

General Notice

When using this document, keep the following in mind:

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

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

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

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

SEGA Confidential

SEGA OF AMERICA

**Introduction to Saturn
Game Development**

April 13, 1994

Contents

Preface.....	iii
Organization of this document.....	iii
For more information.....	iii
Conventions.....	iii
Chapter 1: The Saturn System.....	1
System Control Unit (SCU).....	3
System Manager and Peripheral Control (SMPC).....	3
SH-2 CPUs.....	4
Cart port.....	4
CD-ROM subsystem.....	4
SH-1.....	4
MPEG decompression chip.....	5
Video subsystem.....	5
VDP 1.....	5
Dual frame buffer.....	6
VDP 2.....	6
Sound subsystem.....	7
SCSP.....	7
68EC000.....	7
Memory configuration.....	8
Chapter 2: Overview of VDP 1.....	11
Textured and nontextured parts.....	11
The display list.....	12
Specifying colors.....	15
Color calculations.....	16
Changing and erasing the frame buffer.....	16
Rotating the entire frame buffer.....	17
Chapter 3: Overview of VDP 2.....	19
Types of backgrounds.....	19
VRAM and the display interval.....	20
Character patterns and scroll planes.....	22
Scroll plane display.....	23
Scaling and rotation.....	23
Priority functions.....	24
Color processing.....	24
Chapter 4: Developing for Saturn.....	25
Graphics tools.....	25
SCONVERT.....	25
BRIP.....	26
SaturnSp_C PhotoShop/Debabelizer module.....	26
SaturnBRIP PhotoShop/Debabelizer module (under development).....	26
3DS2SAT (under development).....	26
Programming tools.....	27
CartDev system.....	28
The Hitachi E7000 ICEs.....	29
Sherry SH-2 simulator.....	30
High-level languages vs. assembler.....	30

Documentation.....	30
General Saturn documentation.....	30
Hitachi documentation	31
SCU documentation	31
CD-ROM subsystem documentation.....	31
Video subsystem documentation	31
Sound subsystem documentation.....	32
Glossary	33

SEGA Confidential

Preface

This document introduces the Saturn system, provides an overview of the video subsystem, and summarizes some of the resources available to Saturn game developers. You should read this document before reading any other Saturn documentation and before attending your first Saturn training session.

All information in this document is preliminary and subject to change.

Organization of this document

This document includes the following chapters:

- Chapter 1, “The Saturn System,” includes a block diagram of the system, a description of each of the main components, and a memory map.
- Chapter 2, “Overview of VDP 1,” describes the kinds of parts that VDP 1 can plot, the way VDP 1 uses its VRAM when it plots parts to the frame buffer, and some of its most important capabilities.
- Chapter 3, “Overview of VDP 2,” describes the kinds of backgrounds VDP 2 can plot, the mechanism it provides for accessing VRAM during the display interval, and some of the calculations it can perform as it displays each pixel.
- Chapter 4, “Developing for Saturn,” summarizes some of the content tools, programming tools, and other resources that Sega provides for Saturn developers.

The document ends with a glossary of key terms.

For more information

For more detailed information about Saturn, see the documents listed under “Saturn Documentation” in Chapter 4.

Conventions

This document describes memory in terms of kilobytes (KB) and megabytes (MB), not kilobits (Kbits) and megabits (Mbits). 1 Mbit = 1024 Kbits = 128 KB.

SEGA Confidential

Chapter 1: The Saturn System

Figure 1-1 shows the three buses and major components of the Saturn system, including the following:

- **SH-2 CPUs.** The main processor is a 28-MHz Hitachi 32-bit RISC chip (SH-2) that uses a second SH-2 as a “slave CPU” to speed processing of calculations such as matrix transformations. Both SH-2s have access to 1.5 MB of synchronous DRAM (labeled Work RAM in Figure 1-1).
- **System Control Unit (SCU).** Includes a programmable DSP, a DMA, and a bus controller that transparently translates addresses specified by the SH-2s into appropriate control signals for the other buses.
- **CD-ROM subsystem.** Includes a 20-MHz Hitachi SH-1 processor and an optional MPEG decompression chip, which if present connects directly to the video and sound subsystems.
- **Video subsystem.** VDP 1 plots parts (including textured parts) and supports 15-bit color. VDP 2 can plot up to five backgrounds simultaneously and supports 15-bit or 24-bit color. VDP 1 uses a dual frame buffer that allows it to plot a new frame while the previously plotted frame is being displayed, permitting display at 60 frames per second or at slower rates that divide evenly into 60 frames per second.
- **Sound subsystem.** Includes a custom SCSP chip that combines a PCM/FM sound source and sound-exclusive DSP, and a 68EC000 that runs at 11.3 MHz and can be programmed for 3-D sound and other effects.

You can program both SH-2s, the SCU DSP, the SCSP DSP, and the 68EC000 to achieve simultaneous processing of different kinds of data. For example, Saturn can play up to 32 sounds while calculating transformations of 3-D models and displaying the resulting 2-D sprites in real time.

The sections that follow summarize the capabilities of Saturn’s major components.

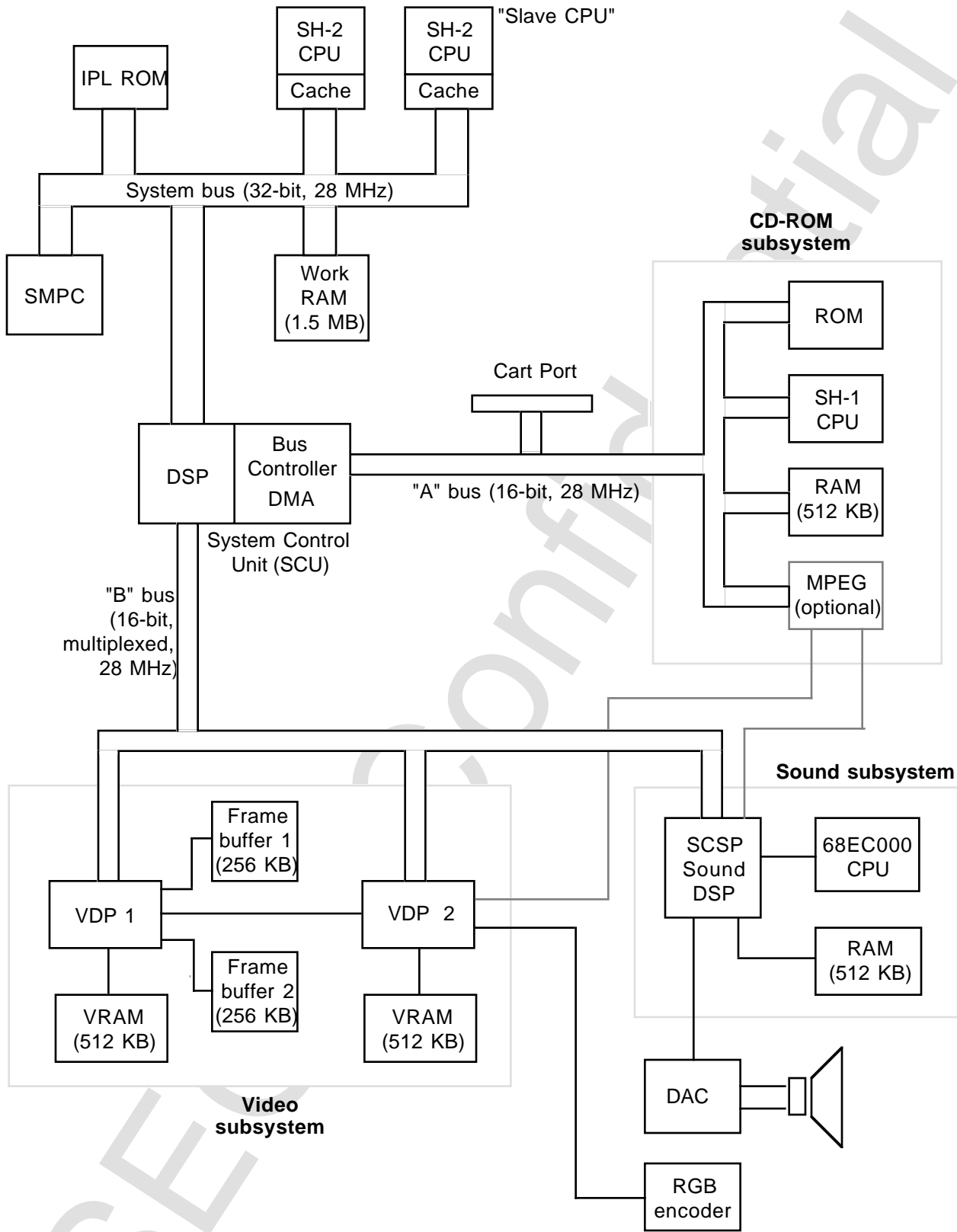


Fig. 1-1 Saturn block diagram

System Control Unit (SCU)

The SCU is built around a Harvard-architecture digital signal processor (DSP), a bus controller, and a direct memory access (DMA) chip. The bus controller translates addresses specified by the SH-2 CPU on the system bus into appropriate control signals for the other buses. This allows the SCU to integrate the A bus and B bus memory and processors into one large SH-2 memory map.

