Face Color R	(ignored)
Face Color G	Base Intensity
Face Color B	(ignored)

<b>(10) Intensity A Textured use Offset Color 32bit UV</b>	
Global Parameter Polygon Type 2	Vertex Parameter Polygon Type 7
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U
(ignored)	V
<i>Data Size for Sort DMA</i>	Base Intensity
<i>Next Address for Sort DMA</i>	Offset Intensity
Face Color Alpha	
Face Color R	
Face Color G	
Face Color B	
Face Offset Color Alpha	
Face Offset Color R	
Face Offset Color G	
Face Offset Color B	

<b>(11) Intensity A Textured use Offset Color 16bit UV</b>	
Global Parameter Polygon Type 2	Vertex Parameter Polygon Type 8
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U / V
(ignored)	(ignored)
<i>Data Size for Sort DMA</i>	Base Intensity
<i>Next Address for Sort DMA</i>	Offset Intensity
Face Color Alpha	
Face Color R	
Face Color G	
Face Color B	
Face Offset Color Alpha	
Face Offset Color R	
Face Offset Color G	
Face Offset Color B	

<b>(12) Intensity B Non-Textured</b>	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 2
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
(ignored)	Base Intensity
(ignored)	(ignored)
<i>Data Size for Sort DMA</i>	(ignored)
<i>Next Address for Sort DMA</i>	(ignored)

<b>(13) Intensity B Textured</b>
--------------------------------------

<b>(14) Intensity B Textured</b>
--------------------------------------



<b>32bit UV</b>	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 7
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U
(ignored)	V
Data Size for Sort DMA	Base Intensity
Next Address for Sort DMA	Offset Intensity

<b>16bit UV</b>	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 8
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U / V
(ignored)	(ignored)
Data Size for Sort DMA	Base Intensity
Next Address for Sort DMA	Offset Intensity

<b>(15) Packed Color Non-Textured Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 9
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
(ignored)	Z
TSP Instruction Word 1	Base Color 0
(ignored)	Base Color 1
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)

<b>(16) Packed Color Textured 32bit UV Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 11
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	U0
Texture Control Word 1	V0
Data Size for Sort DMA	Base Color 0
Next Address for Sort DMA	Offset Color 0
	U1
	V1
	Base Color 1
	Offset Color 1
	(ignored)
	(ignored)
	(ignored)
	(ignored)

<b>(17) Packed Color Textured 16bit UV Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 12
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	U0 / V0
Texture Control Word 1	(ignored)
Data Size for Sort DMA	Base Color 0
Next Address for Sort DMA	Offset Color 0
	U1 / V1
	(ignored)
	Base Color 1
	Offset Color 1
	(ignored)
	(ignored)
	(ignored)
	(ignored)

<b>(18) Intensity A Non-Textured Two Volumes</b>	
Global Parameter Polygon Type 4	Vertex Parameter Polygon Type 10
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
(ignored)	Z
TSP Instruction Word 1	Base Intensity 0
(ignored)	Base Intensity 1
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)
Face Color Alpha 0	
Face Color R 0	
Face Color G 0	
Face Color B 0	
Face Color Alpha 1	
Face Color R 1	
Face Color G 1	
Face Color B 1	

<b>(19) Intensity A Textured 32bit UV Two Volumes</b>	
Global Parameter Polygon Type 4	Vertex Parameter Polygon Type 13

<b>(20) Intensity A Textured 16bit UV Two Volumes</b>	
Global Parameter Polygon Type 4	Vertex Parameter Polygon Type 14

Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	Uo
Texture Control Word 1	Vo
<i>Data Size for Sort DMA</i>	Base Intensity 0
<i>Next Address for Sort DMA</i>	Offset Intensity 0
Face Color Alpha 0	U1
Face Color R 0	V1
Face Color G 0	Base Intensity 1
Face Color B 0	Offset Intensity 1
Face Color Alpha 1	(ignored)
Face Color R 1	(ignored)
Face Color G 1	(ignored)
Face Color B 1	(ignored)

Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	Uo / Vo
Texture Control Word 1	(ignored)
<i>Data Size for Sort DMA</i>	Base Intensity 0
<i>Next Address for Sort DMA</i>	Offset Intensity 0
Face Color Alpha 0	U1 / V1
Face Color R 0	(ignored)
Face Color G 0	Base Intensity 1
Face Color B 0	Offset Intensity 1
Face Color Alpha 1	(ignored)
Face Color R 1	(ignored)
Face Color G 1	(ignored)
Face Color B 1	(ignored)

