
Visual Memory Simulator Manual Version 1.02



Dreamcast™

Introduction

Thank you for cooperating in the development of applications for Sega hardware.

This manual explains how to install and use the Visual Memory Simulator, which emulates, through software, the operation of a "Visual Memory" memory card for Dreamcast.

Contents of This Manual

Chapter 1 Overview

This chapter provides an overview of the Visual Memory Simulator, and describes its main features.

Chapter 2 Implemented Devices

This chapter describes the devices that are included in the Visual Memory Simulator.

Chapter 3 Basic Operation

This chapter describes the procedures for starting up the Visual Memory Simulator, and for loading and executing applications.

Chapter 4 Descriptions of Windows and Panels

This chapter describes the functions and use of the Visual Memory Simulator's various windows, panels, and menus.

Chapter 5 Networking

This chapter describes how to connect Visual Memory Simulators through a TCP interface.

Chapter 6 Related Files

This chapter describes files that the Visual Memory Simulator references.

Chapter 7 Warning Messages

This chapter describes the warning messages that may be displayed when an application is run on the Simulator.

Note:

This manual is written for "Visual Memory Simulator Version 1.01" (Sega Enterprises). The second edition includes revisions that reflect changes that were made in "Visual Memory Simulator Version 1.02."

The specifications of Visual Memory and the contents of this manual are subject to change without notice.

Sega Enterprises cannot bear any responsibility for any harm resulting from the use of Visual Memory or this manual.

Trademarks

"Sega," "Dreamcast," and "Visual Memory" are trademarks of Sega Enterprises.

"Microsoft," "Windows," and the Windows CE logo are registered trademarks of Microsoft Corporation in the U.S. and other countries.

Other product names that appear in this manual are trademarks or registered trademarks of their respective owners.

The TM and ® symbols are not used in this text.

Revision History

August 31, 1998	First Edition
November 30, 1998	Second Edition
Copyright © 1998 Sega Enterprises	
Edited and Published by Ascii AAP Publications Editing Department	

Relationship To Other Manuals

Visual Memory-related Manuals

Visual Memory Hardware Manual

This manual is a technical reference for the Visual Memory hardware and the system BIOS specifications. The Overview at the beginning of the manual includes a simple summary of the Visual Memory specifications.

Game designers should read the Overview, and programmers should read the entire manual.

Caution

The "Visual Memory Simulator Guide" (this manual) assumes an understanding of the Visual Memory hardware. Developers who are developing software for Visual Memory for the first time should first read the "Visual Memory Hardware Manual."

Visual Memory Programmer's Guide

This manual explains how to install and use the Assembler for the Sanyo Electric LC86700, which is installed in the Visual Memory unit, the Linker, the Library Manager, and "E2H86K.EXE", which creates execution files for visual memory. This manual also explains the LC86700 instruction set and the Assembler syntax.

Refer to this manual for the installation procedure for the Visual Memory SDK, beginning with the Visual Memory Simulator.

Visual Memory Tutorial

This manual describes the procedure for developing applications for Visual Memory and the method for transferring applications to Visual Memory. This manual also includes sample programs with detailed comments.

Restrictions

Differences in Operating Speeds

Because the Windows system idle is used for the operating clock of the virtual CPU, the operating speed changes according to the specs of the computer that is being used. The operating speed of the virtual CPU can be adjusted to a certain extent in the Environment Settings Window. On the other hand, delay times such as the startup characteristics from the clock stopped state are ignored.

The operating speed also affects timer 0, timer 1, and the base timer. As with the virtual CPU, the system idle is also used for the operating clock that is supplied to the internal virtual timers.

The timer interrupt cycle will not serve as an accurate clock, even if the parameters are set to correspond with actual time.

Differences in Devices

Although devices are designed to operate in an equivalent manner from the standpoint of the application, there are slight differences in the actual hardware and operation timing. However, such differences do not have fatal impact in regards to the production of applications.

Audio Output

Visual Memory audio is output by using the PWM function. This function operates according to an internal timer in Visual Memory. Because the operation speed of this timer in the Visual Memory Simulator differs from the that of the timer in the actual machine, proper audio output is not possible. PWM output should be checked on the actual machine.

Differences in the Operating Speeds of the LCDs

Because the dots on the virtual LCD are drawn through graphics, the operating speed of the actual hardware cannot be achieved.

Maple Bus Serial Functions Not Implemented

No simulation for bus-type serial circuits has been implemented. Although this area can be accessed, there are no functions to execute, so doing so is invalid. Simulation functions related to the communications buffer memory have been implemented, however.

Visual Memory Simulator Version

This manual is based on Visual Memory Simulator Version 1.02.

Technical Support

If, in the course of your development work, you have any technical questions, encounter hardware difficulties, need information that is not included in this manual, or experience a hardware malfunction, contact technical support at: Sega Enterprises

Technical Support Center

1-2-12, Haneda, Ota-ku, Tokyo 144-8531

Japan

Tel: 03-5736-7355

Fax: 03-5736-5357

E-mail: katana@sft.sega.co.jp

contents

Introduction	2
--------------------	---

Relationship To Other Manuals	4
-------------------------------------	---

Restrictions	5
--------------------	---

Technical Support	6
-------------------------	---

Chapter 1

Overview	11
----------------	----

1.1 Features	11
--------------------	----

1.2 Visual Memory Simulator Operating Environment	11
---	----

1.3 Checking Operation on Actual Visual Memory Hardware	12
---	----

1.4 Notes Concerning Startup for the First Time	13
---	----

Chapter 2

Implemented Devices

17

2.1	Virtual CPU	17
2.2	Memory	18
2.3	LCD Controller (LCDC)	19
2.4	Serial Interface (SIO)	19
2.5	Timer	19
2.6	Interrupt Controller	19
2.7	I/O Ports	19
2.8	External Input Devices	20

Chapter 3

Basic Operation

21

3.1	Starting Up the Visual Memory Simulator	21
3.2	Loading the System BIOS	21
3.3	Loading and Executing Applications	22
3.4	MAP File	22
3.5	Drag & Drop	23

Chapter 4

Descriptions of Windows and Panels

25

4.1 Main Window	26
4.1.1 Menus	26
4.1.2 Toolbar	29
4.1.3 CPU Register Display Function	30
4.1.4 Execution Control	31
4.1.5 Disassembly Function	32
4.1.6 Visual Memory Image	33
4.1.7 Status Lamp	33
4.1.8 Changing the Size of the Main Window	34
4.1.9 System Console	34
4.2 Memory Control Window	35
4.2.1 RAM#0, RAM#1	36
4.2.2 FLASH#0	38
4.2.3 XRAM	39
4.2.4 SFR	40
4.2.5 VTRBF	41
4.3 Break Control Window	42
4.3.1 Break by Breakpoint Address Comparison	42
4.3.2 Display When an Interrupt Is Received	45
4.3.3 Access Reference Monitor	46
4.4 Special Function Register Control Window	47
4.4.1 CPU Control	48
4.4.2 LCD	49
4.4.3 INT Control	50
4.4.4 Timer 0	51
4.4.5 Timer 1	52
4.4.6 SIO	53
4.4.7 PORT1	54
4.4.8 PORT3/7	55
4.4.9 External INT	56
4.4.10 VMS Special	57
4.4.11 Base Timer	58

4.5 LCD Snapshot Window	59
4.5.1 Description of Tool Bar Buttons	59
4.5.2 Display by STAD Checkbox	60
4.5.3 Menus	60
4.6 Network Monitor Window	60
4.7 Trace Panel	62
4.8 Hexadecimal Input Pad	64
4.9 Environment Settings Window	66
4.9.1 Settings	66
4.9.2 Work Settings	68
 Chapter 5	
Networking	71
<hr/>	
 Chapter 6	
Related Files	73
<hr/>	
6.1 System Files	73
6.2 Application Files	74
 Chapter 7	
Warning Messages	75
<hr/>	

Chapter 1 Overview

The Visual Memory Simulator is a virtual machine system that simulates the Visual Memory hardware. This system can execute, without any special additional processing, programs that were developed for Visual Memory.

1.1 Features

- The Visual Memory Simulator implements almost all of the hardware in the Visual Memory System through software.
- The status of CPU registers and memory can be displayed during execution.
- Program execution can be traced.
- The status of the Special Function Registers can be displayed.
- The Visual Memory Simulator supports debugging functions such as breakpoints and memory fetch breaks.
- Two Visual Memory Simulators can be connected through a network.

The Visual Memory Simulator is composed of several virtual devices, the central one being a virtual CPU.

The virtual CPU is designed as an interpreter. Execution files are fetched and executed one instruction at a time. Because peripheral devices are also implemented in the Virtual Memory Simulator, roughly 100% of a program can be checked. Furthermore, no special programs or hardware are needed in order to load execution files into the Visual Memory Simulator.

Because the Visual Memory Simulator is almost entirely software based, there are differences in the operating speed of the Visual Memory Simulator versus actual Visual Memory. Checks that require the actual speed, such as checks of operation timing and sound, should be performed on the actual hardware. The purpose of the Visual Memory Simulator is to verify the program logic; the use of the Visual Memory Simulator in the development cycle should be differentiated from that of the actual hardware.

1.2 Visual Memory Simulator Operating Environment

Table 1-1

CPU	Pentium 150MHz or higher recommended
RAM	At least 32MB recommended
OS	Windows 95
HDD free space	At least 5MB
Colors	At least 256 colors

Resolution	At least 1280 × 1024 recommended
------------	----------------------------------

Note

Because the Visual Memory Simulator is a Windows application, it can basically run on any CPU on which Windows 95 is running. However, the virtual CPU is a type of interpreter, and when the operation of other virtual devices is also included, the Visual Memory Simulator can run slowly. In order to run at a reasonable speed, a PC with a fairly fast clock speed is required.

1.3 Checking Operation on Actual Visual Memory Hardware

The Memory Card Utility, which is provided with the Visual Memory SDK, is used to transfer an application that has been developed into Visual Memory. The Memory Card Utility is a utility that is used to transfer Visual Memory applications between a PC and the Dev.Box, and between the Dev.Box and Visual Memory.

The Memory Card Utility is located in the "Utility" folder in the folder where the Visual Memory SDK was installed, as an ELF file that runs on the Dev.Box. For details on how to use the Memory Card Utility, refer to the "Visual Memory Tutorial." After the logic in an application has been checked by using the Visual Memory Simulator, be certain to check the timing and operating speed of the application on the actual hardware.

The following environment is needed in order to run the Memory Card Utility:

Items provided by Sega:

- Dreamcast SDK
- CodeScape (including DA Checker)
- GD WorkShop
- Dev.Box (Set 5.2X or later)
- Dreamcast controller
- Visual Memory

Items That Must Be Obtained Separately

- RS-232C cross cable
- Communications program that runs under Windows

1.4 Notes Concerning Startup for the First Time

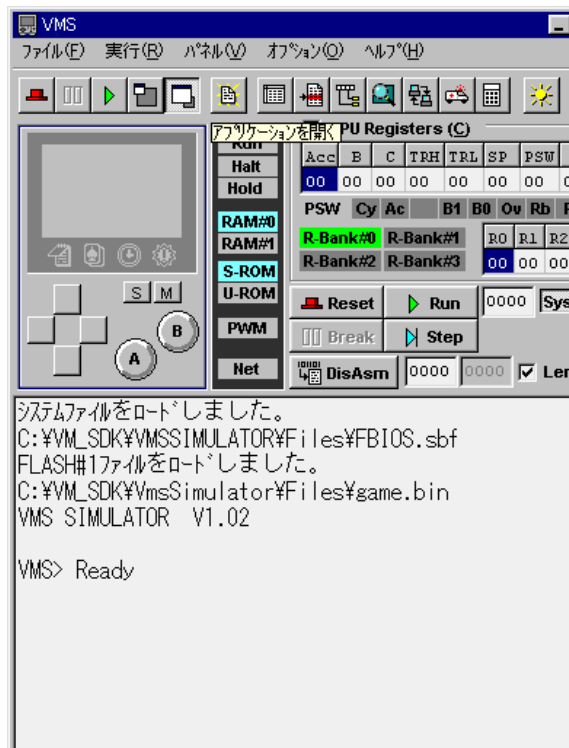
When the Visual Memory Simulator is started up for the first time, the contents of flash memory bank 1 are undefined, so the Visual Memory Simulator may indicate that "Visual Memory has not been formatted."

The operation described below must be performed the first time that the Visual Memory Simulator is started up.

Caution

Be certain to perform the procedure described below when starting up the Visual Memory Simulator for the first time after installing the Visual Memory SDK.

- ⌘ Execute the Visual Memory Simulator.



- 3 From the [File] menu, select [Open FLASH#1 Memory].
The following screen appears:



From the "Files" folder, select "GAME.BIN" and then click the [Open] button.

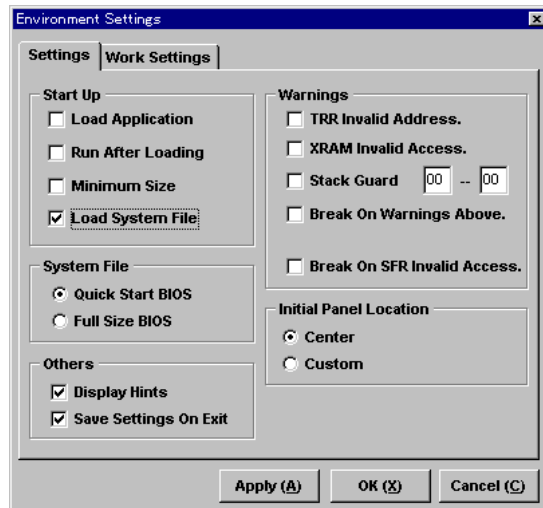
Caution

"GAME.BIN" contains the memory image for flash memory bank 1 in Visual Memory. Because this file includes the FAT information and the system management information, if this information is not found in flash memory bank 1, Visual Memory will be recognized as not having yet been formatted.

- ✎ Once "GAME.BIN" is loaded, the following screen appears:



- ⑧ From the [Option] menu, select [Environment Variables]; the following screen appears:



- ⊗ Make the settings described below in the dialog box that appears. All of these items are displayed under the [Settings] tab.
In the [Start Up] group, click the [Load System File] checkbox so that the box is checked.
In the [System File] group, select the [Quick Start BIOS] option.
After you have made all of the settings, click the [OK] button.
- ⊕ The display returns to the Visual Memory Simulator screen.
From the [File] menu, select [Exit] to quit the Visual Memory Simulator.

Caution

Be certain to quit the Visual Memory Simulator.

When you have completed the above procedure, the Simulator will recognize Visual Memory as having been formatted.