The SCU's DSP has a small program area in its RAM and some multiplication units. These can be useful for tasks such as 3-D transformations. For example, you can load a program into the DSP that performs a coordinate transformation for rotating an object. When you need to rotate that object, you send the matrix of vertices you want to multiply through the DMA with a command that runs the program in the DSP for each vertex. The resulting transformed matrix of vertices ends up wherever the DMA is sending it, in this case VDP 1's VRAM.

You must load any program you want to use into the DSP; it doesn't contain any hard-wired programs. Sega provides libraries of programs that perform matrix calculations and other common tasks.

The work RAM on the system bus can also be controlled via the SCU. This allows DMA between any parts of memory without involving the SH-2 CPU. For example, you can DMA from work RAM to VDP 1's VRAM or from the cartridge port ROM to VDP 2's RAM. For an overview of the system memory map, see "Memory Configuration" later in this chapter.

System Manager and Peripheral Control (SMPC)

The SMPC is built around a 4-bit single-chip Hitachi microcontroller that controls a real-time clock and can reset either the entire system or individual microprocessors (SH-2, SH-1, 68EC000, and SCSP). The SMPC also controls nonmaskable interrupts sent to the SH-2 and via the SCU to the 68EC000, SCSP, VDP 1, and VDP 2. This capability permits the SH-2, for example, to interrupt the 68EC000's processing to request that it play a particular sound. The SMPC runs continuously and is powered by a battery when the system is off.

The SMPC handles all input and output using one of two modes. In **direct mode**, the SH-2 CPU can access the peripheral directly. In **indirect mode**, the SMPC regularly polls for and buffers the latest information from a variety of peripheral devices, including the eight-button Saturn controller and other devices that use a 4-bit parallel protocol, three-line handshake devices, serial devices, and Genesis controllers like the six-button controller, the mouse, and the Genesis team player.

When you use the SMPC in indirect mode, you don't need to provide any I/O driver routines or poll the peripherals directly. Instead, your program can check SMPC registers whenever it needs peripheral data, and you can be sure that they contain the latest data. The SMPC continually polls for data and buffers whatever it finds in registers.

When you use the SMPC in direct mode, you must provide the appropriate drivers and poll the devices as necessary.

SH-2 CPUs

The main CPU for the Saturn system is an SH-2 microprocessor with a twin "slave" SH-2. Both chips run at 28 MHz. Because they are on a single system bus, one has to wait for the other if they both need to access the work RAM or anything else on the bus at the same time. However, each SH-2 includes cache RAM that you can configure either as a 4-KB 4-way write-through unified cache or as a 2-KB 2-way write-through unified cache plus 2 KB of RAM for use as private work RAM.

The Saturn comes with the main SH-2's cache RAM configured as a 4-KB 4-way cache and the slave SH-2's cache RAM configured as a 2-KB two-way cache with 2 KB of additional RAM.

Cart port

The cartridge port has 32 MB available in the system memory map. You can plug the CartDev system module directly into the cartridge port and perform debugging and other tasks via a SCSI connection with a personal computer. For more information about the CartDev system, see "Programming Tools" in Chapter 4.

CD-ROM subsystem

The CD-ROM subsystem is an independent device with its own SH-1 processor, a "2x" CD drive that reads data at 300 KB/sec, and a 512-KB data cache. It reads CD+G, CD Red book (audio CD), CD Yellow book (CD-ROM), and CDX-A formats.

SH-1

The 20-MHz SH-1 microprocessor provides fully independent control of the CD-ROM subsystem. For example, if you are writing a fast-paced game and you want the title screen to appear as soon as a player's character gets killed, you can send

commands to SH-1 from SH-2 that queue the title screen data into the data cache so that it will be ready for immediate display when you need it. This capability avoids the wait times that occur with most CD drives.

The ROM connected to the SH-1 houses the CD-ROM driver. You can't access the SH-1 directly; you can only request, via the driver, that the SH-1 read or write to the data cache.

MPEG decompression chip

The optional MPEG chip provides standard motion-picture-industry decompression for both sound and audio. If this chip is available, it pipes decompressed video directly to VDP 2 and decompressed audio directly to the SCSP without having to use the buses.

Video subsystem

VDP 1 plots parts to the frame buffer. VDP 2 integrates those parts with the backgrounds that it plots and displays the resulting image via the RGB encoder.

VDP 1 and VDP 2 work independently of each other and of the SH-2 CPU. For example, the SH-2 can perform matrix transformations or other processing at the same time that VDP 1 is plotting the display list to the inactive frame buffer and VDP 2 is displaying the contents of the active frame buffer and several backgrounds.

This section describes the basic functions of VDP 1 and VDP 2. For more information, see Chapters 2 and 3.

VDP 1

VDP 1 plots polygons and other shapes called **parts** independently of the backgrounds displayed by VDP 2. VDP 1 plots parts in the frame buffer one pixel at a time according to a list of commands, texture bitmaps, and other information in its VRAM.

Parts can be either textured or nontextured. A **textured part**, also called a **sprite**, is a polygon with four vertices that's filled with a bitmapped texture. You can specify the colors for a textured part's pixels as 15-bit RGB codes (from a total of 32,768 possible colors), palette offsets (from up to 256 entries) from a base address in VDP 2's color RAM, or entries in a 16-color color lookup table (CLUT). A **nontextured part** is a polygon (interior filled), a polyline (outline colored, interior empty), or a line. VDP 1 can apply a single RGB or paletted color to a nontextured part.

VDP 1 can enable or disable clipping, meshing, end codes, and see-through pixels for any part. If you specify the color for a part using RGB values, VDP 1 can also perform color calculations on the part's pixels when it plots them to the frame buffer, including Gouraud shading, shadowing, half luminance, and half transparency.

Dual frame buffer

A dual frame buffer allows VDP 1 to plot one frame while the previous frame is being displayed by VDP 2. VDP 2 integrates each frame with the current backgrounds, taking account of priority settings to determine how to display overlapping pixels.

The default display rate is 60 frames per second, but you can also manually control the way individual frames are erased and switched, which in turn determines how many frames are displayed per second. Display at 60 frames per second makes for smoother animation and a clearer image, but slower rates allow you to display more parts in a single frame.

VDP 2

VDP 2 can plot up to five backgrounds based on pattern name tables, character pattern tables, and other information in its VRAM. It can access four 128-KB banks of VRAM simultaneously during the four- or eight-cycle **display interval** after it displays one pixel and before it displays the next. You have complete control, via registers, of the way VDP 2 uses the cycles available in the display interval to access each bank of VRAM.

VDP 2 also includes 4 KB of color RAM that defines color palettes for use by both VDP 1 and VDP 2. You can use either 15-bit or 24-bit color for palette entries.

Four of the five backgrounds that VDP 2 can display are scroll planes. The picture in a **scroll plane** is larger than the TV display area and consists of tiled character patterns or a single bitmap image with an RGB color assigned to each pixel. Pixels specified with RGB colors may be 15-bit or 24-bit.

VDP 2 supports two kinds of scroll planes: normal scroll planes and rotation scroll planes. A **normal scroll plane** supports vertical and horizontal scrolling and can rotate around the z axis only. A **rotation scroll plane** supports two-axis rotation and scaling as well as vertical and horizontal scrolling. VDP 2 can display four normal scroll planes, or two normal scroll planes and one rotation scroll plane, or two rotation scroll planes.

The **back plane** is the same size as the TV display area and is visible only when all other backgrounds are transparent. The back plane can display a single RGB color or a different RGB color for each line, but can't display paletted colors.