<b>(21) Intensity B Non-Textured Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 10
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
(ignored)	Z
TSP Instruction Word 1	Base Intensity 0
(ignored)	Base Intensity 1
<i>Data Size for Sort DMA</i>	(ignored)
<i>Next Address for Sort DMA</i>	(ignored)

<b>(22) Intensity B Textured 32bit UV Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 13
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	Uo
Texture Control Word 1	Vo
<i>Data Size for Sort DMA</i>	Base Intensity 0
<i>Next Address for Sort DMA</i>	Offset Intensity 0
	U1
	V1
	Base Intensity 1
	Offset Intensity 1
	(ignored)
	(ignored)
	(ignored)
	(ignored)

<b>(23) Intensity B Textured 16bit UV Two Volumes</b>	
Global Parameter Polygon Type 3	Vertex Parameter Polygon Type 14
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word 0	Y
Texture Control Word 0	Z
TSP Instruction Word 1	U <sub>0</sub> / V <sub>0</sub>
Texture Control Word 1	(ignored)
<i>Data Size for Sort DMA</i>	Base Intensity 0
<i>Next Address for Sort DMA</i>	Offset Intensity 0
	U <sub>1</sub> / V <sub>1</sub>
	(ignored)
	Base Intensity 1
	Offset Intensity 1
	(ignored)
	(ignored)
	(ignored)
	(ignored)

#### <Quad Polygon Input Parameters>

There are two types of parameters that are input to the TA for a Quad polygon, depending on whether textures are used or not. The configuration of the input parameter data can be determined by checking the table below. Settings that are not found in the table below are not supported.

Switching of parameters inside/outside of a volume	Shading data type	Use of texture	Use of Offset Color	Number of UV bits	Input parameter number
Does not switch	Packed Color	Non-Textured	Not used	Not applicable	(1)
		Textured	Not used	32bit	Not supported
				16bit	(2)
			Used	32bit	Not supported
				16bit	(2)

<b>(1) Non-Textured</b>	
Global Parameter Sprite	Vertex Parameter Sprite Type 0
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	AX
TSP Instruction Word	AY
(ignored)	AZ
Base Color	BX
(ignored)	BY
<i>Data Size for Sort DMA</i>	BZ
<i>Next Address for Sort DMA</i>	CX
	CY
	CZ
	DX
	DY
	(ignored)
	(ignored)
	(ignored)
	(ignored)

<b>(2) Textured 16bit UV</b>	
Global Parameter Sprite	Vertex Parameter Sprite Type 1
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	AX
TSP Instruction Word	AY
Texture Control Word	AZ
Base Color	BX
Offset Color	BY
<i>Data Size for Sort DMA</i>	BZ
<i>Next Address for Sort DMA</i>	CX
	CY
	CZ
	DX
	DY
	(ignored)
	AU / AV
	BU / BV
	CU / CV

#### <Shadow Volume Input Parameters>

There is only one type of parameter (shown in the table below) that is input to the TA for a shadow volume.

Global Parameter Shadow Volume	Vertex Parameter Shadow Volume
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	AX
(ignored)	AY
(ignored)	AZ
(ignored)	BX
(ignored)	BY
(ignored)	BZ
(ignored)	CX
	CY
	CZ
	(ignored)
	(ignored)
	(ignored)
	(ignored)
	(ignored)
	(ignored)

### <Control Parameter>

There are three types of Control Parameters (shown in the table below) that are input to the TA.