The following devices are implemented in the Visual Memory Simulator:

- Virtual CPU
- Memory
- LCD controller
- Serial interface
- Timer
- Interrupt controller
- I/O ports
- External input devices

2.1 Virtual CPU

A CPU interpreter, called the "virtual CPU," and which executes the Sanyo Electric LC86 Series instruction set, is implemented in the Visual Memory Simulator. This virtual CPU executes binary code that is stored in the memory area in the same manner as the actual CPU. There is no need to add special programs for the Simulator.

The Windows system idle is used as the operating clock for the virtual CPU. "n" instructions are executed per idle. The number of instructions that are executed per idle can be set in the Environment Settings Window. Adjust this value according to the clock speed of the PC that you will be using. for a detailed description of how to make this setting, refer to [Settings] - [[CPU Loop Count](#)] in section 4.9, "Environment Settings Window." However, if this value is increased, the timing by which Windows messages are acquired becomes skewed, with the result that the response of buttons, etc., becomes sluggish.

2.2 Memory

The Visual Memory Simulator simulates all memory areas in the Visual Memory System.

RAM Area

Bank 0	00H to FFH (256 bytes)	System work area
Bank 1	00H to FFH (256 bytes)	User work area

ROM Area

This area stores the OS program and the system application. This area cannot be manipulated by the user.

Flash Memory

Bank 0	0000H to FFFFH (64K bytes)	User program area
Bank 1	0000H to FFFFH (64K bytes)	Backup memory area

Work RAM

This is a buffer for communications with Dreamcast that can be accessed through the Special Function Registers.

If there are no communications with Dreamcast, this area can be used as RAM by an application.

VTRBF	0000H to 01FFH (512 bytes)
--------------	-----------------------------------

XRAM

This is the LCD display memory. This memory is equivalent to video memory in a typical PC.

This memory consists of three banks. Two banks are allocated to bitmap display, and one bank is allocated for icons.

Bank 0	0180H to 01FBH (96 bytes)	Includes unused areas.
Bank 1	0180H to 01FBH (96 bytes)	Includes unused areas.
Bank 2	0180H to 0185H (6 bytes)	

In addition to direct access by the virtual CPU, these memory areas can be edited through the Memory Control Window.

2.3 LCD Controller (LCDC)

The LCDC is designed to operate in an equivalent manner to the actual hardware from the standpoint of the CPU. The LCDC is accessed through the LCD-related Special Function Registers.

If data is written to XRAM while the LCD is ready for display, the data appears in the drawing area.

2.4 Serial Interface (SIO)

The serial interface is used to connect two Visual Memory units. A Visual Memory unit has two SIOs. SIO0 is allocated for sending, and SIO1 is allocated for receiving. Together, full-duplex communication is implemented through these two interfaces.

In the Simulator, Visual Memory units are connected by using TCP communication for communications between SIOs.

Only the Special Function Registers for the SIOs are visible from the standpoint of the virtual CPU; the network is hidden. Connection control is implemented through the [Network Monitor] command on the [Panels] menu.

2.5 Timer

Visual Memory has three timers. The Simulator supports timer 0, timer 1, and the base timer.

The Simulator supports all interrupts that are generated by the timers.

Timer Restrictions

The counter function based on external input to timer 0 is not supported in the Simulator. Although pulses can be generated through timer 1 PWM output, no sound is actually output.

Operating Clock for Timers

As is the case for the virtual CPU, the Windows system idle is used for the clock that is supplied to the timers. As a result, the actual speed at which the timers operate is different from that of the actual hardware.

2.6 Interrupt Controller

The LC86 Series supports interrupts with variable priority levels and nested interrupts. The Visual Memory Simulator simulates interrupt operations in the same manner. There are no restrictions concerning interrupts. Both internal and external interrupts are supported.

2.7 I/O Ports

There are three I/O ports: port 1, port 3, and port 7.

Port 1	Assigned to SIO and PWM output ports.
Port 3	The Visual Memory buttons are connected to this port.
Port 7	The voltage detection and other VMS detection signals are connected to this port.

2.8 External Input Devices

Buttons Connected to Port 3

Visual Memory has eight buttons that serve as input devices. These buttons are connected to port 3. The current status (pressed/not pressed) of each button can be detected by reading this port. When a button is not being pressed, the corresponding signal is high; when a button is being pressed, the corresponding signal is low. Port 3 interrupts are also supported, so it is possible to simulate interrupts that are generated when a button status changes.

Starting from the most significant bit, the buttons assigned to the bits are: SLEEP button, MODE button, B button, A button, right button, Left button, Down button, Up button.

Control Signals Connected to Port 7

Bits 0 to 3 of port 7 are input signal ports for external interrupts. Interrupt control for external interrupts is specified through the IO1CR register and the I23CR register.

Four input signals are connected to port 7.

+5V Supply Signal as External Power Supply Connected to P70

When no external power supply is connected, this signal is low; when external power is supplied, this signal is high. This is simulated through the "+5V Test" checkbox connected to P70 on the SFR panel. When "ON," external power is being supplied.

Internal Battery Voltage Drop Signal Connected to P71

This signal is generated when the voltage of the internal battery drops. When this signal is high, the battery voltage is normal. When this signal is low, the battery voltage drops. This is simulated through the "Low Voltage Test" checkbox connected to P71 in the SFR panel. When "ON," the voltage is low.

Input Signals ID0, Connected to P72, and ID1, Connected to P73

These signals are normally low; they are high when an input is connected. This is simulated through the "ID0 Test" and "ID1 Test" checkboxes connected to P72 and P73 in the SFR panel.

Chapter 3 Basic Operation

This chapter explains the procedure for loading and executing application programs in the Visual Memory Simulator.

3.1 Starting Up the Visual Memory Simulator

Startup the Visual Memory Simulator either from the Windows [Start] menu, or directly from the folder where it was installed. Once the Simulator is started up, the Main Window is displayed and the Simulator begins waiting for input.

3.2 Loading the System BIOS

Right after the Visual Memory Simulator has been started up, the system ROM area is initialized. Because applications developed by users are called from the system BIOS, it is necessary to load the system BIOS first.

1. From the Visual Memory Simulator's [File] menu, select [Open System File].
2. Select the system BIOS file (SBF) to be loaded.
3. Click the [Open] button. The system BIOS is loaded into the internal system ROM.

"FBIOS.SBF" is the full-size BIOS; this program manages the system when Visual Memory is started up. "QBIOS.SBF" is the quick start BIOS; this BIOS can skip the clock setting that is requested when Visual Memory is reset.

Applications are called from BIOS and started up. A setting can be made in the Environment Settings window that automatically loads the system BIOS when the Visual Memory Simulator is started up. For details on making these settings, refer to [Startup Settings] - [[Load System File](#)] in section 4.9, "Environment Settings Window."

Caution

Quick start BIOS supports exactly the same functions as full-size BIOS, except that the clock setting can be skipped at startup.

3.3 Loading and Executing Applications

The application execution files that can be loaded into the Visual Memory Simulator are HEX files. The extension for such files is ".HEX" or ".H??".

Caution

The Visual Memory Simulator cannot load binary files (".BIN") created by H2BIN.

1. From the [File] menu, select [Open Application].
2. Select the application execution file (".HEX" or ".H??") to be loaded.
3. Click the [Open] button. The file is loaded into flash memory bank 0. The memory area where such files are loaded is fixed to "bank 0."

After the file has been loaded, click the Reset button; the Simulator virtual machine is reset and the CPU begins operating. As soon as the CPU begins operating, the system BIOS is executed.

Although the operations performed in the CPU's internal registers while the system BIOS is running can be checked, displaying the registers consumes CPU time, so the Simulator will run more slowly as a result. The display of these registers can be stopped in order to speed up processing.

To stop an application that is running, click the [Break] button. The values in the registers as of the moment when execution was stopped are displayed on the console, and operation stops. Furthermore, the program counter value for the next instruction to be executed is set in the text box where the execution address is stored.

To resume program execution, click the [Run] button. Click the [Step] button to step through the instructions one at a time.

3.4 MAP File

When a MAP file is in the same folder as the application, this file is loaded after the application. The extension for symbol files is ".MAP", and this type of file can be output by the Linker. Although this file is not required, it allows symbol names to be displayed during disassembly.

The symbols that are loaded are stored in list format in the hexadecimal input pad.

Caution

The extension for files output by the Linker is ".EVA". This type of file is converted to a HEX file by E2H86K.EXE. Because the Visual Memory Simulator cannot load EVA files, these files must be converted to HEX files.

3.5 Drag & Drop

The drag & drop technique can be used with the text boxes where addresses are input. In order to begin dragging from a given text box, hold down **Shift** key and press the left mouse button. The mouse cursor changes to a drag cursor, confirming that dragging is enabled.

The address labels and the hexadecimal input pad text boxes that are displayed on the Special Function Register Panel can be dragged without using **Shift** key. When the mouse is moved to one of these areas, it changes to a drag cursor.

Descriptions of Windows and Panels

When the Visual Memory Simulator is started up, the Main Window is displayed first. If all that is necessary is to load and execute an application program that has been created, the functions in the Main Window are all that are needed. Debugging requires the use of functions on a number of other windows.

Main Window

This is the main window for the Visual Memory Simulator. Applications can be loaded and execution can be controlled through this window.

Memory Control Window

This window displays the contents of memory implemented in Visual Memory. The contents of memory can also be edited on this screen.

Break Control Window

This window is used to set execution stop triggers for breakpoints.

Special Function Register Control Window

This window displays the status of the Special Function Registers.

LCD Snapshot Window

This window gets and enlarges images that are displayed on the LCD.

Network Monitor Window

This control window is used to connect two Visual Memory Simulators.

Trace Panel

This panel is used to perform program traces.

Hexadecimal Input Pad

This window is used to easily input hexadecimal numbers. A symbol table is also stored here.

Environment Settings Window

This window is used to make the basic settings for Visual Memory Simulator operation.

4.1 Main Window

The following functions are implemented in the main window:

- Loading applications and system files
- Calling up control windows and panels
- Displaying CPU registers
- Executing, stopping, and step-executing applications
- Outputting disassembled listings
- Simulating the Visual Memory LCD and buttons
- Switching the Main Window between reduced and normal size display

The following Main Window functions are described in this section:

- Menus
- Speed button
- CPU register display function
- Execution control
- Disassembly function
- Visual Memory image
- Status lamps
- Changing the Main Window size
- System console

4.1.1 Menus

File Menu

[Open Application] Command

This command loads an application in HEX file format. The application is loaded into flash memory bank 0. Flash memory bank 0 is used as memory for Visual Memory applications.

[Re-open Application] Command

This command reloads the application that is currently open. This command cannot be selected initially; it can only be used after an application has been loaded. The name of the file that was loaded is displayed in the title bar on the Main Window.

[Open System File] Command

This command loads the system BIOS into the internal ROM area. A setting can be made in the Environment Settings Window that will load the system BIOS automatically when the Visual Memory Simulator is started up.

[Open RAM File] Command

This command loads a RAM file that was saved. A RAM file contains the contents of RAM that were saved. RAM banks 0 and 1, work RAM, and XRAM are included in a RAM file.

The file format is binary. The memory map is as shown below.

0000H - 00FFH	RAM bank #0
0100H - 017FH	SFR (reserved for system)
0180H - 01FFH	XRAM bank #0
0200H - 027FH	Reserved for system
0280H - 02FFH	XRAM bank #1
0300H - 037FH	Reserved for system
0380H - 03FFH	XRAM bank #2
0400H - 04FFH	RAM bank #1
0500H - 06FFH	VTRBF
0700H - FFFFH	Reserved for system

[Save RAM File] Command

This command saves the current contents of RAM provided for the virtual CPU. The file format is binary. This type of file can be loaded by using the [Open RAM File] command.

[Open FLASH#1] Command

This command loads a file into flash memory bank 1. The file format is binary. The size of flash memory bank 1 is 64K. Writing to flash memory is accomplished by writing directly to the memory area, ignoring the flash write simulation facility.

Flash memory bank 1 is a system area that is used to manage Visual Memory files, and an area for saving Dreamcast game data. Visual Memory applications are loaded into flash memory bank 0.

Caution

If "GAME.BIN" is not loaded through this menu before starting up an application, an error message stating that Visual Memory has not been initialized will be displayed. Before starting up an application, be certain to first load "GAME.BIN" through this menu.

[Save FLASH#1] Command

This command saves the current contents of flash memory bank #1 in a file. The file format is binary. This type of file can be loaded by using the [Open FLASH#1] command.

[Print] Command

This command prints the character string that is displayed in the text box (system console) at the bottom of the Main Window.

[Save Console to File] Command

This command saves the character strings displayed in the system console in a text file.

[Exit] Command

This command quits the Visual Memory Simulator.

[Execute] Menu

[Break] Command

This command halts application execution. The effect of this command is identical to that of the Break button.

[Reset] Command

This command resets the virtual Visual Memory, and starts Visual Memory operation.

[Run/Continue] Command

This command executes the program, starting from the instruction following the instruction at which program execution was stopped.

[Step Execution] Command

This command executes one instruction according to the current program counter.

[Disassemble] Command

This command displays a disassembled listing.

[Panel] Menu

[Break Control] Command

This command displays the Break Control Window.

[Memory Control] Command

This command displays the Memory Control Window.

[SFR Display] Command

This command displays the Special Function Register Window.

[LCD Snapshot] Command

This command displays the LCD Snapshot Window.

[Network Monitor] Command

This command displays the Network Monitor Window.

[Trace Panel] Command

This command displays the Trace Panel.

[Hexadecimal Input Pad] Command

This command displays the Hexadecimal Input Pad.

[Reduce Main Window] Command

This command changes the size of the Main Window to the size of the Visual Memory image.

[Normal Main Window] Command

This command restores the Main Window to its normal size.

[Options] Menu

[Environment Settings] Command

This command displays the Environment Settings Window.

[Clear Console] Command

This command clears the system console.