Sound subsystem

The Saturn sound subsystem functions as a MIDI-compatible 32-channel sound source for either frequency modulation (FM) sounds or pulse-code modulation (PCM) sounds. Musicians can compose music or generate sound effects using standard MIDI programs on a personal computer. To play those sounds in a game, you load tone bank data, AIFF files, MIDI sequence data, DSP programs, and commands into the appropriate parts of the sound memory map and trigger the commands when necessary. The system can handle 8- or 16-bit samples.

A number of sound tools are available, including sampling and recording tools, a sound wave editor, and a sound compiler that creates sound banks and patch banks. These tools are described in the sound documentation listed under "Documentation" in Chapter 4. Sega also provides a library of general MIDI sounds.

SCSP

The custom SCSP chip has three main parts: a digital oscillator with 32 dual-purpose (PCM/FM) slots, a sound-exclusive 128-step DSP, and a digital mixer. The SCSP also acts as a DRAM controller for the 68EC000.

The oscillator feeds up to 32 channels into the DSP and converts the sampling frequency of each sample to the DSP sampling frequency of 44.1 KHz. You can use the 128-step program area in the DSP to apply various effects to all 32 channels, including reverberation, early reflection, echo, pitch shifting, surround sound, voice canceling, distortion, filters, panning, and parametric EQ. Sega provides tools that allow you to construct programs for the DSP on a Macintosh and to generate code you can download to the DSP from your program.

You can funnel output from the 32 slots into 16 input slots for the digital mixer. Each of the mixer's slots can accept more than one channel's output.

68EC000

The 68EC000 is a 16-bit CPU running at 11.3 MHz. Because the 68EC000 shares RAM on a time-sharing basis with the SCSP, the 68EC000 runs at half this speed when the SCSP is running at full specification.

The sound driver provided by Sega for use with the 68EC000 accepts MIDI commands and, like any MIDI sequencer, takes care of playing the sounds with the specified instrument, controller information, and so on. In most cases you don't need to provide your own sound driver, although you can if you want to.

Memory configuration

The SCU integrates the A bus and B bus memory and processors into one large SH-2 memory map. The SH-2 doesn't need to process additional instructions to access the multiplexed B bus. Instead, the SCU translates signals as necessary to provide transparent access to the entire system. You can access all memory and devices via the SH-2 memory map.

The SCU uses the 25-bit SH-2 address bus and four SH-2 chip selects to create the four 32-MB areas shown in Figure 1-2. These four areas represent the entire SH-2 memory map. The IPL ROM occupies a small portion of CS0, the SCU uses CS1 and CS2, and CS3 is SDRAM.

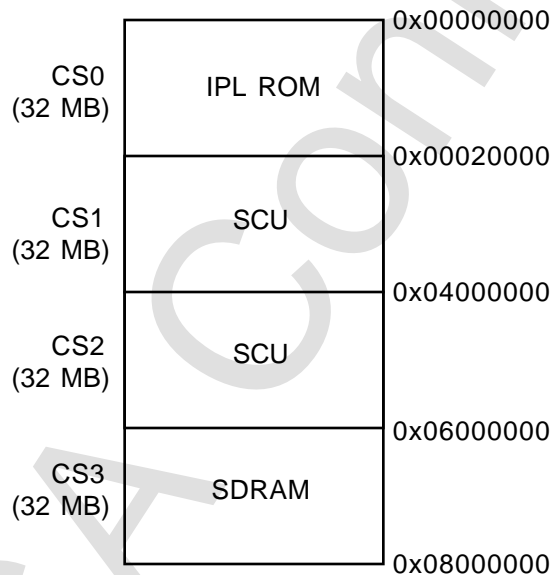


Fig. 1-2 Address space for the Saturn system

Figure 1-3 shows a simple Saturn memory map. Figure 1-2 and Figure 1-3 show addresses as **cache addresses**. If you use these addresses, SH-2 looks first in its RAM cache for the specified address and uses it via the cache. Alternatively, if you use a **cache through** address to refer to the same location in the memory map, SH-2 looks directly in external memory without checking the cache first, even if the cache controller is turned on.

Note: The memory map shown in Figure 1-3 is subject to change. For detailed memory maps, see the documentation for each component.

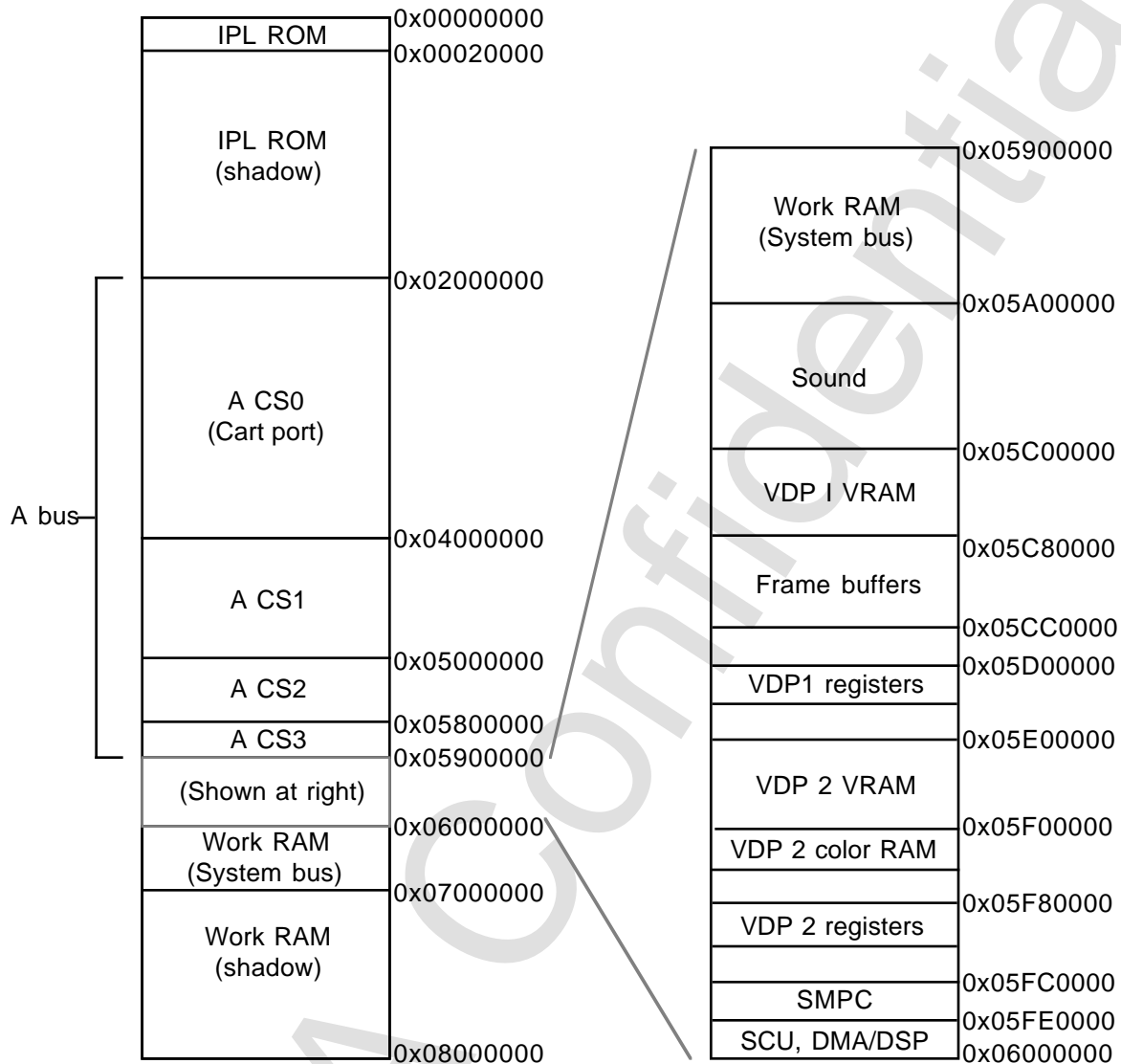


Fig. 1-3 Preliminary Saturn memory map

You can access the 1.5 MB of work RAM on the system bus either via the SCU at 0x5900000 or from the SH-2 at 0x06000000. If you use the SCU address, you can perform a DMA, for example, from work RAM directly through to VDP 1. In this case the SCU generates an address for the work RAM, gets the data, puts it in the work RAM at that address, generates an address for VDP 1, and passes the data directly to VDP 1. This is faster than having the SH-2 perform the same task.