Control Parameter End Of List
0x0000 0000
(ignored)
(ignored)
(ignored)
(ignored)
(ignored)
(ignored)
(ignored)

Control Parameter User Tile Clip
0x2000 0000
(ignored)
(ignored)
(ignored)
User Clip X Min
User Clip Y Min
User Clip X Max
User Clip Y Max

Control Parameter Object List Set
0x4000 0000
Object Pointer
(ignored)
(ignored)
Bounding Box X Min
Bounding Box Y Min
Bounding Box X Max
Bounding Box Y Max

~~\* The parameters shown below are changed in HOLLY2 versus HOLLY1.~~

<del>(1) Packed Color</del> <del>Non-Textured</del>	
Global Parameter Polygon Type 0	Vertex Parameter Polygon Type 0
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
(ignored)	Base Color
(ignored)	(ignored)
Data Size for Sort DMA	(ignored)
Next Address for Sort DMA	(ignored)

<del>(7) Intensity Mode 1</del> <del>Non-Textured</del>	
Global Parameter Polygon Type 1	Vertex Parameter Polygon Type 2
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
(ignored)	Z
Face Color Alpha	(ignored)
Face Color R	(ignored)
Face Color G	Base Intensity
Face Color B	(ignored)

<del>(11) Intensity Mode 1</del> <del>Textured</del> <del>use Offset Color</del> <del>16bit UV</del>	
Global Parameter Polygon Type 2	Vertex Parameter Polygon Type 8
Parameter Control Word	Parameter Control Word
ISP/TSP Instruction Word	X
TSP Instruction Word	Y
Texture Control Word	Z
(ignored)	U/V
(ignored)	(ignored)
Data Size for Sort DMA	Base Intensity
Next Address for Sort DMA	Offset Intensity
Face Color Alpha	
Face Color R	
Face Color G	
Face Color B	
Face Offset Color Alpha	
Face Offset Color R	
Face Offset Color G	
Face Offset Color B	

## § 9 Bug List

A list of the bugs in each Holly revision that affect software development is provided in this section. The Holly chip revision number can be determined through the registers indicated in Table 9-1. The subsequent bug list must be referenced according to the revision number of the chip as determined by the values of the registers listed below.

\* Regarding Holly 2.4, the bug list includes the contents of version 2.41.

No.	Register	Address	Holly 5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
1	REVISION	0x005F8004	0x01	0x11	0x11	0x11	0x11
2	SB_REVISION	0x005F689C	0x02	0x08	0x09	0x0A	0x0B

Table 9-1

<<Register-related bugs>>

A list of register-related bugs is shown below.

No.	Problem	Restriction/remedy	Holly 1.5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
1	Not possible to read registers, palette RAM, or fog table RAM correctly.	There software work-around.	×	○	○	○	○
2	Incorrect description of the SOFTRESET register (0x005F8008) in the documentation	<Wrong>: bit 0 = TA soft reset, bit 1 = Pipeline soft reset <Right>: bit 1 = Pipeline soft reset, bit 0 = TA soft reset	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
3	Incorrect description of the ISP_FEED_CFG register in the documentation	<Wrong>: Address = 0x005F8090 <Right>: Address = 0x005F8098	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
4	Incorrect description of the SPG_VBLANK_INT register (0x005F80CC) in the documentation	<Wrong>: bit 25-16 default = 0x015 <Right>: bit 25-16 default = 0x150, recommended value = 0x015	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
5	Incorrect description of the SPG_VBLANK register (0x005F80DC) in the documentation	<Wrong>: bit 25-16 default = 0x015 <Right>: bit 25-16 default = 0x150, recommended value = 0x015	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
6	Incorrect description of the FPU_PARAM_CFG register (0x005F807C) in the documentation	<Wrong>: bit 7-4 default=0xF <Right>: bit 7-4 default=0x7, recommended value = 0x015	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
7	Incorrect description of the FB_R_CTRL register (0x005F8044) in the documentation	<Wrong>: bit 20-16 = fb_stripsize size of strip buffer in multiples of 32 lines. <Right>: bit 21-16 = fb_stripsize size of strip buffer in multiples of 32 lines. (bit 16 = 0)	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion	Corrected documenta tion
8	Does not operate correctly even when a "1" is written to the lowest bit of fb_stripsize in the FB_R_CTRL register (0x005F8044).	The strip buffer size is restricted to 32-line units only. → <b>Will become part of the specifications.</b>	×	×	×	×	×
9	The wrong value is read from the SPAN_SRT_CFG register (0x005F8030).	The value of bit 8 is read for both bit 8 and bit 16.	×	○	○	○	○

Table 9-2

<<SB-related bugs>>

A list of SB (System Bus) block-related bugs is shown below.