[Help] Menu

[Reference Guide] Command

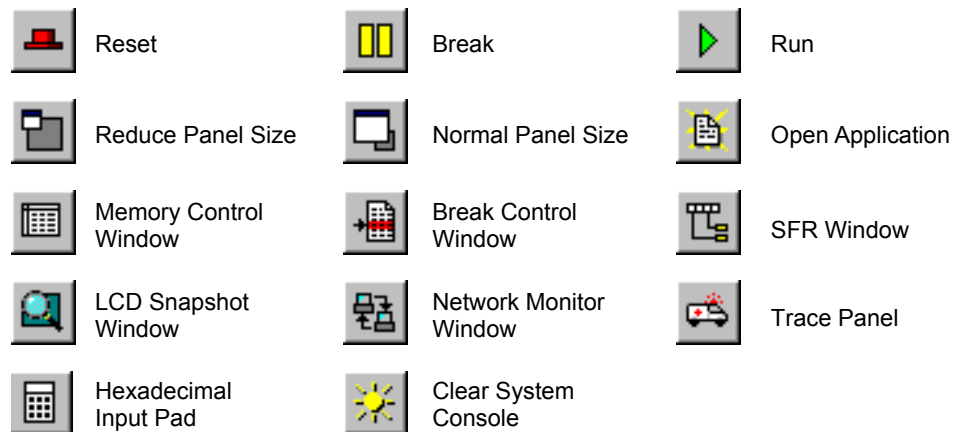
This command displays help.

[Version Information] Command

This command displays the version information for the Visual Memory Simulator.

4.1.2 Toolbar

The toolbar is located at the top of the panel. The toolbar buttons all correspond to menu items or buttons on panels.



4.1.3 CPU Register Display Function

This function displays the values of the virtual CPU's registers in the Main Window.

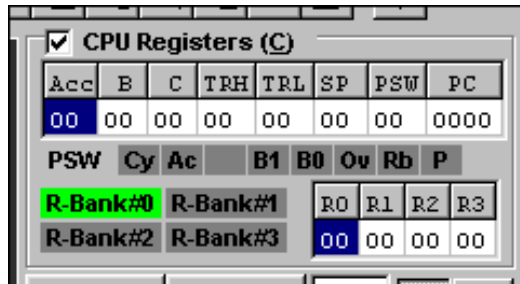


Fig. 4-1

The registers are Acc, B, C, TRH, TRL, SP, PSW, and PC. Each value is expressed in hexadecimal notation.

Each register may be edited. After selecting the register to be edited by clicking on the register, click on the register again to begin editing it. In the case of the PSW (Program Status Word), the status of each bit is displayed.

Meanings of the Bits in the PSW

Cy	Carry flag
Ac	Auxiliary carry flag
B1	Indirect register bank specification bit
B0	Indirect register bank specification bit
Ov	Overflow flag
Rb	RAM bank switching bit
P	Parity bit

Each bit of the PSW, except for the parity bit, can be inverted by clicking on the bit with the mouse. The results of the change are reflected in the value of the PSW.

The selected bank and the current indirect register values are displayed in the indirect registers.

The contents of the register scan be edited. It is also possible to change the current bank for the indirect registers by clicking on the label that indicates the bank. If B1 and B0 in the PSW are changed, the bank label is also updated.

Because the CPU registers can be displayed while an application is in progress, so the changes in register values can be observed. However, because it takes time to update the value of each register, the operating speed of an application will slow down if the register values are displayed. To suppress the register display, uncheck the [CPU Registers] checkbox. When this checkbox is in the checked state, the contents of the registers are displayed.

4.1.4 Execution Control

There are four buttons that are used for execution control.



Fig. 4-2

Execution Control Buttons

Reset Button

If the [Reset] button is clicked, all Visual Memory devices are reset.

All of the CPU registers are initialized with "00H," RAM bank 0 is the current RAM bank, and the internal ROM is selected as the program ROM. "0000H" is loaded into the program counter, and then the CPU begins running.

Run Button

If the [Run] button is clicked, execution begins, starting from the address that is shown in the execution start address text box (program counter). In this case, the devices are not reset. The [Sys]/[Usr] button indicates whether the program that is currently executing is located in ROM or in flash memory. The [Sys] button indicates ROM, and the [Usr] button indicates flash memory.

Break Button

If the [Break] button is clicked while an application is executing, Visual Memory displays the current register values on the console and halts execution. At this point, the value of the program counter, which is the address of the instruction that is to be executed next, is substituted into the execution start address text box. Execution can be resumed if the [Run] button is pressed right after the [Break] button.

Step Button

If the [Step] button is clicked while program execution is halted, the next instruction in the program is executed and then execution halts again. This button can be used to execute a program one instruction at a time in a deliberate fashion.

Register Dump Display Format

The format of the register dump that is displayed on the console is shown below.

```
A=10 B=01 C=03 TRH=05 TRL=ED SP=45 PSW=01 PC=1:027D
RBANK=0:0 R0=00 R1=00 R2=A6 R3=A7
1:027D 02 77                LD                0077H
```

The values from "A" to "PSW" show the current values in each of the registers.

"PC" indicates the program counter value, consisting of the bank and the address, separated by a colon.

Bank 0 indicates ROM, and bank 1 indicates flash memory. In the above example, "PC" indicates address 027DH in flash memory. The digits that are used to represent the bank are "0" and "1".

The next line shows the indirect registers. "RBANK" indicates the indirect register bank that is currently selected. The left side of the colon indicates the RAM bank. The value is either "0" or "1". The right side of the colon indicates the indirect register bank. The value can range from "0" to "3". The values from "R0" to "R3" indicate the indirect register values in the selected bank. A disassembled list is displayed only if the dump was executed while a flash memory program was running.

4.1.5 Disassembly Function

Clicking on the [DisAsm] button displays a disassembled list on the console.



Fig. 4-3

Text boxes are provided for the starting and ending addresses. The [Length] box is used to switch between either performing disassembly according to the number of lines, or performing disassembly according to the ending address specification. When the [Length] box is checked, disassembly is performed according to the specified number of lines. Under the default setting, 32 lines are displayed. The number of lines can be specified by specifying any number of lines in the environment settings window. For details on how to make this setting, refer to [Work Settings] - [Disassemble Lines] in section 4.9, "Environment Settings Window."

The results of execution are shown below.

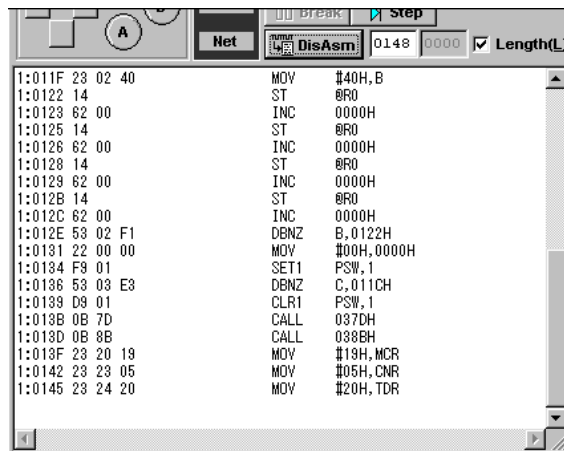


Fig. 4-4

4.1.6 Visual Memory Image

The Visual Memory image is a virtual target machine that is patterned on Visual Memory.



Fig. 4-5

The Visual Memory image includes an area that is equivalent to the LCD, icons, and eight buttons.

The buttons can be clicked through either the mouse or the keyboard. When the Visual Memory image is active, the image is framed in blue. To make the Visual Memory image active, click on any portion of the Visual Memory image other than a button.

The keys that correspond to the Visual Memory buttons can be changed through the environment setting window. For details on how to make this setting, refer to [Work Settings] - [VMS Button Configuration] in section 4.9, "Environment Settings Window."

4.1.7 Status Lamp

There are lamps that indicate the status of Visual Memory located on the right side of the Visual Memory image.



Fig. 4-6

Run	Lights when the CPU is running. Turns off when the CPU is stopped.
Halt	Lights when the CPU is in the HALT state.
Hold	Lights when the CPU is in the HOLD state.
RAM#0	Lights when RAM bank 0 is selected.
RAM#1	Lights when RAM bank 1 is selected.
S-ROM	Lights when ROM is selected.
U-ROM	Lights when flash memory is selected.
PWM	Lights when PWM is output by an application.
NET	Lights when the Visual Memory unit is connected to another Visual Memory unit.

4.1.8 Changing the Size of the Main Window

The Main Window can be reduced to a size that displays only the Visual Memory image and the [Reset], [Break], and [Run] buttons on the toolbar, and the panel size change button. In addition, it is possible to make a setting in the Environment Settings Window that sets this reduced size for the Main Window when the Visual Memory Simulator is started up. For details on how to make this setting, refer to [Settings] - [Minimum Size] in section 4.9, "Environment Settings Window."



Fig. 4-7

4.1.9 System Console

The system console outputs a variety of information from the Visual Memory Simulator. Text information that is output on the console can be printed or saved in a file.



Fig. 4-8

Under the default setting, the console has buffer space for 300 lines. The number of lines can be adjusted in the Environment Settings window. For details on how to make this setting, refer to [Work Settings] - [Console] in section 4.9, "Environment Settings Window."

If the text output is longer than the number of lines set for the system console, the text is deleted, starting from the beginning.

Caution

When the console buffer is set to a default of 300 lines or more, the operation of the Visual Memory Simulator will slow down.

4.2 Memory Control Window

This window displays the contents of memory. Memory is divided into tab pages by category. The memory that is displayed on each page can be the target of editing.

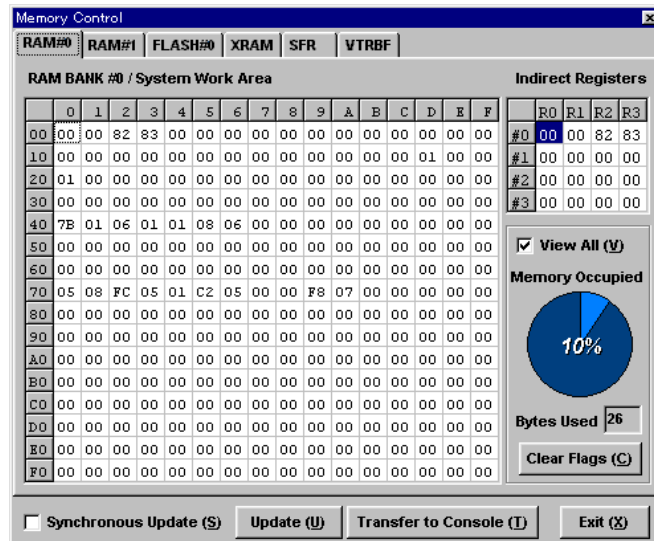


Fig. 4-9

Synchronous Display Function

There is a "Synchronous Update" checkbox in the Memory Control Window. If a check is placed in this checkbox, the displayed contents are updated whenever the virtual CPU writes to memory. However, because it takes time to update the screen, the Visual Memory Simulator will run more slowly if this function is used.

Dump Function

The current contents of the Memory Control Window can be transferred to the system console by clicking the [Transfer to Console] button. The 256 bytes of Flash Memory #0 and work RAM that are currently displayed are transferred to the console.

Update Button

Clicking the [Update] button causes the Memory Control Window to be updated with the current, most recent data. Normally, this button is used to update the data if the synchronous display function checkbox is not checked.

RAM#0	System work area for system BIOS, etc. Size: 256 bytes
RAM#1	Application area. Size: 256 bytes
FLASH#0	Flash memory area that is used to store user applications.
XRAM	LCD display memory.
SFR	Special Function Registers.
VTRBF	Work RAM. Size: 512 bytes

4.2.1 RAM#0, RAM#1

RAM is divided into bank 0 and bank 1. Bank 0 is a system work area that is used by the system BIOS. Bank 1 is a work area that is open to user applications.

The Memory Control Window display format is the same for both of these banks. The size of each bank is 256 bytes.

In the LC86 Series CPU, the first 16 bytes of RAM are allocated as the indirect register area. The indirect register area is displayed separately in an easy-to-read format on the panel.

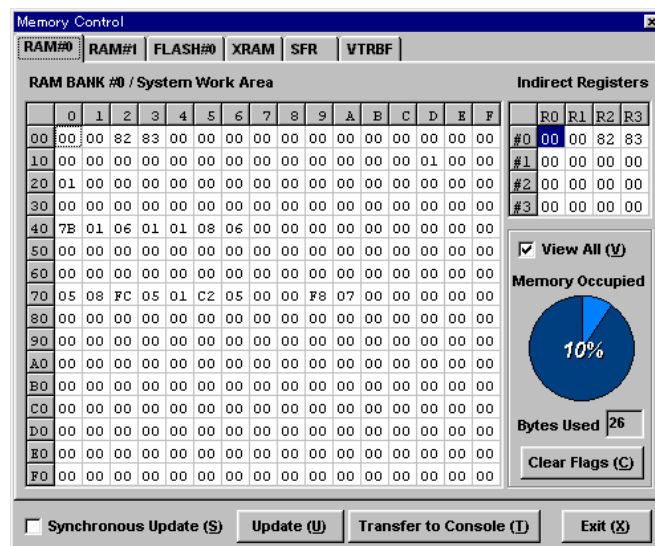


Fig. 4-10

Because RAM bank 0 is allocated for the system BIOS work area and the stack area, it should not be accessed by applications.

RAM bank 1 has 256 bytes that are open to applications.

Calculation of Memory Usage Rate

RAM contains internal flags that indicate whether a location was accessed by the CPU. When the CPU writes to a given location in memory, the corresponding flag is set. These flags are counted and then used to calculate the memory usage rate. These flags are cleared when the CPU is reset. Specific flags can also be deleted by clicking the [Clear Flags] button.

If the [View All] checkbox is checked, the flags are ignored and all 256 bytes are displayed. If this checkbox is not checked, only memory for which flags have been set is displayed. In other words, the memory that the virtual CPU has written to is displayed.

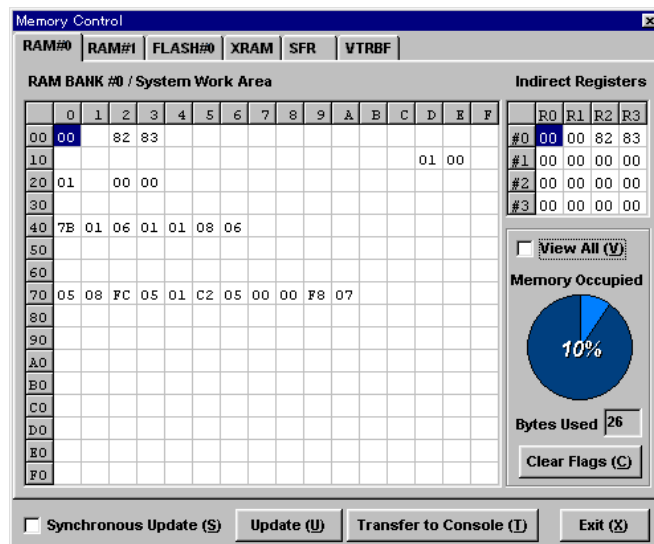


Fig. 4-11

4.2.2 FLASH#0

Flash memory is divided into bank 0 and bank 1. The size of each bank is 64 kilobytes each.