SEGA Confidential

Chapter 2: Overview of VDP 1

VDP 1 plots parts to the frame buffer on the basis of commands, texture bitmaps, and other information in its VRAM. While VDP 1 is plotting parts to the inactive buffer, VDP 2 integrates the active frame buffer with the backgrounds defined by its VRAM and displays the result. The frame buffers then switch roles and the process is repeated. Figure 2-1 shows the relationship between VDP 1, the frame buffers, and VDP 2.

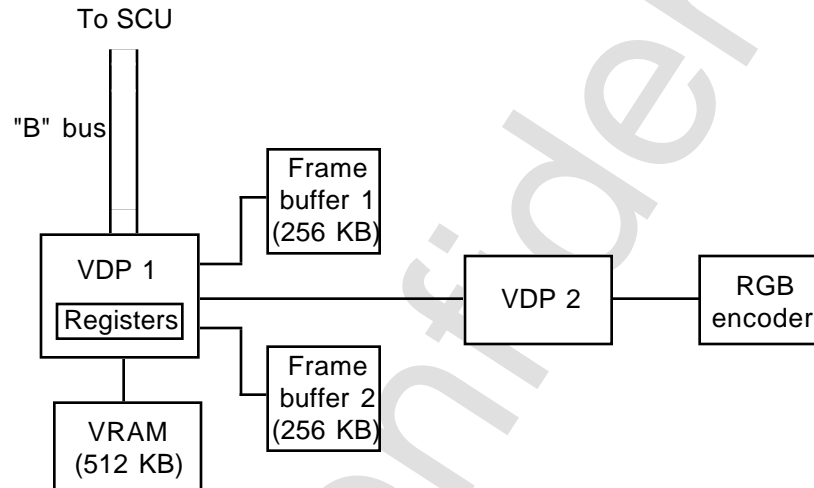


Fig. 2-1 Configuration of VDP 1

This chapter describes the kinds of parts that VDP 1 can plot, the way VDP 1 uses its VRAM when it plots parts to the frame buffer, and some of its most important capabilities. For detailed information about VDP 1, see the *VDP 1 Manual*.

Textured and nontextured parts

Parts can be either textured or nontextured. A **textured part**, also called a **sprite**, is a polygon with four vertices that's filled with a texture bitmap. VDP 1 can plot three kinds of textured parts:

- A **normal sprite** can be flipped horizontally, vertically, or both.
- A **scaled sprite** behaves like a standard sprite and can also be magnified or reduced horizontally, vertically, or both horizontally and vertically.
- A **distorted sprite** behaves like a standard sprite and can also be rotated and distorted by specifying the coordinates of four corner points. VDP 1 maps the

sprite's bitmapped image within those coordinates. This capability is useful for displaying 2-D transformations of 3-D objects.

You can specify colors for a sprite as RGB values, color lookup table (CLUT) entries, or offsets within palettes in VDP 2's color RAM. If you specify RGB colors for a sprite, you can also apply Gouraud shading and other color calculations to it.

A **nontextured part** is a shape to which VDP 1 can apply a single RGB value or a single palette color. VDP 1 can plot three kinds of nontextured parts:

- A **polygon** consists of an area specified by four points and filled with a single color. VDP 1 can apply Gouraud shading and other effects to polygons.
- A **polyline** is similar to a polygon except that VDP 1 colors and applies Gouraud shading and other effects to its outline only, not the area it encloses.
- A **line** is specified by two points, and VDP 1 can color and apply Gouraud shading and other effects to it.

When you specify Gouraud shading for a part, you must supply RGB color offsets for each of its vertices. VDP 1 can then interpolate intervening color offsets across all the part's pixels. For example, if you specify color offsets for the vertices of a polygon that's part of a three-dimensional object, VDP 1 can interpolate the intervening color offsets across the polygon's pixels and shade the color smoothly as if the polygon were reflecting light from a nearby source.

The display list

The first item in VDP 1's VRAM is the first entry in the **display list**, a list of commands that tell VDP 1 what to plot for a single frame. Each command in the display list is specified by a **command table**, a 32-byte block that also indicates which command to execute next and other information VDP requires to execute the command successfully. For sprites, this information includes coordinates within which to plot and the address of a sprite bitmap elsewhere in VRAM. Depending on the command, a command table may also specify the address of a color lookup table or a Gouraud shading table.

As long as VRAM begins with the display list, you can organize it however you like. Figure 2-2 shows a simplified example.

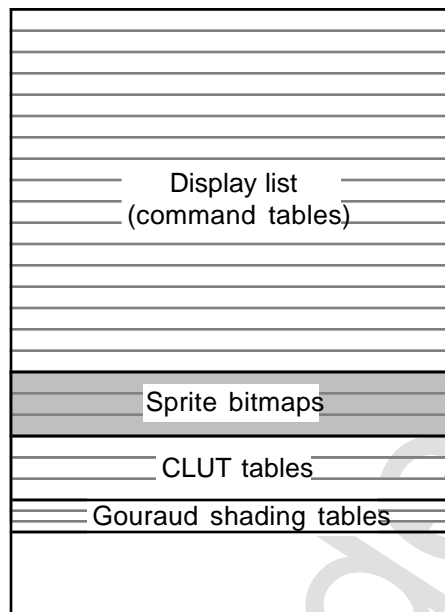


Fig. 2-2 Organization of VDP 1's VRAM

One way to manage VDP 1 is to maintain a shadow of the display list and related information in the SH-2's work RAM and replace the relevant parts of VRAM with a copy of the shadow during the VBL interrupt. VDP 1 then plots the sprites defined by the revised display list to the inactive frame buffer while VDP 2 is displaying the contents of the active frame buffer. Both byte access and word access are possible from the SH-2 or by DMA. All access to VRAM and the frame buffers occurs using burst transfer.

A command table in VDP 1's display list can specify one of the following commands:

- **Set User Clipping Coordinates.** Defines boundaries for the clipping of sprites that are plotted after this command. The command for each sprite specifies whether or not to clip and if so whether to clip inside or outside the currently defined boundaries.
- **Set System Clipping Coordinates.** Defines boundaries for the clipping of the entire frame buffer.
- **Set Local Coordinates.** Defines local coordinates for sprites that are plotted after this command.
- **Plot Normal Sprite.** Plots a normal sprite bitmap in the frame buffer.
- **Plot Scaled Sprite.** Plots a scaled sprite bitmap in the frame buffer.

- **Plot Distorted Sprite.** Plots a distorted sprite bitmap in the frame buffer.
- **Plot Polygon.** Plots a quadrangle with a colored interior in the frame buffer.
- **Plot Polyline.** Plots the outline of a quadrangle in the frame buffer.
- **Plot Line.** Plots a line in the frame buffer.
- **End Plotting.** Ends plotting of this frame.

You control the order in which VDP 1 processes the command tables in the display list by specifying a jump mode for each command. Jump modes instruct VDP to jump (after processing the current table) or skip (without processing the current table) to the next command table or to another command table elsewhere in the list. You can also use jump modes to specify another command as a subroutine of the current one, so that VDP 1 returns, after processing the subroutine, to the original command table or the one that follows it. Thus, if you want to keep a group of related commands that plot a particular object in one place in VRAM, you can move them in and out of the display list by manipulating jump modes.

Commands that plot parts also enable or disable the following modes:

- **Clipping.** If clipping is enabled, the command table format includes clipping coordinates and indicates whether clipping specified by a previous clipping command should be performed inside or outside of the area defined by the clipping coordinates. If clipping is disabled, VDP 1 ignores any previously processed Set User Clipping Coordinates command when it processes the part.
- **Meshing.** If meshing is enabled, VDP 1 plots every other pixel of a part in a fine checkerboard pattern that simulates transparency.
- **End codes.** If end codes are enabled and occur within a row of pixel descriptors in VRAM, VDP 1 plots only the portion of the scan line that lies between the end codes. End codes make it possible to display sprites with nonrectangular shapes much faster than would otherwise be possible.
- **See-through pixels.** If see-through pixels are enabled, see-through color codes in the sprite bitmap are not plotted, and sprites and backgrounds plotted underneath those pixels (that is, with a lower priority) will be visible. If see-through pixels are disabled, see-through color codes in the sprite bitmap are plotted like other color codes and make those pixels opaque.