No.	Problem	Restriction/remedy	Holly 1.5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
1	Addition of a TA processing end interrupt signal for Punch Through lists	Internal signal name: TA_ptendint_n	×	○	○	○	○
2	<G1 i/f> In the case of DMA with a transfer size of "32 bytes ???, 32 bytes or more, or where bit 5 is 0", DMA does not end.	Do not perform DMA transfers of these sizes. To transfer data in these sizes, use PIO access.	×	○	○	○	○
3	<G1 i/f> If a DMA-Read and a PIO-Read overlap, control of the DMA transfer may be lost.	Wait until the DMA-Read ends, or interrupt it, before conducting the PIO access.	×	○	○	○	○
4	<G1 i/f> If a PIO access overlaps with the end of a DMA-Read, the G1 block may hang.	Wait until the DMA-Read ends, or interrupt it, before conducting the PIO access.	×	○	○	○	○
5	<Maple i/f> Correct DMA initiation by V-Blank is not possible.	Using DMA initiation by V-Blank is prohibited.	×	○	○	○	○
6	<Maple i/f> If there is a Maple command set in system memory, and it is part of a special pattern that includes single instructions to the Maple-Host (output reset, switch gun mode, illegal command), the Maple block may hang.	Send single instructions (output reset, switch gun mode, illegal command) in a special format that does not cause the Maple block to hang. (The current Systems Laboratories driver does not support single instructions.) - Will become part of the specifications. → <b>Will become part of the specifications.</b>	×	×	×	×	×
7	<DDT i/f> The C2DMAXL (ch2-DMA maximum burst length) register does not function as it should.	The settings are restricted to 0, 1, or 2. → <b>Will become part of the specifications.</b>	×	×	×	×	×

Table 9-3

<<CORE & TA-related bugs>>

A list of CORE and TA block-related bugs is shown below.

No.	Problem	Restriction/remedy	Holly 1.5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
1	A red ghost pixel appears on the left side a white pixel, etc.	This is a problem with the internal circuitry that manifests itself with certain pixel data patterns. There is no software work-around.	○	○	○	○	○
2	If FB is specified in the first 4MB, vertical or diagonal lines appear.	FB must not be specified in the first 4MB (32-bit area). Specify FB in the second 4MB.	○	○	○	○	○
3	During drawing, vertex data is sometimes not drawn properly.	There is no software work-around.	○	○	○	○	○
4	The PAL non-interlaced VSYNC width is 3H.	The DVE compensates to 2.5H, so this is not a problem. → <b>Will become part of the specifications.</b>	×	×	×	×	×
5	Modifier Volumes do not work for sprites (quad polygons).	There is no software work-around. Modifier Volumes cannot be applied to sprites.	×	○	○	○	○
6	Lines are shifted to the right on the screen display.	The following software work-arounds are available: 1) Do not store FB and texture data in the same bank in SDRAM. 2) Set texture filtering to "point sampling."	×	○	○	○	○
7	Using the Vertex Parameter type 11, 12, 13, or 14 for TA, the block hangs.	When using type 11, 12, 13, or 14, set the partition strip length as either 1 strip or 2 strips. 4 strips or 6 strips cannot be specified.	×	○	○	○	○
8	The TSP cache circuit does not operate properly, causing tiles to be missing or copied, or for polygons to be distorted.	Do not set bit 16 of the SPAN_SORT_CFG register (0x005F8030) to "1" and then use the TSP cache.	○	○	○	○	○
9	Control on the TSP side of the FIFO between the Span Sorter and TSP does not operate correctly. The rendering operation hangs.	There is no software work-around. Setting "0x0001000" in the SPAN_SORT_CFG register can lessen this problem.	○	○	○	○	○
10	pixel write operation to the FB is not performed, and the previous pixels remain as is. This problem occurs more when the X-Scaler is used.	There is no software work-around. Using the X-Scaler makes the problem worse.	○	○	○	○	○
11	The "End_Of_Render" interrupt is not output.	Because this has the same cause as No. 10, there is no software work-around.	○	○	○	○	○
12	In Strip Buffer mode, the Hazard interrupt is output even though drawing has not been completed.	Ignore the first Hazard interrupt that is output. The Hazard interrupt will be output correctly after drawing is started a second and subsequent times. → <b>Will become part of the specifications.</b>	×	×	×	×	×
13	In Strip Buffer mode, if the Strip Buffer size is set to a number of lines that divides the display screen by an odd number, incorrect pixels will be drawn in the upper left corner of the screen.	The Strip Buffer size must be set to a number of lines that divides the screen by an even number. → <b>Will become part of the specifications.</b>	×	×	×	×	×