Flash memory bank 0 is used for application programs. User-created programs are loaded into this area. Flash memory bank 1 is a data area, so programs cannot be loaded into flash memory bank 1.

Caution

The contents of flash memory bank 1 cannot be changed.

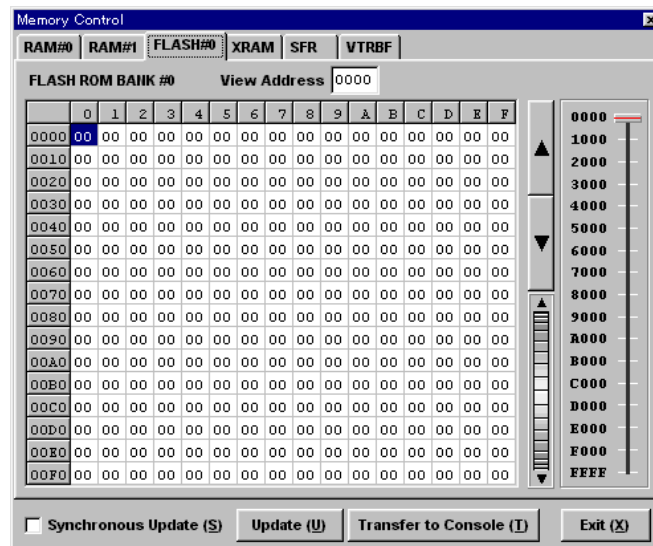


Fig. 4-12

The memory panel displays 256 bytes at one time.

Scroll Buttons

The up and down scroll buttons can be used to scroll through the display 256 bytes at a time.

Slider

The slider box can be dragged with the mouse in order to move the display to any desired position in memory. The display can also be moved to a certain position by clicking on the label in which that address is displayed.

Dial

The dial can be moved up and down to change the displayed address accordingly. The unit of movement is 16 bytes.

Display Start Address

If an address is input in the text box, the 256 bytes that start from that address are displayed. The lower four bits are discarded so that the display starts from "0".

4.2.3 XRAM

XRAM is video memory for the LCD.

XRAM is divided into three banks. Banks 0 and 1 are matrix memory, and bank 2 is icon memory. Banks 0 and 1 are displayed in the Memory Control Window.

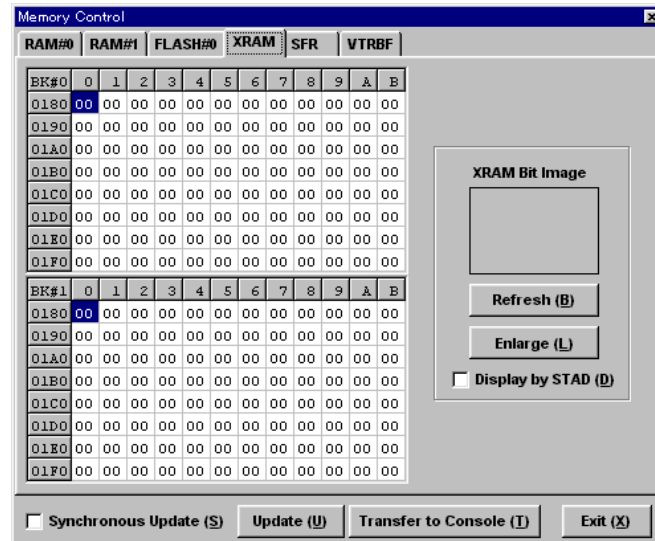


Fig. 4-13

LCD Bit Image Display

The current status of XRAM can be displayed as a bit image. This display area can be displayed even when the LCD is off. Although the bit image is synchronized with user writes, it is not synchronized with writes by the virtual CPU. Clicking the [Refresh] button causes the latest contents of XRAM to be displayed.

The XRAM display start address can be changed. This specification is made through the Special Function Register STAD. When the [Display By STAD] checkbox is checked, the value in STAD is used as the display start address. If the [Display By STAD] checkbox is not checked, the display starts at the start of XRAM. This is the same as if STAD = 0.

Note

The LCD resolution is 48 dots (H) x 32 dots (V), and one line of the LCD corresponds to 6 bytes. The MSB of data that is written corresponds to the left-side dot.

XRAM bank 0 is displayed in the top half of the LCD, and bank 1 is displayed in the bottom half. Bank 2 is icon memory.

Caution

Because bank 2 is used for the icon that displays the Visual Memory mode, do not change the contents of bank 2 from within an application.

4.2.4 SFR

Although the Special Function Registers are displayed, areas that are not actually implemented are also displayed. Normally, locations for which no device is connected are indicated as "0FFH". The data that is displayed can be edited.

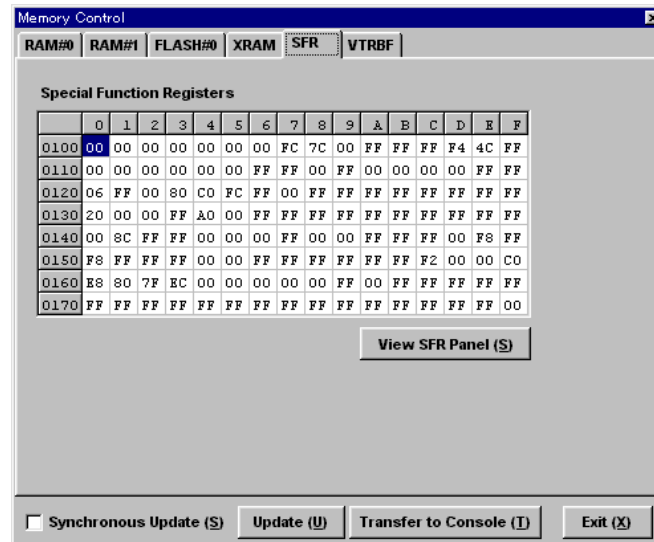


Fig. 4-14

Caution

If data is edited in an address that does not exist in any storage that is connected to the device, the data is not actually written.

4.2.5 VTRBF

VTRBF is allocated as buffer memory for communications between Visual Memory and Dreamcast. If communications with Dreamcast are not being performed, however, this area is open to the user as work RAM. This memory is accessed through the SFRs, and is not decoded in the CPU memory space. The size of this area is 512 bytes, and the addresses range from 0000H to 01FFH.

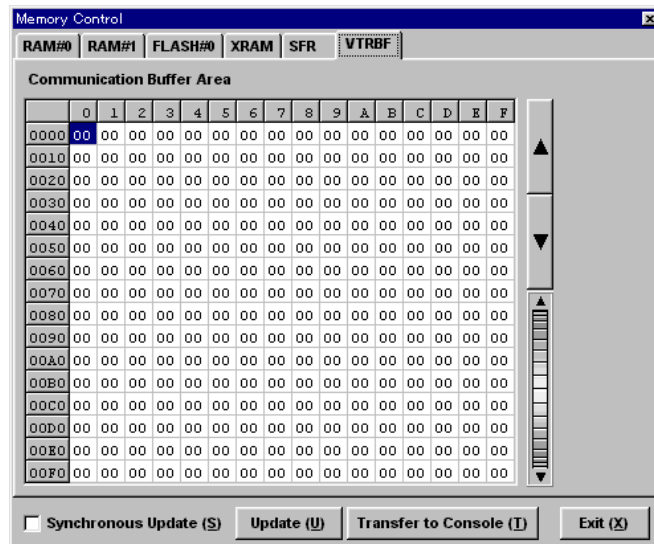


Fig. 4-15

4.3 Break Control Window

There are three execution control functions that are implemented in the break control window.

Break by Breakpoint Address Comparison

Aside from breakpoints, program execution can be stopped by memory fetches.

Display When an Interrupt Is Received

This indicates that the virtual CPU has received an interrupt and that an interrupt routine has been called.

Access Reference Monitor

This function displays the position in a program where the specified memory is being accessed.

To enable the address set in the Break Control Window, click the [Apply] button.

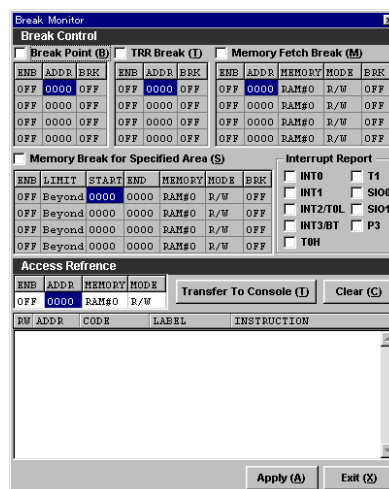


Fig. 4-16

4.3.1 Break by Breakpoint Address Comparison

Break Control

Break control specifies monitoring of application execution. There are four monitoring groups: breakpoint specification, TRR fetch breaks, memory fetch breaks, and memory fetch breaks with a range specification. Four compare addresses can be set for each group. The common items for all of the groups are explained below.

Group ON/OFF

This item turns individual groups on and off. The switches for the individual groups are provided in order to minimize address comparison overhead in the Visual Memory Simulator. If a group is turned on, it is displayed on a white background and its settings are enabled.

Address ON/OFF

Each compare address in a group has its own ON/OFF switch. If set to ON, that compare address is enabled. Addresses can be turned on and off by clicking in the first column.

Break Mode

The break mode specifies whether to stop or continue execution when a compare address matches. When a compare address matches, the current register values are dumped. If "Break" is ON, the virtual CPU stops executing the program after the register dump. If "Break" is OFF, the register dump is still performed, but the virtual CPU continues executing.

Breakpoints

Execution is halted when the program counter matches the specified address. Four addresses can be specified as compare addresses.

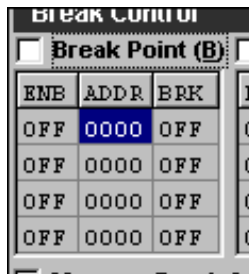


Fig. 4-17

TRR Fetch Break

Execution halts when the program counter matches the memory address that is referenced by the TRH and TRL (indirect address) registers.

Essentially, the program counter is compared with the address that is referenced when the LDC instruction was executed.

Therefore, when the LDC instruction is executed, the address indicated by TRH and TRL is the object of comparison, with no distinction made for flash memory.

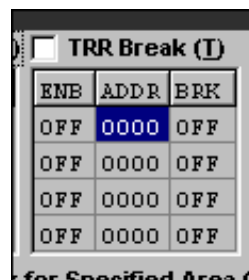


Fig. 4-18

Memory Fetch Break

A memory fetch break halts execution when the CPU accesses the specified address in memory. This group permits specification of the target memory and the access mode.

The target memory can be RAM#0, RAM#1, SFR, XRAM#0, XRAM#1, or VTRBF. To select the target memory, click on the column that is to be set, and then make the selection in the popup menu that appears.

Select the access mode from among READ, WRITE, and R/W.

If the access mode is READ, execution is halted when a read is executed in the target memory; if the access mode is WRITE, execution is halted when a write is executed in the target memory. If the access mode is R/W, execution is halted when a read or a write is executed in the target memory.

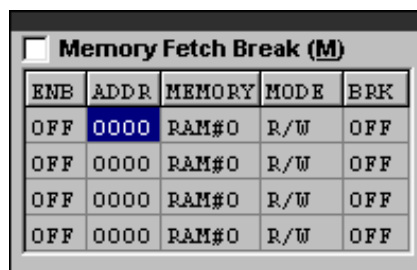


Fig. 4-19

Memory Fetch Break With Range Specification

A memory fetch break with range specification halts execution when an access is made inside or outside of the specified memory address range.

The compare range is specified with a [Start] address and an [End] address. The compare condition can be selected as either "inside the range" or "outside the range." If "outside the range" is specified, then execution stops when memory is accessed outside of the specified address range. This condition does not include the specified addresses. If "inside the range" is specified, then execution stops when memory is accessed inside of the specified address range. This condition does include the specified addresses.

Just as in the case of a memory fetch break, this group permits specification of the target memory and the access mode.

The target memory can be RAM#0, RAM#1, SFR, XRAM#0, XRAM#1, or VTRBF.

To select the target memory, click on the column that is to be set, and then make the selection in the popup menu that appears.

Select the access mode from among READ, WRITE, and R/W.

If the access mode is READ, execution is halted when a read is executed in the target memory.

If the access mode is WRITE, execution is halted when a write is executed in the target memory.

If the access mode is R/W, execution is halted when a read or a write is executed in the target memory.

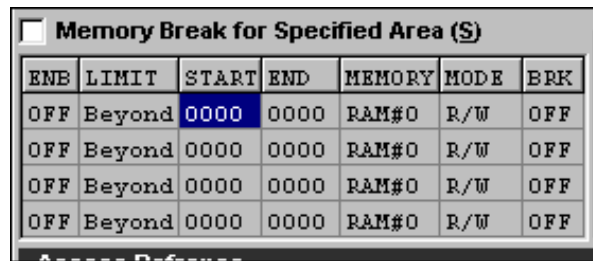


Fig. 4-20

4.3.2 Display When an Interrupt Is Received

Interrupt Report

When the virtual CPU accepts an interrupt, it outputs an acceptance message on the system console.

This message is output after the virtual CPU has gotten the interrupt vector.

This is valid when the interrupt source checkbox has been checked.

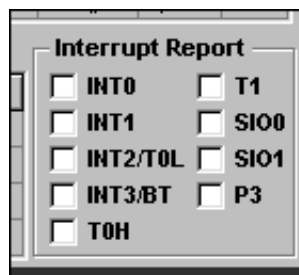


Fig. 4-21

These checkboxes will function correctly even if their settings are changed while the CPU is running. The interrupt sources are described below:

- | | |
|-----------------|---|
| INT0 | External interrupt. This interrupt is generated when +5V is supplied to the Visual Memory unit. |
| INT1 | External interrupt. This interrupt is generated when the Visual Memory unit's internal battery voltage drops. |
| INT2/T0L | The external interrupt is generated by ID0, and the internal interrupt is generated by the lower timer 0 register. |
| INT3/BT | The external interrupt is generated by ID1, and the internal interrupt is generated by the base timer. |
| T0H | This interrupt is generated by the upper timer 0 register. |
| T1 | This interrupt is generated by timer 1. |
| SIO0 | This interrupt is generated by SIO0. |
| SIO1 | This interrupt is generated by SIO1. |
| P3 | This interrupt is generated by port 3. |

4.3.3 Access Reference Monitor

The access reference displays the position in a program that is accessing the specified memory. Usually, this function is used to pinpoint a position in a program that is destroying memory.

The access reference monitor function permits selection of the access mode.

The access mode may be specified as either READ, WRITE, or R/W.

The displayed contents are the mode in which the access was made (R or W), and a disassembly of the program position. This information is output on a special console.