In addition, commands that plot parts specify color modes and color calculations as described in the next sections.

Specifying colors

For a nontextured part, VDP 1 applies either a single 15-bit RGB color or colors from a single palette or CLUT while it plots the part. Any RGB color may be altered by the color calculations described in the next section while VDP 1 is plotting individual pixels to the frame buffer.

For a textured part, VDP 1 reads a sprite bitmap from VRAM and writes it to the frame buffer. Depending on a color mode set in the command table, VDP 1 interprets the sprite bitmap data as 4-bit offsets into a CLUT, 4- to 8-bit offsets into a color palette, or 16-bit RGB values.

VDP 1 writes a 16-bit **pixel descriptor** to the frame buffer for each of a sprite's pixels. For RGB data, this involves simply copying the pixel data from VRAM. In this case, the high bit of each pixel descriptor is set to 1, and the remaining 15 bits specify an RGB value. For CLUT or palette data, VDP combines bits set in the **color control word** of the command table with the 4- to 8-bit offsets specified in the bitmap to obtain the values for all 16 bits. In this case, the high bit of the color control word (and thus of the pixel descriptors that VDP 1 plots to the frame buffer) should be set to 0. Figure 2-3 shows these two basic formats for pixel descriptors.

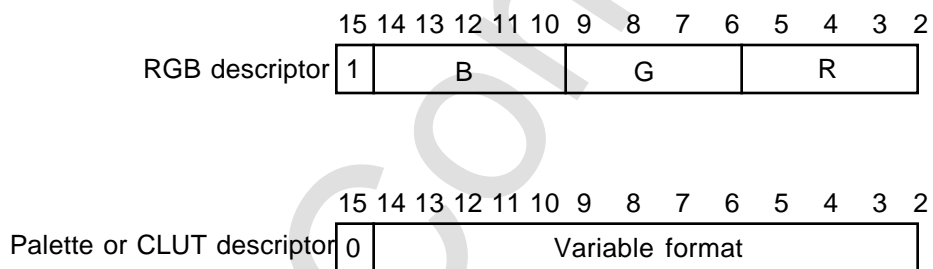


Fig. 2-3 Sprite pixel descriptors

The bits in a palette descriptor that are copied from the control word can specify the descriptor's format and, depending on the format, the pixel's priority and information related to color calculation. The bits in a CLUT descriptor specify the CLUT's base address, which VDP 1 adds to the 4-bit offset in the sprite bitmap to locate an entry in that CLUT. The entry in the CLUT can be either a 16-bit RGB descriptor or a complete palette descriptor. If it is a palette descriptor in a format that permits priority settings, a CLUT entry may be used to specify the priority of an individual pixel as well as its color.

The ability to set priorities for individual sprites is a major advantage of using paletted colors. All sprites whose pixels are defined with RGB values have the same priority, which is determined by a register set in VDP 2. A sprite whose bitmap is defined using paletted colors can have its own priority as determined by the color control word in its command table. If you want to set the priorities for a single

sprite, for example if you want place the sprite in front of or behind a specific backgrounds, use paletted colors. Priority has no effect between parts; to place parts in front of other parts, you must draw them in sequence, backmost part first.

Color calculations

VDP 1 can perform the following color calculations on any part whose pixels are specified with RGB color:

- **Replace.** Writes new pixel data over any previously written pixel in the frame buffer.
- **Shadow.** Divides the RGB values previously plotted in the frame buffer under the specified pixels by two and writes the result, producing color half as bright.
- **Half luminance.** Divides the RGB values of the new pixel data by two and writes the result to the frame buffer, producing color half as bright.
- **Half transparency (or half translucence).** Divides the RGB values previously plotted in the frame buffer by two, divides the RGB values of the new pixel data by two, adds the results, and plots the sum, producing a semitransparent effect.
- **Gouraud shading.** Calculates color offsets for all of a part's pixels based on offsets for each of its vertexes that are specified in a Gouraud shading table in VRAM. VDP 1 interpolates the intervening color offsets across the part to produce a smooth gradation.
- **Half luminance/Gouraud shading.** Combines half luminance and Gouraud shading.
- **Half transparency/Gouraud shading.** Combines half transparency and Gouraud shading.

Changing and erasing the frame buffer

VDP 1's registers control various aspects of its operation, such as TV mode, interlace mode, plot trigger mode, fill data for erasing, and the frame buffer change mode. The frame buffer change mode determines the way VDP 1 changes and erases the frame buffers, which in turn determines how many frames are displayed per second.

When set to the automatic 1-cycle frame buffer change mode, VDP 1 erases each pixel in the active frame buffer after it is displayed and switches the frame buffers every 1/60 of a second for a display rate of 60 frames per second. Because the number of parts that VDP 1 can plot to a single frame is limited by the size and scaling of each part, the way colors are specified, the color calculations applied, and other factors, it may sometimes be necessary to plot more than once to the same frame buffer to display a large number of parts. You can do this by setting the frame buffer change mode register during the VBL interrupt to one of three manual modes (valid only for the next frame):

- **Erase (manual mode).** During the next cycle, VDP 1 erases each pixel in the active frame buffer after VDP 2 displays it, but doesn't switch the frame buffers.
- **Change (manual mode).** During the next cycle, VDP 1 doesn't erase pixels in the active frame buffer after VDP 2 displays them, but does switch the frame buffers.
- **Erase & Change (manual mode).** During the next cycle, VDP 1 erases each pixel in the active frame buffer after VDP 2 displays it, and also switches the frame buffers.

For example, if you want to plot to each frame buffer twice before displaying the frame (for display at 30 frames per second), you can set the frame buffer change mode as follows:

Change mode set during next VBL	Frame buffer 0	Frame buffer 1
Erase	Displays	Plots
Change	Displays and erases	Plots
Erase	Plots	Displays
Change	Plots	Displays and erases

You can return to displaying 60 frames per second at any time by setting the automatic 1-cycle mode, which only needs to be set once, or the Erase & Change manual mode, which needs to be reset for each cycle.

Rotating the entire frame buffer

You can set TV modes for VDP 1 that allow VDP 2 to read the frame buffer diagonally, in effect rotating the entire frame buffer. Pixels that lie beyond the frame buffer coordinates are treated as transparent. Clipping areas remain fixed with respect to the frame buffer, so they are also rotated.

You can't rotate the frame buffer with double interlace display or when the TV mode is set to high resolution or HDTV.

SEGA Confidential

Chapter 3: Overview of VDP 2

VDP 2 plots up to five backgrounds based on pattern name tables, character patterns, and other information in its VRAM. VDP 2 also includes 4 KB of color RAM that defines color data for use by both VDP 1 and VDP 2. Figure 3-1 shows the configuration of VDP 2.

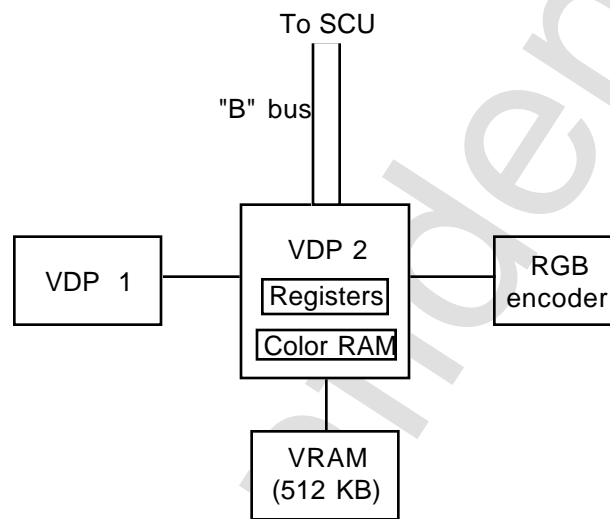


Fig. 3-1 Configuration of VDP 2

This chapter describes the kinds of backgrounds VDP 2 can plot, the mechanism it provides for accessing VRAM during the display interval, and some of the calculations it can perform as it displays each pixel. For detailed information about VDP 2, see the *VDP 2 User's Manual*.

Types of backgrounds

Four of the five backgrounds that VDP 2 can display at the same time are scroll planes. The picture in a **scroll plane** is larger than the TV display area and consists either of tiled cells or a single bitmap image with an RGB color assigned to each pixel. A **cell** for a scroll plane consists of the color data for an 8-by-8-pixel area, defined either as palette offsets or as RGB colors.