No.	Problem	Restriction/remedy	Holly 1.5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
14	If X clipping is used in Strip Buffer mode, incorrect pixels are drawn at the right edge of the screen.	The X clipping function must not be used in Strip Buffer mode. The FB_X_CLIP register value specifies the horizontal direction size of the display screen. → <b>Will become part of the specifications.</b>	×	×	×	×	×
15	Pixels are sometimes not drawn. (Apart from bug Nos. 10 and 11.) When this happens, the "End_Of_Render" interrupt is also not output.	There is no software work-around.	×	○	○	○	○
16	The TSP cache circuit does not operate properly. (Apart from bug No. 8.) The rendering operation hangs.	Do not set bit 16 of the SPAN_SORT_CFG register (0x005F8030) to "1" and then use the TSP cache.	×	○	○	○	○
17	If a user tile clip that is the same size as the screen or smaller is used, the TA sometimes hangs.	The following software work-arounds are available: 1) Set the strip length to "1". 2) Do not use user tile clips. Replace them with global tile clips.	×	○	○	○	○
18	The timing of switching of user tile clips (the area according to the control parameters, or the usage method according to the global parameters) is one polygon (= one strip) too early.	The following software work-arounds are available: 1) Add a dummy polygon before switching the clip. 2) Do not use user tile clips. Replace them with global tile clips. → <b>Will become part of the specifications.</b>	×	×	×	×	×
19	The "ispdone" flag in the ISP2 block is not cleared, making drawing impossible.	Perform a CORE reset each time before drawing.	—	×	○	○	○
20	The "End_Of_Video" interrupt is sometimes not output. (Apart from bug Nos. 11 and 15.)	Replace with "End_Of_TSP". → <b>Will become part of the specifications.</b>	—	×	×	×	×
21	Misshapen tiles or missing tiles occur.	Use multi-path processing to add a dummy region array. (4 data elements/tile) 1) Region Array for Opaque, Opaque MV, or Punch Through 2) Region Array for full-screen dummy translucent polygon (Autosort) 3) Region Array for Translucent or Translucent MV 4) Region Array for Accumulation Buffer Flush	—	×	×	○	○
22	If the translucent polygon sort mode is switched for each tile, the drawing operation may hang.	The following software work-arounds are available: 1) Do not switch the sort mode for each tile. 2) Register a full-screen opaque polygon for the background. → <b>Will become part of the specifications.</b>	—	×	×	×	×
23	At a boundary edge where a non-twiddle texture is switched with a palette texture, the colors of two pixels on the non-twiddled texture side are incorrect.	Do not use non-twiddled texture polygons and palette texture polygons at the same time.	—	×	×	○	○

No.	Problem	Restriction/remedy	Holly 1.5	Holly 2.2	Holly 2.3	Holly 2.4*	Holly 2.42
24	If the Y Scaler is enlarged beyond 0x400 and filtering is applied, the pixel color at the left end of the last line of FB output for the final tile will be affected as a result of filtering (32 pixels).	The following software work-arounds are available: 1) Reduce the number of display lines by one. 2) Add dummy final region array data in the upper right corner outside of the screen (Y = 0). → <b>Will become part of the specifications.</b>	—	×	×	×	×
25	If the polygon edge is located on the negative side (near "0") of the pixel center position, the gap will be written twice.	There is no software work-around. Making a running change is not expected to cause a problem. → <b>Will become part of the specifications.</b>	—	×	×	×	×
26	Operation hangs if a non-twiddled format bump texture is used.	Use only twiddled format for bump textures. → <b>Will become part of the specifications.</b>	—	×	×	×	×
27	The Group_En bit (bit 23) in the global parameters is not valid for the User_Clip bits (bits 167 and 16).	Specify the User-Clip bits correctly for each global parameter. → <b>Will become part of the specifications.</b>	×	×	×	×	×
28	The texture data flickers (VQ is obvious). The data in the texture memory may even be damaged.	There is no software fix for this problem. In Holly 2.41, this problem can sometimes be resolved by clamping. → <b>Will be incorporated into specifications.</b>	?	×	×	×/Δ	○

Table 9-4