Because program positions are checked twice when output, they are not listed for each access. If you wish to know the access sequence over time, use the memory fetch break function. If you use the access reference monitor, the information will be output on the system console each time an access occurs.

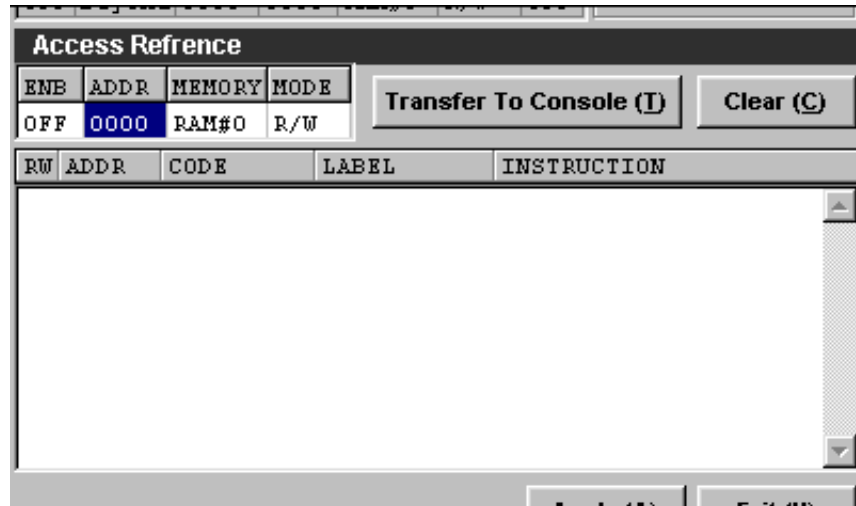


Fig. 4-22

4.4 Special Function Register Control Window

This window displays the Special Function Registers that Visual Memory leaves open to users.

The display in this window is divided into several groups.

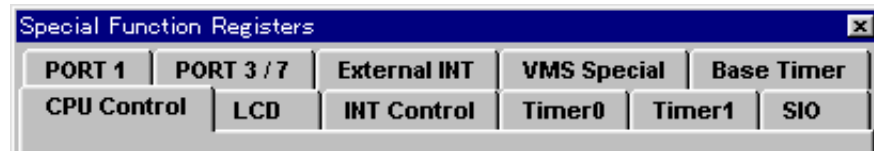


Fig. 4-23

Each group is a tabbed page; click on the tab for the group that you want to display.

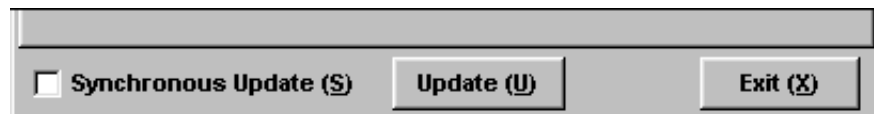


Fig. 4-24

Click the [UPDATE] button in order to display the most recent information. Checking the [Synchronous Update] box causes the contents of registers to be updated as soon as the virtual CPU performs a write.

Each register can be edited at the bit level. Clicking on one of the displayed bits causes the value of that bit to be inverted. Bits can also be inverted by clicking on the label connected to that bit.

The Special Function Register groups displayed in this window are listed below.

- CPU Control
- LCD
- INT Control
- Timer0
- Timer1
- SIO
- PORT1
- PORT3/7
- External INT
- VMS Special
- Base Timer

4.4.1 CPU Control

This group includes the CPU power control, system clock oscillation source control, and external memory control registers. The target registers are PCON, OCR, and EXT.

PCON is the power control register, OCR is the oscillation control register, and EXT is the external memory register.

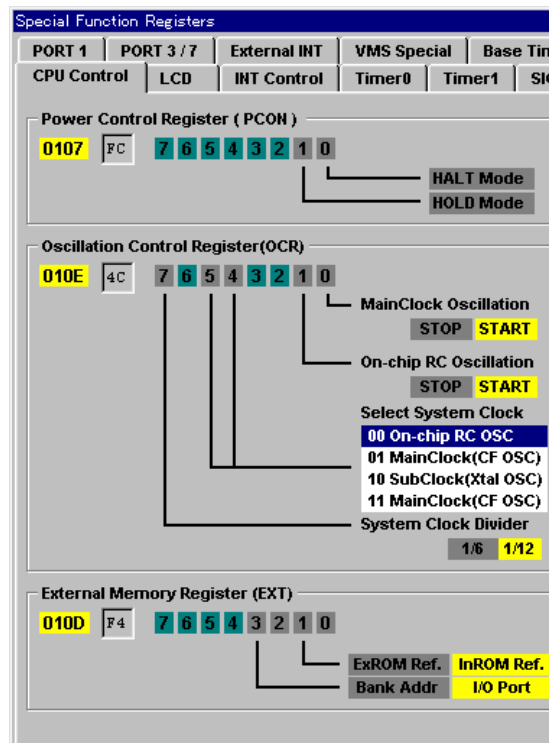


Fig. 4-25

4.4.2 LCD

This group displays the LCD control registers. The target registers are MCR, STAD, CNR, RDR, XBNK, and VCCR.

STAD, CNR, TDR, and VCCR cannot be written while the liquid crystal display controller is stopped. This also applies to accesses from an application.

XBNK is unrelated to the operation of the LCD controller, and can be accessed at any time. Although it may appear that it is possible to set this to an unused bank, such a setting is corrected to bank 0.

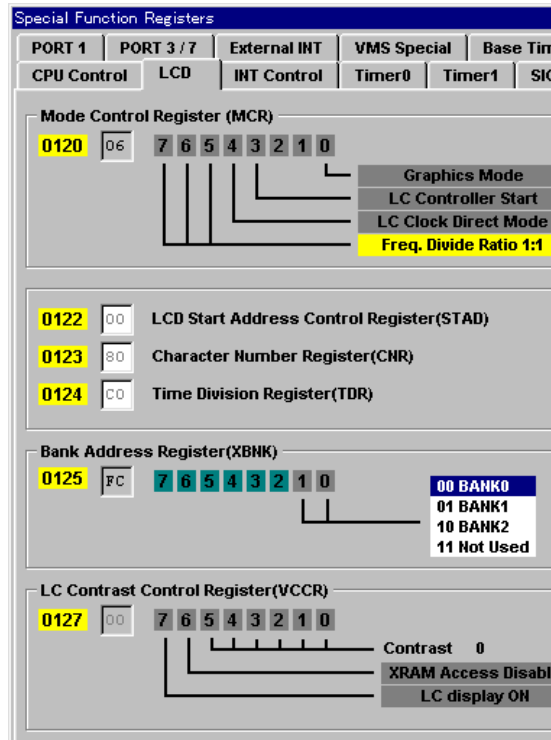


Fig. 4-26

4.4.3 INT Control

This group displays the interrupt-related registers. The target registers are IE and IP.

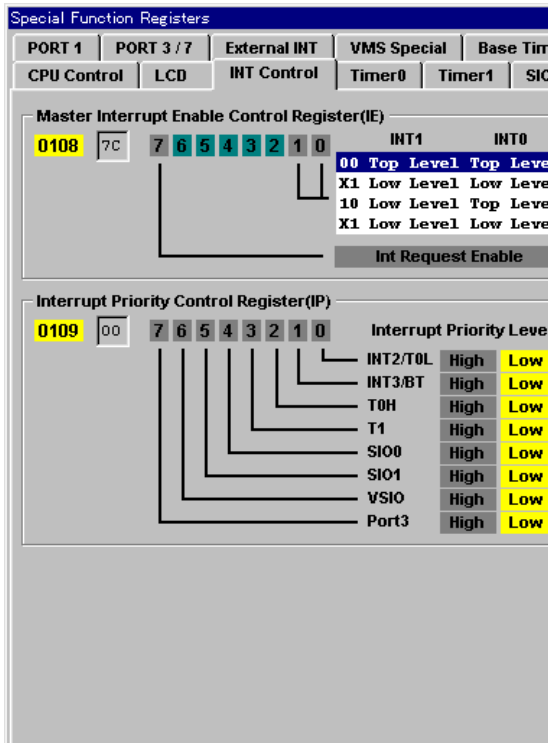


Fig. 4-27

4.4.4 Timer 0

This group displays the registers that are related to timer 0. The target registers are T0CNT, T0PRR, T0L, T0LR, T0H, and T0HR.

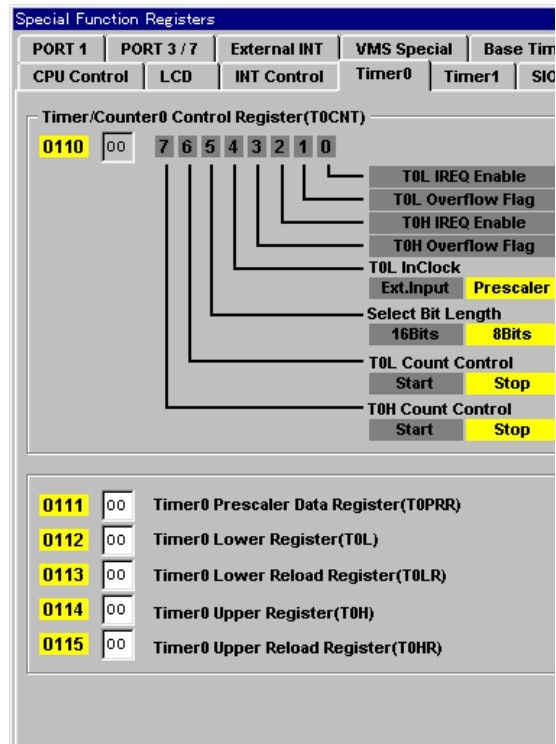


Fig. 4-28

4.4.5 Timer 1

This group displays the registers that are related to timer 1. The target registers are T1CNT, T1LC, T1L, T1HC, and T1H.

The roles of registers T1L and T1H differ, depending on whether they are being read or written. When read, they return the counter value; when written, the value becomes the reload value. Each status can be checked on the SFR panel.

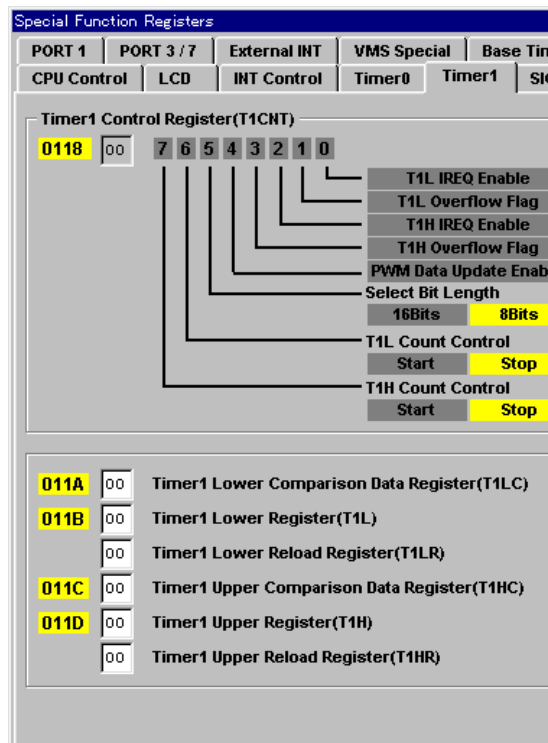


Fig. 4-29

4.4.6 SIO

This group displays the registers for the serial communications-related circuits for two channels. The target registers are SCON0, SCON1, SBUF0, SBR, and SBUF1.

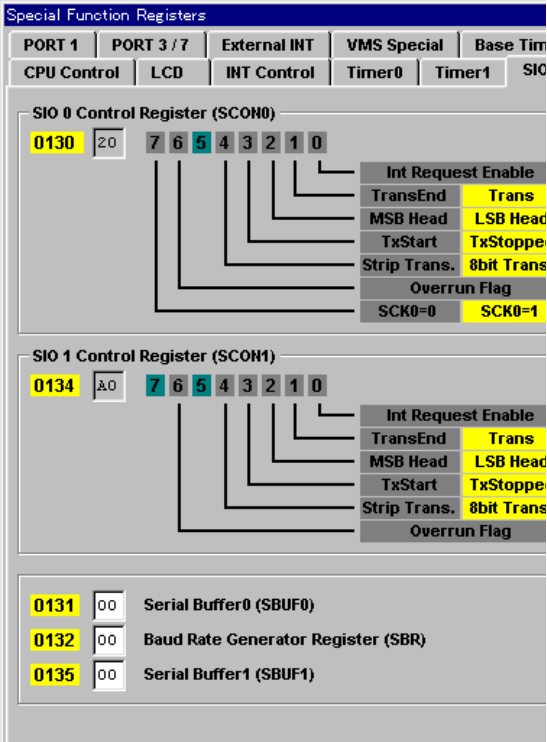


Fig. 4-30

4.4.7 PORT1

This group displays the registers related to port 1. The target registers are P1, P1DDR, and P1FCR.

Because port 1 is used for serial communications and PWM (buzzer) output control, it cannot be used as a general I/O port.

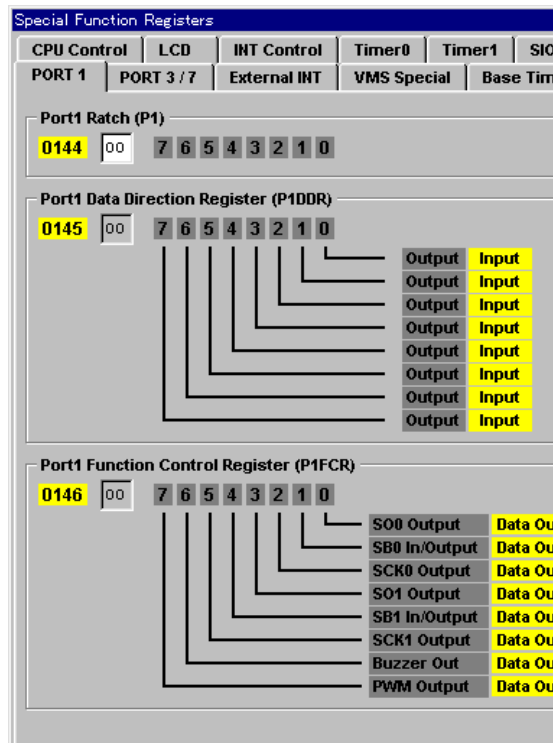


Fig. 31

4.4.8 PORT3/7

This group displays the registers related to port 3 and to port 7. The target registers are P3, P3DDR, P3INT, and P7.

Eight buttons of the Visual Memory unit are connected to port 3. The signals corresponding to each button are normally high, but if a button is pressed the signal goes low. Port 3 interrupts can be generated through P3INT. Note that P3 interrupts are level interrupts.