VDP 2 supports two kinds of scroll planes:

- A **normal scroll plane** supports vertical scrolling of cells, horizontal scrolling of lines, vertical and horizontal flipping of cells, and line zooming from 256x to 0.25x normal size.

- A **rotation scroll plane** supports two-axis rotation and scaling as well as vertical scrolling of cells, horizontal scrolling of lines, and vertical and horizontal flipping of cells. You can also display the contents of two pattern name tables in different windows within a single rotation scroll plane.

VDP 2 can display the following combinations of normal and rotation scroll planes:

- Up to four normal scroll planes and no rotation scroll planes.
- One or two normal scroll planes and one rotation scroll plane.
- No normal scroll planes and one or two rotation planes.

The **back plane** is the same size as the TV display area and therefore can't be scrolled. It is visible only when all other backgrounds are transparent. The back plane can display a single RGB color or a different RGB color for each line, but can't display paletted colors.

VRAM and the display interval

VDP 2 calculates and displays one pixel at a time based on the contents of the frame buffer, scroll plane data it reads from VRAM, priority settings for backgrounds and sprites, any color calculations to be performed, and various settings in its registers. When the TV screen is in normal mode, VDP 2 has an eight-cycle **display interval**: that is, it has eight clock cycles after the display of each pixel to get the scroll plane data it needs from VRAM to display the next pixel. When the TV screen is in high-resolution mode, VDP 2 has a four-cycle display interval.

You have complete control over how VDP 2 uses the available cycles in the display interval. VDP 2's VRAM consists of two 256-KB banks labeled VRAM-A and VRAM-B. You can optionally divide each of these into two 128-KB banks, for a total of four banks, each of which has a corresponding cycle pattern register as shown in Figure 3-2.

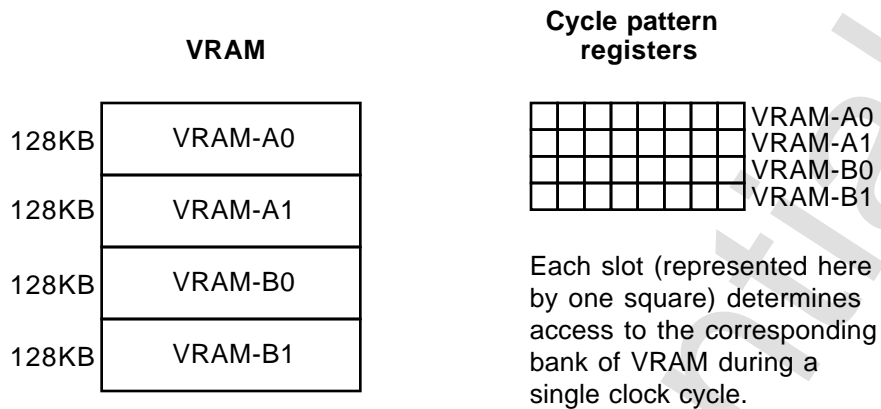


Fig. 3-2 Four banks of VDP 2 VRAM and their corresponding cycle pattern registers

VDP 2 can access all four banks simultaneously during each cycle of the display interval. Four eight-slot **cycle pattern registers** determine how VDP 2 uses the cycles it has available. You can set each slot in each register to specify that VDP 2 read a specific table in the corresponding bank of VRAM, provide read and write access to that bank from the SH-2 CPU, or not allow access the bank at all during that cycle. This means, for example, that you can calculate line scrolling for a single scroll plane without having to calculate it for all scroll planes.

The rules governing the use of cycle pattern registers are described in the *VDP 2 User's Manual*. For example, certain kinds of accesses must be performed at or before specific cycles in the display interval. If you are trying to do something complex, you may find that you need to use two VRAM banks and split up the accesses across two different cycle pattern registers. Similarly, because of the additional data and accesses required for a rotation scroll plane, you need to use two 128-KB banks and two cycle pattern registers to plot each pixel in the plane.

If you define a scroll plane using character patterns and paletted colors, the data in a VRAM bank consists, at a minimum, of a pattern name table, a character pattern table, and character patterns. If you define a scroll plane using RGB colors, the data in VRAM consists, at a minimum, of the bitmap data. In addition, a bank of VRAM may also include the following tables, depending on the kind of scroll plane and what you want to do with it:

- **Line scroll table.** Specifies coordinates and other information required for horizontal scrolling of lines.
- **Vertical cell scroll table.** Specifies coordinates and other information required for vertical scrolling of cells.
- **Rotation parameter table.** Specifies parameters for rotation scroll planes, including where in the pattern name table to begin the upper-left corner or edge of the rotation plane, matrix parameters that specify the degree of

rotation, viewpoint coordinates that determine the point from which the rotation is observed, and center coordinates that determine the point around which the plane rotates.

- **Coefficient table.** Specifies k coefficients used to determine the rate at which VDP II steps through the original pattern name table as it plots each pixel. For example, if the horizontal coefficient is 0.1, VDP II stretches out each pixel horizontally to occupy ten times its normal width. You can use coefficient tables to stretch or squeeze the plotting of one or more pixels vertically or horizontally to produce scaling, bowing, and other effects.
- **Line color screen table.** Specifies line colors for use in color calculations or to specify colors for the back screen.
- **Line window table.** Specifies horizontal start and end coordinates of lines that make up a window.

Character patterns and scroll planes

You can specify the colors for a single 8-by-8-pixel cell from among 16, 256, 1024, or 2048 palette entries or from 32,768 or 16,777,216 RGB colors. The amount of RAM required to specify each of a cell's pixels varies from 4 bits to 32 bits, depending on the number of colors. Cells are referenced through several levels of indirection from tables in VRAM that identify character patterns and collections of character patterns to be displayed as a scroll plane.

A **character pattern** consists of color data, defined as an entry in a **character pattern table**, for a square made of either one or four cells. Each character pattern is referenced from a **pattern name table**, which references all the characters for a 32-by-32- or 64-by-64-character **page**. References to pages can be grouped in **planes**, and references to planes can be grouped in a **map** that defines a single scroll plane. You can combine these references in various ways to build up a scroll plane as large as 8192 by 8192 pixels from just a few character patterns occupying a small portion of VRAM.

One entry in a pattern name table identifies the address of a character pattern in a character pattern table. Depending on the way the character pattern's colors are specified, the pattern name table may also identify the address of a palette in color RAM and additional information about horizontal or vertical flipping and priority and color calculations.

A palette in VDP 2's color RAM consists of 16, 64, 128, or 256 15- or 24-bit RGB values. If you use 15-bit color, each palette entry occupies 16 bits and you can specify a total of 2048 entries. If you use 24-bit color, each palette entry occupies a

long word (4 bytes) even though the color data itself takes up only 3 bytes. Thus, using 24-bit color wastes 1 byte of RAM per palette entry and restricts the total number of entries to 1024.

Scroll plane display

To display one or more scroll planes, you load data into VRAM, initialize VDP 2's registers, and set the cycle pattern registers as necessary during each display interval. Initialization includes setting the scroll rotation matrix parameters, the color mode register (for either 15-bit or 24-bit color), and the priority registers and clearing the color calculation and color offset registers.

To scroll a scroll plane vertically or horizontally, you can set registers that determine the starting point within the pattern name table for display of the upper-left corner of the plane on screen. For continuous scrolling, you should reset these values during each VBL interrupt.

If the pattern name table is larger than you can fit into VDP 2's VRAM, you can load the portion that fits and then replace it with another portion that is shifted in the appropriate direction through the original table.

Scaling and rotation

For each rotation scroll plane, you must provide a table of k coefficients in addition to a pattern name table and a character pattern table. The coefficient table determines the rate at which VDP II steps through the original pattern name table as it plots each pixel, thus allowing for vertical or horizontal scaling and special effects. Each coefficient is a 4-byte representation of a decimal value and applies to a line, groups of pixels, or (potentially) a single pixel.

To achieve smooth effects, you should replace the k coefficients each time VDP 2 displays one frame. Because VDP 2 needs continuous access to the k coefficients to display a rotation scroll plane accurately, you should normally load new k coefficients during the VBL interrupt, either just before or just after you swap frame buffers. It's also possible to load new coefficients during the HBL interrupt.

The way VDP II plots a rotation scroll plane also depends on scroll rotation parameters set in VDP 2's registers. These include the following:

- The rotation transformation matrix determines the degree of rotation.
- Perspective point coordinates determine the point from which the rotation is observed.

- Rotation center coordinates determine the point around which the scroll plane rotates.

You can rotate a rotation scroll plane on the x axis, on the x axis and z axis, on the y axis, or on the y and z axis, but not on the x axis and y axis simultaneously.

Priority functions

VDP 2 determines the priority for the display of any given pixel according to priority settings for each part that uses paletted colors, a single priority setting for all parts that use RGB colors, and a single priority setting associated with each background. For example, if you are displaying a sprite and two backgrounds, one showing trees in front of a main background showing a landscape, you can set priority bits for the sprite's pixels so that Saturn displays the trees in front of the main background and the sprite between the backgrounds. You can then bring the sprite to the foreground by changing the priority settings for all its pixels.

You can also make any portion of a scroll plane transparent to any background underneath it by specifying 0 as the color for those pixels.

Priority has no effect between parts; to place parts in front of other parts, you must draw them in sequence, backmost part first.

Color processing

VDP 2 can apply several types of color processing:

- **Color calculations** make it appear that sprites or backgrounds are partially transparent to each other. The calculations can be performed using up to 32 ratios and make use of the priority settings for the sprite or background pixels involved. VDP 2 color calculations are useful for making sprites and backgrounds appear or disappear gradually.
- **Color offsets** allow you to average, add, or subtract the color values for two backgrounds. This can be useful for dissolves and related effects.
- **Shadowing** involves creating a shadow on a sprite or a background with the aid of a shadow sprite. This works by making the shadow sprite transparent and dividing the RGB values of the sprite or background below it in half.

Chapter 4: Developing for Saturn

This section describes some of the software and hardware tools for Saturn game development that are currently available or will soon be available. Other tools, not discussed here, will also be available later in 1994.

Graphics tools

Sega provides several tools for converting image formats to a form you can use in your program. Three tools are currently available:

- **SCONVERT** converts a PICT or PCX file to a format you can use to specify a sprite.
- **BRIP** converts a PCX file to a format you can use to specify a background.
- **SaturnSp_C** is a PhotoShop or Debabelizer plug-in module that converts any file the Macintosh version of PhotoShop can read to a format you can use to specify a sprite.

Two additional tools are under development:

- **SaturnBRIP** is a PhotoShop or Debabelizer plug-in module that converts any file the Macintosh version of PhotoShop can read to a format you can use to specify a background.
- **3DS2SAT** converts a 3-D model created in 3D Studio to a standard Saturn format called SAT3. You can then extract the information you need from the SAT3 file for your program.

The sections that follow describe the graphics content tools in more detail.

SCONVERT

The sprite ripper, or SCONVERT, is a PC/MS-DOS command-line utility that reads a PICT or PCX file and converts it to an array of 15-bit RGB values in the form of a C header file you can compile directly into your program. The header file defines an array of 16-bit pixels. When necessary at run time, your program copies the array to an address in VDP 1's VRAM so that you can specify it as a sprite bitmap.

BRIP

The background ripper, or BRIP, is a PC/MS-DOS command-line utility that reads a PCX file and converts it to a C header file you can compile directly into your program. In the process, BRIP eliminates duplicate character patterns, including vertical or horizontal mirror images. The PCX file can specify either paletted colors or up to 256 RGB values. Unlike SCONVERT, BRIP outputs three arrays: an array of 8-bit character patterns (a character pattern table), an array of references to those character patterns (a pattern name table), and an array that defines a single palette used by the character patterns. (BRIP doesn't yet support back screens or enforce the nested references to cells, pages, and planes described in Chapter 3.)

SaturnSp_C PhotoShop/Debabelizer module

The SaturnSP_C PhotoShop/Debabelizer module converts any Macintosh PhotoShop or Debabelizer file to a C header file that you can compile directly into your program as a sprite bitmap. The file to be converted can specify either RGB color or paletted color. If the file specifies RGB color, SaturnSP_C outputs an array of 16-bit RGB color values. If the file specifies paletted color, SaturnSP_C outputs two arrays: an array of 8-bit indices into the palette and an array that defines a 256-entry palette.

SaturnBRIP PhotoShop/Debabelizer module (under development)

The SaturnBRIP PhotoShop/Debabelizer module converts a Macintosh PhotoShop or Debabelizer file to a C header file you can compile directly into your program. The file to be converted must specify paletted color. SaturnBRIP outputs three arrays: an array of character patterns (a character pattern table), an array that defines an array of references to those character patterns (a pattern name table), and an array that defines the palette used by the character patterns.

3DS2SAT (under development)

To simulate 3-D animation on other game machines, you first create a model with 3-D software tools such as 3D Studio and use that model to export a series of bitmapped images. You then display the bitmapped images in a series of frames to create the illusion of movement in three dimensions. Each bitmapped image takes up a lot of memory and you must be able to predict all potential movements of an object in order to provide the appropriate bitmaps.

To create 3-D animation on Saturn, you can use 3DS2SAT to convert the entire 3-D model to the SAT3 format, the standard Saturn file format for 3-D information. For example, suppose you want to display a rotating cube. The 3-D model consists of six four-sided polygons, each of which is filled with a bitmapped image. When you

need to display the object from a particular vantage point, Saturn can calculate transformations for the vertices of the six polygons in 2-D space, distort the bitmaps for each polygon accordingly, and display the resulting image on the screen—all at run time at 30 or 60 frames per second. Thus, instead of limiting potential motion to preselected views of a 3-D model, you can orient it any way you want and update each frame at run time.

Each object defined by a SAT3 file includes an array of vertex coordinates for that object, an array that defines the object's faces by specifying their vertices, and a variety of other information such as vertex normals used in Gouraud shading. This arrangement allows you to perform a matrix transformation on the vertex coordinates before associating those vertexes with specific faces, thus avoiding repeated calculations for shared points.

Depending on the nature of the game you are programming, you may not need all the information provided by the SAT3 file format. For example, if you don't intend to apply Gouraud shading to an object, you don't need to include that information in your program. In most cases you should create a third file from the SAT3 file that includes only the information you need for your program.

Programming tools

Sega provides a standard GNU C compiler that has been modified to work with the SH-2 CPU. GNU supports registerized parameters—that is, it allows you to refer to up to four registers directly; other parameters are on the stack.

Hitachi provides an assembler, a C compiler, a link editor, and a librarian that you can use to compile and link assembly and C files. You can also use other third-party assemblers to compile and link assembly files. Typically, these assemblers provide a debugger, a linker, and a command-line DOS interface that requires a DOS extender.

If you wish, you can write some parts of your program in C using the GNU compiler and other parts in assembly language using a third-party assembler, then link both the GNU C modules and the assembly files and output C files in the COFF file format.

Whether you program in C, assembler, or both, you can use a variety of software debuggers with the CartDev to debug your program, as shown in Figure 4-1.

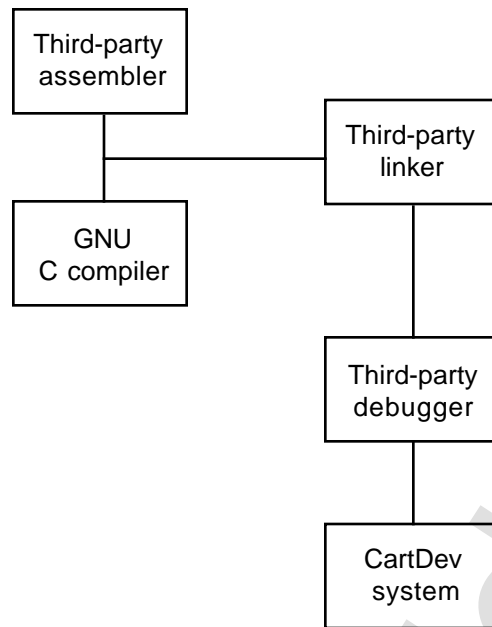


Fig. 4-1 Using GNU with third-party tools and the CartDev system

Alternatively, you can use Hitachi software with an E7000 series in-circuit emulator (ICE) as a debugging system. Unlike software debuggers used with the CartDev, ICEs provide a history display and can deal with more esoteric problems such as interrupts or anything that requires strict tracing of code. In some cases it may be desirable to use both the CartDev and an ICE at the same time for different purposes.

To learn about the way the SH-2 handles pipelines and optimize for RISC architecture, you can load files into the Sherry simulator, an SH-2 simulation program that runs on a PC.