Port 7 is a four bit input port, with special input signals connected. Each bit of port 7 is an external interrupt input port. Interrupt control is handled through the I01CR and I23CR registers.

- P70 is connected to the +5V supply test checkbox. Normally, this signal is low, but when +5V is supplied this signal goes high. This signal can generate interrupts as external interrupt INT0.
- P71 is connected to the low voltage detection test checkbox. Normally, this signal is high, but when low voltage is detected this signal goes low. This signal can generate interrupts as external interrupt INT1. The signal goes low when the checkbox is checked.
- P72 is connected to the special signal ID0 checkbox. Normally, this signal is low. This signal can generate interrupts as external interrupt INT2.
- P73 is connected to the special signal ID1 checkbox. Normally, this signal is low. This signal can generate interrupts as external interrupt INT3.
- The [VMS Connect] checkbox simulates another Visual Memory unit being connected. When this checkbox is checked, the port values for the connected state are simulated.

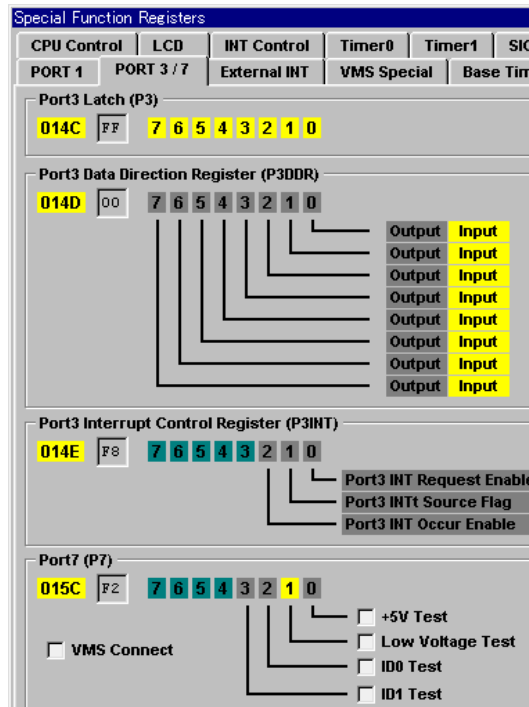


Fig. 4-32

4.4.9 External INT

This group displays the external interrupt control-related registers. The target registers are I01CR and I23CR.

INT0 is the +5V supply test, and INT1 is the low voltage detection test. In addition, INT2 is connected to ID0, and INT3 is connected to ID1.

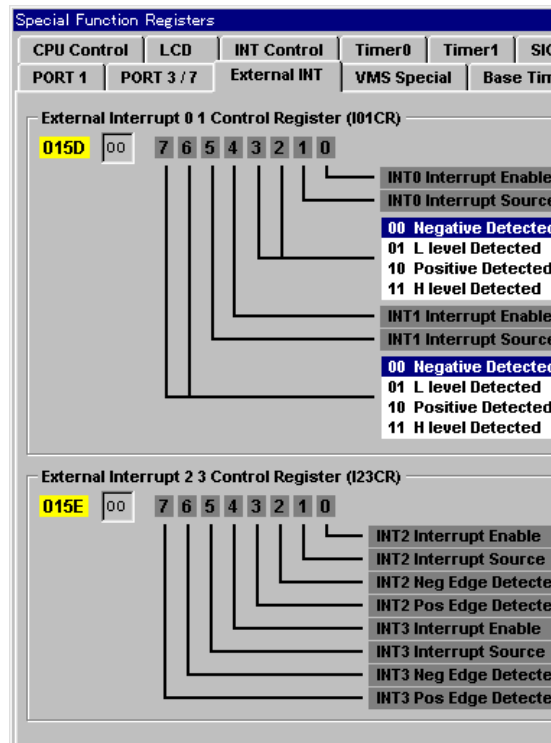


Fig. 4-33

4.4.10 VMS Special

This group displays related registers among the registers that are related to the Visual Memory special serial circuitry. The target registers are VCFLG2, VSEL, VRMAD1, VRMAD2, and VTRBF.

VTRBF has [Read] and [Write] buttons. VTRBF can also be listed in the Memory Control Window.

The Visual Memory Simulator only supports registers for access to VTRBF.

The screenshot shows a software interface titled "Special Function Registers". At the top, there is a navigation bar with tabs: CPU Control, LCD, INT Control, Timer0, Timer1, SIO, PORT 1, PORT 3 / 7, External INT, VMS Special (selected), and Base Tim. Below the tabs, the "VMS Special" section is active. It contains several register entries:

- Control Flag2 (VCFLG2)**: Register 0162, value 7F. Bit fields 7, 6, 5, 4, 3, 2, 1, 0 are shown. Bit 5 is connected to a line labeled "SRES".
- Control Register (VSEL)**: Register 0163, value EC. Bit fields 7, 6, 5, 4, 3, 2, 1, 0 are shown. Bit 5 is connected to a line labeled "ASEL". Bit 4 is connected to a line labeled "SIOVSEL". Bit 3 is connected to a line labeled "VRMAD Enable Auto increment". To the right, there are labels: OTAMA, CPU, Maple, SIO.
- Address Register 1 for System (VRMAD1)**: Register 0164, value 00.
- Address Register 2 for System (VRMAD2)**: Register 0165, value 00.
- Send/Receive Buffer (VTRBF)**: Register 0166, value 00. It has "Write" and "Read" buttons.

Fig. 4-34

4.4.11 Base Timer

This group displays the registers related to the base timer. The target registers are BPCR and ISL.

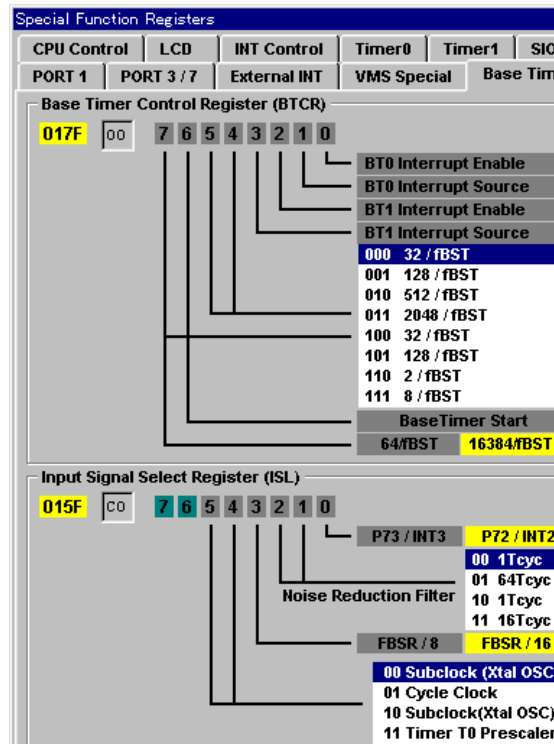


Fig. 4-35

4.5 LCD Snapshot Window

This window displays an enlarged version of the bit image that is currently displayed on the LCD. The bit image is fetched either when this window is called or when the [Get Screen] button is clicked. If this window is displayed, the bit image shown is not synchronized with writes by the virtual CPU.

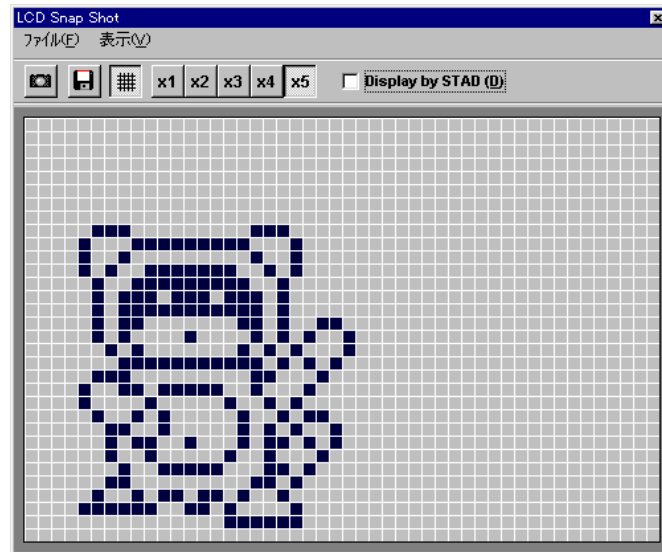


Fig. 4-36

4.5.1 Description of Tool Bar Buttons



Get Screen Button

This button gets the current LCD bit image. The contents that are gotten are the contents of XRAM. The dot size is determined by the current magnification.



Save Button

This button saves the current bit image in a file with the displayed magnification. The file is saved in ".BMP" format. The grid is not saved.



Grid Button

This button displays a grid in the display area. This button functions as a toggle switch; each time the button is clicked, it turns the grid on or off.



Zoom Button

This button can be used to select a magnification from 1x to 5x.

4.5.2 Display by STAD Checkbox

The [Display by STAD] checkbox is a switch that enables the display start address register STAD. When this checkbox is checked, drawing is based on addresses converted according to the STAD register. In other words, the same image as that which is displayed on the virtual LCD is displayed on the screen.

If this checkbox is not checked, the STAD register value is ignored when drawing the image. In other words, the contents of XRAM are drawn as is, starting from the beginning of XRAM.

4.5.3 Menus

[File] Menu

[Save Bit Image] command
[Exit] command

Same function as the [Save] button.
Closes the LCD Snapshot Window.

[Display] Menu

[Get Image] command
[Display Grid] command

Same function as the [Get Screen] button.
Same function as the [Grid] button.

4.6 Network Monitor Window

The Visual Memory Simulator simulates data transfers between Visual Memory units via TCP.

The Network Monitor Window supports TCP communications between two Visual Memory Simulators.

This panel consists of several buttons related to connection control, a console for outputting the status, a data monitor for displaying the transferred data, and a status bar for displaying the current status.

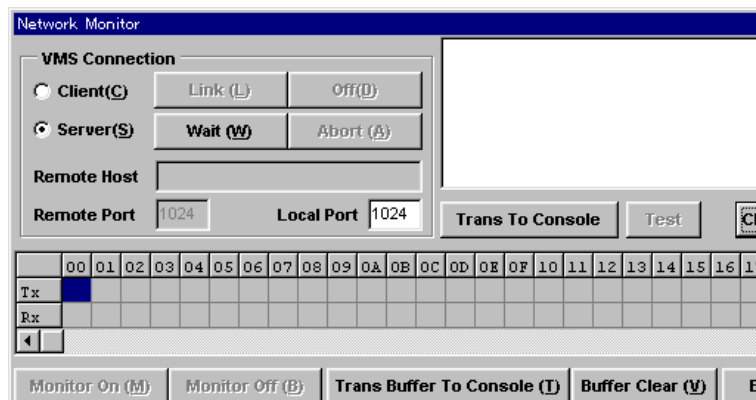


Fig. 4-37

Connection Control

To perform communications, each unit must select either client mode or server mode. If one is set to server mode, the other must be set to client mode.

Setting the Unit as the Server

1. The option buttons permit selection of either [Client] or [Server]; select [Server].
2. In order to set the unit as the server, the local port number must be set. The default setting is 1024.
3. If this number is OK, click the [Wait] button. The Visual Memory Simulator is now in server mode in the standby state.

The server performs connection processing when there is a connection request from the client. When the connection is completed, the "Net" lamp in the Main Window lights.

Stopping Server Operation

Click the [Abort] button to release the server standby state, or to disconnect. When in the standby state, clicking the [Abort] button puts the unit into the stopped state. If the unit is connected when the [Abort] button is clicked, it performs disconnect processing and then enters the stopped state.

Setting the Unit as the Client

1. Select [Client] with the option buttons.
2. Input the machine name or IP address that was set for the server in the [Remote Host] text box.
3. Input the port number that was set for the server in the [Remote Port] text box.
4. Click the [Link] button.
5. Once the connection is made properly with the server, a confirmation message is displayed.

When the connection is made with the server, the "Net" lamp in the Main Window lights.

Stopping Client operation

To release the connection with the server, click the [Off] button. When this button is clicked, the client issues a disconnection request to the server and then enters the unconnected state.

At this point, the server is in standby state. Reconnection is possible by clicking the [Link] button.

Console

The console displays statuses related to connection/disconnection.

The contents displayed in this console can be transferred to the system console by clicking the [Trans To Console] button.

The [Clear] button clears the contents displayed on the console.

Clicking the [Test] button while a connection is established sends a test message to the other side.

Data Monitor

The data monitor monitors the data that is sent between the two Visual Memory Simulators. This data is the data that is transferred from the virtual SIO.

Data that is sent is displayed on the Tx grid, and data that is received is displayed on the Rx grid.

The data monitor function begins operating when the [Monitor On] button is clicked, and stops when the [Monitor Off] button is clicked. When the buffer becomes full, the oldest displayed contents are overwritten first.

The [Trans Buffer to Console] button transfers the currently displayed contents of the data monitor buffer to the system console. The [Buffer Clear] button clears all of the current contents of the buffer.

Status Bar

Starting from the left, the status bar consists of:

- Client/server mode indication
- Data monitor operating mode indication
- Client, server connection status

4.7 Trace Panel

The Trace Panel traces the execution of an application.

The trace results are output on the trace console.

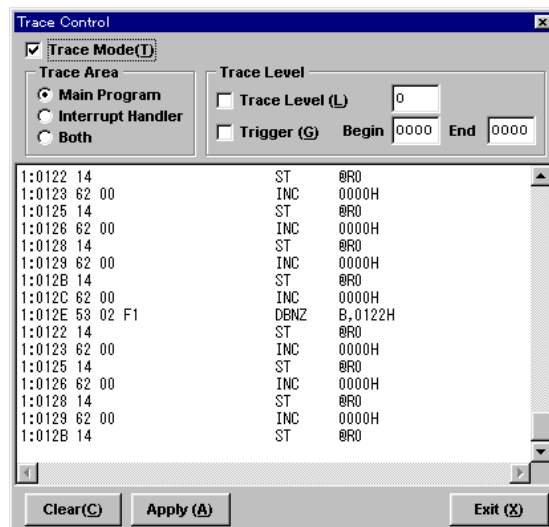


Fig. 4-38

Trace Mode Checkbox

Tracing starts when the [Trace Mode] checkbox is checked. The setting of this checkbox can be changed even while an application is running.

Trace Area

This specifies the area to be traced.

Main Program

This limits tracing to the main program. Here, "main program" indicates areas other than the interrupt processing routine.

Interrupt Handler

This traces only the interrupt processing routine (interrupt handler). Tracing starts when an interrupt is received, and continues until the RETI instruction is executed.