The sections that follow describe the CartDev, the Hitachi 7000 series ICEs, and the Sherry simulator in more detail.

CartDev system

The CartDev is a low-cost development system that plugs into Saturn's cartridge port. It has its own controller that handles all communication. A SCSI (SCSI 2) interface permits communication with the CartDev from an IBM PC, a Macintosh, or other hosts, such as SGI workstations. The CartDev communicates with Saturn through dual port RAM. It also includes dual-access emulation RAM (RAM that can be read or written to from either the CartDev or from Saturn) that includes 64 KB of static RAM, with room for up to 8 MB of optional DRAM.

In addition to a SCSI interface, the CartDev provides two RS-232 or RS-485 ports and one in-and-out high-speed parallel port for an auxiliary interface. Currently, the auxiliary interface consists of a sound adapter with a digital audio interface and two sets of in, out, and through MIDI ports that can handle up to 32 voices. Other auxiliary interfaces may be developed in the future.

The CartDev provides some hardware control (such as resets, NMIs, and interrupts) and, when used with appropriate debugging software, allows you to set software breakpoints, single step through code, and read and write to memory.

In addition to using the CartDev system for software debugging tasks, you can use it to download or upload art or sound files. In general, downloads with the CartDev are ten or twenty times faster than downloads with the E7000 ICEs. The CartDev has an open interface that allows third parties to use its communication capabilities for additional game development tools.

The Hitachi E7000 ICEs

Hitachi currently provides three E7000 series in-circuit emulators (ICEs) that have similar hardware debugging capabilities and can emulate the SH-2 processor. Each version of the E7000 ICE knows about all internal workings of the processor on a cycle-per-cycle basis (including pipelining), and maintains a history of previous instructions.

These are the E7000 ICEs that are currently available:

- The E7000 includes a floppy drive and emulation RAM, and it has room for up to 4 MB of expansion RAM. It is controlled through Ethernet or Cheapnet from a host with telnet capabilities.
- The E7000PC has the same capabilities as an E7000 except that it doesn't include a floppy drive and is controlled by a PC via a proprietary parallel interface card.
- The E7000 Eval Board doesn't include a floppy drive and has 512 KB of nonexpandable emulation RAM. It is controlled from a PC through the same proprietary parallel interface card used with the E7000PC.

You can control any of these ICEs from a personal computer, if necessary at the same time as the CartDev. The E7000 ICEs can be helpful with low-level problems such as interrupts or crashes that corrupt the system, or with any problem that you can't isolate using a software debugger.

Sherry SH-2 simulator

The Sherry simulator allows you to examine the SH-2's pipelines. It consists of a program you can run on a PC without any connection to Saturn or any other hardware. If you load an S record into the Sherry program, it allows you to set breakpoints and single step through code using a memory map that emulates the SH-2 memory map. This is especially valuable for observing clock cycle counts for pipelined RISC instructions and for learning how new instructions and the ordering of instructions affect the pipeline.

Sherry has a built-in assembler and debugger and simulates all SH-2 instructions. It only uses as much of the PC's memory as your program requires, even if you are addressing a larger memory space. It doesn't simulate the cache.

High-level languages vs. assembler

You can use assembly language or high-level languages such as C to write Saturn programs. Because the SH-2 chips are RISC, assembly-language programming for Saturn can be more complex than for other game machines. For example, you must deal with pipelining if you write in assembly language. This means development in assembly language may take longer than development in a high-level language.

Although the GNU C compiler provided by Sega may produce less efficient code in some circumstances than an assembler, it takes care of many of the complexities of RISC programming, such as pipelining, automatically. Hand-crafted assembly code can be useful in situations where slight performance improvements are significant, but compiled C code works just as well for many tasks.

Documentation

This section lists the most important documents you will need as you begin writing programs for Saturn.

General Saturn documentation

Boot ROM Specification, ST-079B

Macintosh Micon Soft Development Specification, ST-066

Programming Tools, ST-80-R1

Saturn File System Library Specification, ST-39-R2

Saturn Outline, ST-30-R1
Saturn Stream System, ST-098

Software Library User's Manual, ST-078

Hitachi documentation

E 7000 Primer

HS 7000 E101SE IBM PC/IF

SH 2 Hitachi Tools

SH 7000 Pipeline Operations

SH 7000 Programming Manual

SH 7030 E7000 GUI User's Manual

SH 7030 Emulator Model 1 User's Manual

SH Electrical Characteristics

SH Series C Compiler

SH Cross Assembler User's Manual

SCU documentation

SCU User's Manual, ST-097

CD-ROM subsystem documentation

Saturn CD Communication Interface Specification, ST-38-R2

Saturn CD System, ST-40-R1

Virtual CD System User's Manual, ST-100-R1-A

Video subsystem documentation

VDP 1 Constraint Items, ST-089

VDP 1 Manual, ST-13-R2

VDP 2 User's Manual, ST-58

Sound subsystem documentation

Saturn Standard MIDI File Converter Specification, ST-66

SCSP User's Manual, ST-77-R1

SCSP Wave Edit Tool Specification, ST-067

SCSP/DSP Linker Instruction Manual, ST-70-R1

Sound Development Manual Outline, ST-81-R2

Sound Edit Tool Specification, ST-068

Sound Editor User's Manual, ST-101

Sound Programming Debugger User's Manual, ST-065-R1

Wave Editor User's Manual, ST-99

Glossary

back plane A background that's visible only when all other backgrounds are transparent.

cache address Address accessed via the cache controller.

cache through address Address accessed directly, without checking the cache.

cell Color data for an 8-by-8-pixel area, defined either as palette offsets or as RGB values.

character pattern A square made of one or four **cells**.

character pattern table A block in VDP 2's VRAM that contains the character patterns for a scroll plane.

clipping Plotting parts only inside or outside a designated region; the area not plotted is "clipped."

color control word A word in the **command table** that specifies either a single color for a part or information that VDP 1 combines with the offsets specified in a sprite bitmap to obtain a **pixel descriptor**.

command table A 32-byte block in VDP 1's VRAM that specifies a command, a jump mode, and other information VDP 1 requires to execute the command.

cycle pattern registers Four eight-slot registers that determine how VDP 2 uses the cycles in the **display interval** to access its VRAM.

display interval The four- or eight-cycle interval after VDP 2 displays one pixel and before it displays the next. You have complete control, via the **cycle pattern registers**, of the way VDP 2 uses the cycles available in the display interval to access its VRAM.

display list A list of commands in VRAM that tell VDP 1 what to plot for a single frame.

distorted sprite A **textured part** that behaves like a normal sprite and can also be rotated and distorted by specifying the coordinates of four corner points.

line A **nontextured part** specified by two points that can be filled with a single color.

map A group of **pages** that defines a scroll plane.

meshing Plotting every other pixel of a part to produce a fine checkerboard pattern that simulates transparency.

nontextured part A part to which VDP 1 can apply a single color.

normal scroll plane A scroll plane that supports horizontal scrolling of lines, vertical scrolling of cells, vertical and horizontal flipping of cells, and line zooming from 256x to 0.25x of normal size.

normal sprite A **textured part** that can be flipped horizontally or vertically.

page A group of **character patterns** arranged in a 32-by-32-cell or 64-by-64-cell square.

part A **textured part** or a **nontextured part** plotted by VDP 1.

pattern name table A table of references to all the characters for a **page**.

pixel descriptor A 16-bit word that VDP draws to the frame buffer for each pixel. If the high bit is set to 1, the remaining 15 bits specify an RGB value. If the high bit is set to 0, the pixel descriptor consists of information from the **color control word** combined with an offset in the sprite bitmap, and it specifies either an entry in a color lookup table or a palette entry in VDP 2's color RAM.

plane A group of **pages** in arrangements up to 2 pages square.

polygon A **nontextured part** specified by four points that can be filled with a single color.

polyline A **nontextured part** specified by four points. Its outline (but not the area it encloses) can be drawn with a single color.

rotation scroll plane A scroll plane that supports two-axis rotation and scaling as well as horizontal line scrolling, vertical scrolling of cells, and horizontal and vertical flipping of cells.

scaled sprite A **textured part** that behaves like a normal sprite and can also be magnified or reduced horizontally, vertically, or both horizontally and vertically.

scroll plane Background plotted by VDP 2 using either RGB values or palette colors.

sprite See **textured part**.

textured part A part plotted by VDP 1 that has three or four vertices and is filled with a bitmapped image; also called a **sprite**.