Both

This traces both the main program and the interrupt processing routine.

Trace Level

Trace Level

The trace level refers to the subroutine nesting level. At level 0, no subroutines are traced. If the level number increases, subroutines to the corresponding nesting level are traced. This setting can be used to avoid unnecessary tracing.

To enable the trace level, check the [Trace Level] checkbox.

Trigger

This is a switch that enables a trace start address and a trace end address.

When the program counter matches the start address, tracing starts and continues until the program counter matches the end address. These addresses are valid only for flash memory.

Trace Console

The trace results are displayed on the trace console. The trace results are the disassembled code that the virtual CPU executed.

To clear the contents of the trace console, click the [Clear] button.

Apply Button

Clicking the [Apply] button places the trace level value, trace start address, and trace end address into effect.

4.8 Hexadecimal Input Pad

Hexadecimal Input Pad

The Hexadecimal Input Pad is an auxiliary panel that is used to input hexadecimal numbers. Symbols are displayed on the right side of the panel.

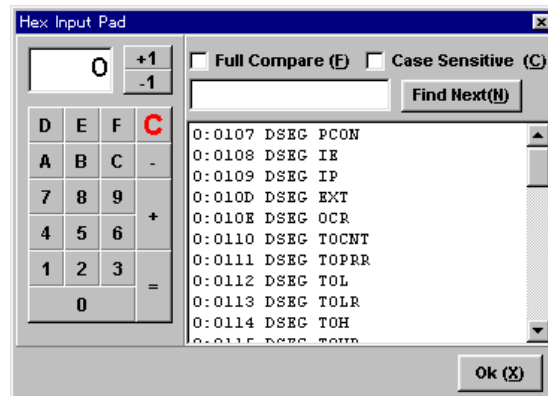


Fig. 4-39

Description of Input Buttons

Numeric Buttons

These buttons are used to input hexadecimal digits. The digits that are input are inserted from the right edge, and are then shifted left. Overflow digits are ignored; only the four digits that are displayed are valid.

C Button

This button clears the displayed digits and returns the display to "0".

+ Button

This button is used for addition in the same manner as a calculator.

- Button

This button is used for subtraction in the same manner as a calculator.

= Button

This button displays the total in the same manner as a calculator.

+1 button

This button adds "1" to the current displayed value. If the current displayed value is "FFFF" and this button is pressed, "0" is the result.

-1 button

This button subtracts "1" from the current displayed value. If the current displayed value is "0" and this button is pressed, "FFFF" is the result.

How To Use the Displayed Number

The displayed number can be dragged. When the mouse cursor is moved to the display area, it becomes a drag cursor. The number can be dragged and dropped in an address text box on any panel.

Keyboard Correspondence

The buttons for the digits also correspond to keys on the keyboard or numeric keypad.

The buttons for the digits "0" through "9", the letters "A" through "F", and the "+" symbol all correspond to the same keys on the keyboard, but the "C" button corresponds to the "*" key and the "=" button corresponds to the "Enter" key. The "+1" button corresponds to the **PageUp** key, and the "-1" button corresponds to the **PageDown** key.

In order to input from the keyboard, it is necessary to first make the numeric buttons on the Hexadecimal Input Pad active. The numeric buttons can be made active by clicking either on or near the buttons with the mouse.

Symbol List Box

The symbol information for an application is displayed in the list box. The Special Function Register symbols are registered as the default.

If an appropriate symbol in the list box is selected, that address is transferred to the display box. An address can also be dragged directly from this list box.

Symbol Search

A symbol search can be performed by inputting the search character in the text box. Each time a character is input, a search is conducted for the symbol that matches that character (incremental search).

The [FindNext] button starts its search from the currently selected position. If the [Full Compare] checkbox is checked, a search is conducted for a symbol that matches the entire character string that was input.

If the [CaseSensitive] checkbox is checked, the search distinguishes between upper and lower case letters.

Caution

The symbol file is a map file that is output by the Linker. If this file resides in the same folder as the application, it is loaded into the Simulator at the same time as the application. If there is no map file, only the default symbols are available.

4.9 Environment Settings Window

The Environment Settings Window is used to make general settings and to make settings concerning the operation of the Simulator.

The items that are set in the Environment Panel are saved in the "VMS.ENV" file in the "Files" folder where the Visual Memory Simulator was installed. This file is loaded when the Visual Memory Simulator is started up, and the environment settings contained in the file are restored.

4.9.1 Settings

The general settings include the settings upon startup, warning specifications, etc.

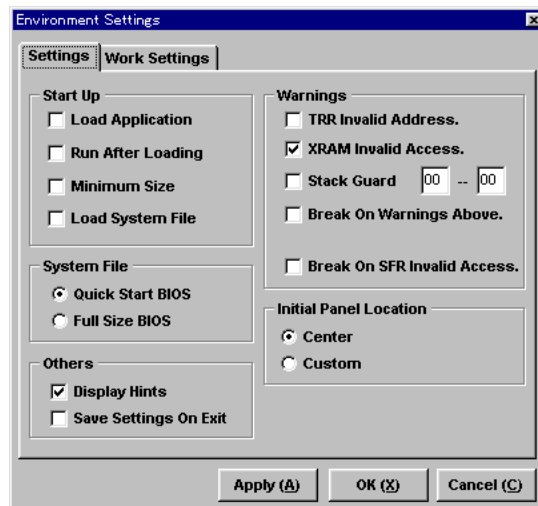


Fig. 4-40

Startup Settings

Load Application

This checkbox is used to automatically load, the next time that the Simulator is started up, the application that is currently loaded. The name of the current application is displayed in the title bar on the Main Window.

Run After Loading

This checkbox automatically performs the reset operation and initiates execution the next time that the Simulator is started up. If this checkbox is used at the same time as [Load Application], that application is loaded and then automatically executed.

Minimum Size

This displays the Main Window at its minimum size the next time that the Simulator is started up.

Load System File

This checkbox automatically loads the system BIOS the next time that the Simulator is started up. The system BIOS file that is loaded is the file that is selected by the system file setting.

Caution

This checkbox must be checked in order to execute an application automatically.

System File Setting

This item selects the system file that is loaded by the [Load System File] checkbox. There are two system files: [Quick Start BIOS] and [Full Size BIOS]. Select one or the other by clicking the option buttons.

Caution

Quick start BIOS supports exactly the same functions as full-size BIOS, except that the clock setting can be skipped at startup.

Warning Specifications

TRR Invalid Address

This outputs a warning message when the address that is referenced during the execution of an LDC instruction is outside the application area. "Outside the application area" is defined as an address that is higher than the last address of the HEX file that was loaded.

XRAM Invalid Address

This outputs a warning message when an access is made using an XRAM address in a memory area that is not implemented. A warning message is also output when bank 3, which does not exist in XRAM, is specified.

Stack Guard

This switch monitors the value of the stack pointer (SP). The monitoring area is specified as a starting and ending value in the text boxes. In the case of an application where the depth of the stack is important, this item can be set in order to output warning messages.

Caution

If the Visual Memory Simulator is reset, the system BIOS sets "7FH" in SP. Because the data is stored in the stack after the SP is incremented, the actual data is processed starting from 80H, heading up to 0FFH.

Break On Warning Above

The virtual CPU does not stop program execution when the above warning messages are output. Check this checkbox in order to stop program execution when a warning message is output.

Break On SFR Invalid Access

A warning message is always output in the event of an invalid access to the Special Function Registers. Check this checkbox in order to stop program execution when an invalid access is made to the Special Function Registers.

Initial Panel Location

This sets the panel display position. [Center] displays panels in the center of the screen. [Custom] stores the position where the user has moved a panel.

Others

Display Hints

This displays hints that have been set up for each GUI control. If this box is checked, hints are displayed; if this box is not checked, hints are not displayed.

Save Setting On Exit

This specifies whether or not to save application environment information when exiting the Visual Memory Simulator.

4.9.2 Work Settings

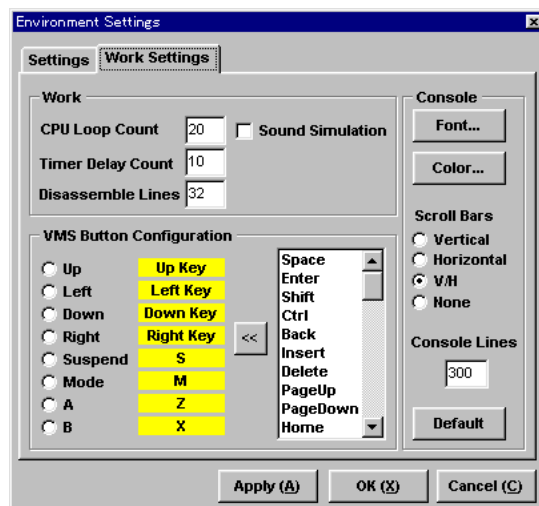


Fig. 4-41

Work Settings

CPU Loop Count

This value determines how many instructions the virtual CPU will execute during one system idle process called from Windows. Increasing this value causes the virtual CPU to run faster. If the results of instruction execution are being drawn at the Simulator level, etc., the graphics speed becomes a limiting factor, so that setting a large value for the loop count will have little effect. On the other hand, a large value tends to slow down message handling in windows, with the result that GUI control response becomes sluggish. The operating speed is also affected by the clock speed of the computer on which the Simulator is running, which is another factor that should be taken into account in order to set this value to a suitable level. The default setting is "20."

Timer Delay Count

The timer delay count value is used to adjust the clock to the virtual Visual Memory timer.

The counter for the timer is started after "n" instructions have been executed. "n" is the timer delay count value.

In other words, this value represents the delay before the timer starts operating. If this value is large, the timer slows down. The default setting is "10."

Disassemble Lines

This specifies the number of lines in the disassemble list. This setting is valid when the [Length] checkbox for the Main Window is checked. The default setting is "32."

Sound Simulation

Because the actual hardware needed for PWM output is not available in the Virtual Memory Simulator, the Simulator is not able to output an accurate frequency. When PWM output becomes possible at the Visual Memory Simulator level, the "PWM.WAV" file will be played. This checkbox is used to enable the playback of "WAV" files. When this box is checked, playback is enabled; when this box is not checked, playback is not enabled.

VMS Button Configuration

The settings for the keys that are allocated as the Visual Memory Image buttons can be changed.

Starting from the left, this group consists of option buttons for selecting the Visual Memory buttons, the name of the key that is currently selected, the setting button, and the setting candidate list box.

Select the Visual Memory button that you wish to set from among the option buttons.

Next, select the key to be set from the list box, and then press the setting button [<<]. You can also double-click on the key in the list box. The key that was set is displayed in yellow.

Console

This group sets the font, color, scroll bar, and other options for the system console.

Font Button

This specifies the character font that is used on the system console. Clicking this button causes the font dialog box to appear. Set whichever font is desired.

Caution

Although vertical fonts are available in the font dialog box, do not specify any of those fonts.

Color Button

This specifies the background color of the system console. Clicking this button causes the color dialog box to appear. Set whichever color is desired.

Scroll Bars

This provides options for the display of scroll bars on the system console.

Vertical	Displays vertical only
Horizontal	Displays horizontal only
V/H	Displays both vertical and horizontal
None	Does not display scroll bars

Console Lines

This specifies the number of lines that are buffered for the system console. The maximum value is 1000 lines. The default setting is 300 lines. Increasing the number of lines increases the load caused by scrolling.

Default Button

This button returns the system console settings to their default settings.

Chapter 5 **Networking**

Two Visual Memory units can be connected to each other through their serial interfaces (SIO). With the Visual Memory Simulator, an equivalent setup can be created by connecting two Simulators through TCP communications.

Although only the various SIO registers are visible from the virtual CPU, data can be transferred to SIO of the other Visual Memory Simulator through the network in response to a transfer request.

The network is controlled through the Network Monitor Window. One of the Visual Memory Simulators is designated as the client, and the other as the server. Because the network connection is not established automatically, it must be established beforehand by using the Network Monitor Window.

Both a client and a server are required for connection. It does not matter which Visual Memory Simulator is the client and which is the server, but it is not possible to have a connection between two clients or two servers.

Start up two Visual Memory Simulators in one PC.

In the Network Monitor Window of the Simulator that will be the client, enter the name of the PC or the IP address as the name of the remote host. Put the server Simulator into the standby state, and then make the connection from the client side.

Start up separate Visual Memory Simulators in different PCs.

Set one of the PCs as the server, and put the Simulator into the standby state. On the client side, enter the name of the server PC or the IP address as the name of the remote host, and then make the connection.

Disconnecting the Network

Although the client and the server can both request disconnection, the disconnection request is usually issued by the client.

Chapter 6 Related Files

This section describes the files that the Visual Memory Simulator references.

6.1 System Files

The system files that the Visual Memory Simulator references reside in the "Files" folder.

VMS.INI

This file contains the initial settings for the Visual Memory Simulator. Modifying this file could cause the Visual Memory Simulator to operate incorrectly. The Visual Memory Simulator cannot start up without this file.

VMS.ENV

This file contains the environment settings that have been made by the user. This file is updated when the Visual Memory Simulator is exited.

DEFAULT.ENV

This file contains initial settings for applications to reference if they do not have their own environment file.

FBIOS.SBF

This file contains the ROM image of the system BIOS that is stored into the Visual Memory unit. The system BIOS is started up whenever Visual Memory is reset. Applications are called from the system BIOS, and when an application is exited, control returns to the system BIOS. The system BIOS includes various subroutine packages that can be used by applications.

QBIOS.SBF

QBIOS skips the clock setting screen that is displayed when FBIOS starts up. Because the clock setting can be skipped, program verification can be performed immediately during debugging. In all other respects, QBIOS provides exactly the same functions as FBIOS.

PWM.WAV

Because the Visual Memory Simulator cannot guarantee complete real-time operation, PWM sound output is not possible. When PWM output becomes possible at the Simulator level, the PWM.WAV file will be played.

6.2 Application Files

The files that are referenced by applications are described below.

APPFILENAME.H00

This is the application execution file. The files that the Visual Memory Simulator can load as applications are H00 files. An H00 file is created by converting an EVA file that is output by the Linker.

E2H86K.EXE is used to convert EVA files to H00 files. Although E2H86K.EXE outputs both a HEX file and an H00 file, only the H00 file is used by the Visual Memory Simulator.

APPFILENAME.MAP

The MAP file contains the symbols that are output by the Linker. The file format is that of a typical text file. The Visual Memory Simulator loads this file and extracts the necessary symbols. Once symbols are loaded, they can be displayed with labels when disassembled.

The MAP file is loaded automatically after the application is loaded. Therefore, the MAP file must reside in the same folder as the application. However, the MAP file is not required by the Visual Memory Simulator, so its absence has no effect on the Visual Memory Simulator.

APPFILENAME.ENV

Information such as panel positions and settings can be stored for each application in a file with the "ENV" extension that resides in the same folder as the application. The next time that the application is loaded, this file is referenced and the settings are restored. If this file does not reside in the same folder as the application, the execution of the application is unaffected, except that the default settings will be used.

Chapter 7 **Warning Messages**

This section describes the warning messages that are displayed when an application is executed.

Stack Guard> Stack overflow occurred.

If the Stack Guard function has been enabled in the Environment panel, this message appears when the stack pointer has gone outside of the specified range.

SFR> Invalid write (read) of SFR was attempted.

This message appears when an invalid write (read) of a Special Function Register was attempted.

An "invalid access" means that a Special Function Register was accessed by a method that is not permitted for users. For example, this message appears in cases where bit access is permitted but byte access is not, or in cases where reading is permitted but writing is not.

TRR> An invalid address was accessed.

This message appears when an address referenced by an LDC instruction was outside the range of addresses where the application is loaded.

XRAM> XRAM cannot be written in subclock mode.

This message appears when a memory area for which XRAM is not implemented was accessed. The XRAM space exists from 180H to 1FFH, but that does not mean that memory is implemented for that entire area. Addresses in which the lower four bits range from 0CH to 0FH are not implemented.

However, the memory that is implemented in XRAM bank 2 is from 180H to 185H.

LCD> Invalid XRAM bank was accessed.

The XRAM bank specification is made in the XBNK register. The bank number is specified by two binary digits, but the value for bank 3 (which does not exist) can be written to this register. In this case, the Simulator switches the bank to bank 0 and displays this message.

LCD> Invalid STAD value was specified.

This message appears when a value that cannot be set is written in the display address start register for the LCD.

SIO#0> Warning: PORT#1 is not ready.

SIO#1> Warning: PORT#1 is not ready.

SIO uses port 1 for input/output. This message appears when none of the bits in port 1 are set for SIO.

SIO> SIO control register values do not match.

This message appears when the settings in the control registers (SCON0 and SCON1) for two Visual Memory Simulators that are attempting SIO communications do not match, making simulation of SIO communications impossible.

Correct the program so that the settings for the transfer bit length, the LSB/MSB first selection, etc., match for both Visual Memory Simulators.

This chapter explains the application development procedure, from coding the program to checking the program on an actual machine. This section assumes that the application specifications have already been established.

1.1 Writing Source Code

The following declarations must be made at the start of the program:

```
Chip LC868700
World external
Public main
Extern _game_end
```

Because all Visual Memory applications will be stored in flash memory, “external” must be declared in the “world” statement.

When an application is called from system BIOS, address 0000H in flash memory is called. Because “jump main” is written in 0000H by GHEAD.ASM, the application provides the label “main” for entry into game mode. Because “main” is referenced from GHEAD.ASM, the “public” declaration is used.

Conversely, when an application ends, it jumps to “_game_end” in GHEAD.ASM, so the “extern” declaration is used to indicate that this label is external to the application.

When an application calls a flash memory-related BIOS or a clock-related BIOS, the “extern” declaration is used to indicate that “fm_wrt_ex”, “fm_vrf_ex”, “fm_prd_ex”, etc., are external programs.

Next, the structure of the indirect address register for the data segment (DSEG) is defined. The entire data segment is expanded in RAM. Because addresses 0000 to 000FH in RAM are indirect address registers, 16 bytes of RAM should be allocated for these registers, whether or not the application will use the indirect address registers. The area in RAM that can be used by an application starts from address 0010H.

Reference

For details on indirect address registers, refer to the “Visual Memory Hardware Manual.”

Start the code segment (CSEG) from an address higher than 0280H (org 280H). GHEAD.ASM uses 0000H to 01FFH, and the information fork uses 0200H to 027FH (minimum).

Reference

For details on the information fork, refer to chapter 2, “Interfacing between Visual Memory and Dreamcast.”



1.2 Correcting GHEAD.ASM

Once you have written the application source code, it is necessary to correct GHEAD.ASM.

If the application uses interrupts, describe the vector table for the interrupts that are to be used in GHEAD.ASM.

Even if interrupts are not to be used, it is still necessary to define an interrupt vector table. Also write the interrupt handler so that it does nothing except execute “RETI”.

Because the program jumps to the start (0000H) of GHEAD.ASM if the user selects game mode, a jump instruction to the main routine of the game should be written at the start of GHEAD.ASM.

The processing that is to be performed for BIOS calls is described starting from address 100H. Do not change this processing. Because the BIOS in ROM specifies addresses directly and then returns control to flash memory, BIOS calls will not be made correctly if addresses change by even one byte.

Although it is necessary to set the system clock to RC oscillation (1/6 division) when reading/writing flash memory in particular, the change should be made within the application program, then “fm_wrt_ex”, “fm_vrf_ex” or “fm_prd_ex” should be called, and then the clock should be switched to crystal oscillation when control returns.

Also be careful not to change the “org” instruction that specifies each BIOS start address.

1.3 Assembly Without Using MAKE

This section explains how to assemble and link the source code, and then build a file in a format that can be actually executed in Visual Memory.

Caution

The Assembler and Linker use EMS. Before starting, display the MS-DOS prompt properties and enable EMS memory usage in the “EMS Memory” group under the “Memory” tab.

EMS memory cannot be used when EMM386.EXE is embedded in CONFIG.SYS and the NOEMS option is specified, so the NOEMS option must be removed.



1.3.1 Assembly

Execute the “M86K” command from the MS-DOS command prompt to assemble the source code.

For example, if the source code file name is “TEST.ASM”, set the current drive and the current directory to the directory where “TEST.ASM” resides, and then perform the assembly process by executing the following command:

```
C>M86K TEST.ASM

SANYO (R) LC86K series Macro Assembler Version 4.0K
Copyright (c) SANYO Electric Co., Ltd. 1989-1995. All rights
reserved.

Pass 1 .....
Source file:      TEST
Chip name:        LC868700
ROM size:         60K bytes
RAM size:         512 bytes
XRAM size:        196 bytes
Pass 2 .....
```

When assembly is completed, an object file with the extension “.OBJ” is created.

Assembling GHEAD.ASM in the same manner creates GHEAD.OBJ.

If an error message similar to the following appears during the assembly process, a problem exists in the line indicated by the line number in the message.

```
Pass 1 .....
TEST.ASM(93):                move                #080h,b
** Error, syntax error near #
0 warning(s) and 1 error(s) were detected. Further execution aborted.
```

Correct the source code so that no warnings or errors are generated.

Reference

For details on the M86K assembler's warning messages and error messages, refer to “Visual Memory Programmer's Manual.”

Caution

Except when incorporating source code equivalent to GHEAD.ASM into your own source code, GHEAD.OBJ and the user program object file are both required. Note that interrupt vectors, interrupt service routines, BIOS call programs, etc., are described in GHEAD.ASM, and are placed in addresses below the user program by the Linker that is executed next.

1.3.2 Linking

After preparing an object file (created by the Assembler) and a GDUMMY.OBJ file that indicates the addresses in internal ROM where BIOS is written, use the Linker to create an EVA-format file.

Note

An EVA-format file is a file that uses special debugging hardware. Because the Visual Memory Simulator is used in the development of applications for Visual Memory, think of the EVA file as a temporary file.



Before executing the linker, make a note of the GDUMMY.OBJ path. Then input the following command line to link each of the object files.

```
D>L86K GHEAD.OBJ IFORK.OBJ TEST.OBJ -C=200
C:\VM_SDK\LC86K\OBJ\GDUMMY.OBJ,
TEST.EVA,,,

SANYO (R) LC86K series Linkage Loader Version 6.00c
Copyright (c) SANYO Electric Co., Ltd. 1989-1997. All right reserved.

Pass 1 ...
Pass 2 ...
Pass 3 ...

Link process complete !!
TEST.EVA created
```

The option “-C=200” specifies the address in flash memory where the user program that is specified immediately afterwards is to be placed.

If an error message is displayed, review GHEAD.ASM and the user program. Check the labels that are used for BIOS calls in particular.

1.3.3 Converting an EVA File Into a HEX File

The Linker combines all of the object files into a single file with the “.EVA” extension. The next step is to convert this file into a file that can be loaded into Visual Memory or the Visual Memory Simulator. Input the following command line.

```
D>E2H86K TEST.EVA
SANYO LC86000 Series EVA-file to HEX-file generator V1.21A
Copyright (C) SANYO Electric Co.,Ltd. 1992-1997

EVA file name: TEST.EVA
ROM data packed: FF(hex)
Chip name: LC868716

All ROM(64KB) block records: 03875
All ROM(64KB) block records: 04096
Module name: GHEAD External CSEG(In) 0000 - 0002 records:
00001
Module name: External CSEG(In) 0003 - 0004 records:
00001
Module name: External CSEG(In) 000B - 000C records:
00001
Module name: External CSEG(In) 0013 - 0014 records:
00001
Module name: External CSEG(In) 001B - 001C records:
00001
Module name: External CSEG(In) 0023 - 0024 records:
00001
Module name: External CSEG(In) 002B - 002C records:
00001
Module name: External CSEG(In) 0033 - 0034 records:
00001
Module name: External CSEG(In) 003B - 003C records:
00001
Module name: External CSEG(In) 0043 - 0044 records:
00001
Module name: External CSEG(In) 004B - 0057 records:
00002
Module name: External CSEG(In) 0100 - 0105 records:
00001
```



```

Module name:      External CSEG(In)    0110 - 0115  records:
00001
Module name:      External CSEG(In)    0120 - 0125  records:
00001
Module name:      External CSEG(In)    0130 - 013B  records:
00001
Module name:      External CSEG(In)    01F0 - 01F4  records:
00001
Module name: IFORK External CSEG(In)    01F5 - 0474  records:
00041
Module name: TEST External CSEG(In)    0475 - 051C  records:
00011

```

There are no option switches.

Executing this command results in the creation of a file with the “.H00” extension and a file with the “.HEX” extension.

Extension	Description
H00	This file can be loaded into the Visual Memory Simulator. The loading time is reduced because only the code itself is saved.
HEX	The 64K-byte image of bank 0 in flash memory is stored in this file. No matter how small the program is, a 64K file is created. Whatever portion that is not filled by the program is filled with “00H”. This file can be loaded into the Visual Memory Simulator, but it will not function properly.

Caution

Normally, only the H00 file is used.

Once the H00 file has been created, an operation check is performed in the Visual Memory Simulator. However, because the Visual Memory Simulator does not have the same clock as the actual machine, a timing check is not appropriate.

Divide the debugging phase so that the program logic is checked in the simulator and the timing and speed are checked on the actual machine.

Reference

For details on the Visual Memory Simulator, refer to the “Visual Memory Simulator Guide.”

1.3.4 Converting a HEX File to a Binary File

This procedure uses H2BIN.EXE to convert an H00 file that was created by the E2H86K into a binary file (extension “.BIN”).

Input the following command line.

```
H2BIN TEST.H00 TEST.BIN
```

There are no option switches. The second parameter “TEST.BIN” may be omitted. If it is omitted, the extension “.BIN” is automatically used.

This procedure creates a file that can be loaded into Visual Memory on the actual machine.



1.3.5 Creating a MAKE File

If a MAKE file is created for the MAKE command, it is possible to perform the assembly, linking, and file format conversion processes through batch processing.

If the dependence information, such as which files to insert in which commands and which files are output, is described in the MAKE file and the MAKE command is executed, the command compares the time stamps of the files that are to be inserted and are to be output, and then assembles and links only those files that have been updated.

For details on the MAKE command, refer to the “Visual Memory Programmer's Manual.”

Caution

Because the MAKE command is provided for a variety of development environments, when the computer that you are using has multiple development environments installed, either change the command retrieval path (the environment variable “PATH”) or change the file name of the MAKE command.

The following file is an example of the type of MAKE file to create in order to MAKE the series of procedures described up to this point. For this example, we will assume that file name is “TEST.MAK”.

```
TARGET = test
OBJECTS = ifork.obj test.obj
HEADOBJ = ghead.obj
SYSOBJ = $(TOOL86)\obj\gdummy.obj

.asm.obj:
    m86k $*

$(TARGET).eva: $(HEADOBJ) $(OBJECTS)
    l86k $(HEADOBJ) $(SYSOBJ) $(OBJECTS),$(TARGET).eva,,

$(TARGET).h00: $(TARGET).eva
    e2h86k $(TARGET)

$(TARGET).hex: $(TARGET).h00
    h2bin $(TARGET).h00
```

This MAKE file is built by specifying MAKE as shown below. This assembles and builds the source files that have been updated.

Caution

Specify the “/F” option when executing the MAKE file. Input the command line in this format: “MAKE /F <MAKE file name>”.

```
D>MAKE /F TEST.MAK

SANYO LC86000 Series MAKE Utility Version 1.00A
Copyright (C) SANYO Electric Co.,Ltd. 1993-1994 All rights reserved.

m86k GHEAD

SANYO (R) LC86K series Macro Assembler Version 4.0K
Copyright (c) SANYO Electric Co., Ltd. 1989-1995. All rights
reserved.

Pass 1 .....
```



```

Source file:      GHEAD
Chip name:       LC868700
ROM  size:       60K bytes
RAM  size:       512 bytes
XRAM size:       196 bytes
Pass 2 .....

m86k IFORK

SANYO (R) LC86K series Macro Assembler Version 4.0K
Copyright (c) SANYO Electric Co., Ltd. 1989-1995. All rights
reserved.

```

(Subsequently, the Linker, E2H86K, and then H2BIN are executed and a binary file is created.)

1.4 Creating the Information Fork

In the sample in the Visual Memory SDK, IFORK.ASM is created and then a binary file is created. It is also possible to use a binary editor, etc., to fill the information fork of the binary file that was produced.

The information fork is filled with the icons and application names that are displayed on the Dreamcast file management screen, the comments that are displayed in Visual Memory file mode, the game names (sort keys), and comments that use larger icons.

Of these, the required items are VM comment data, GUI comment data, game names, the number of icons, visual types, and icon information (for a minimum of one icon).

Refer to chapter 2, “Interfacing between Visual Memory and Dreamcast,” while editing the information fork.

1.5 Transferring the Program to Visual Memory

Once editing of the information fork is complete, transfer the file to Visual Memory. The “Memory Card Utility” that is provided in the Visual Memory SDK is used to transfer the program.

Caution

The Memory Card Utility is provided in ELF file format as a Dev.Box application. It is not a program for general-purpose personal computers.

The following hardware and software is needed in order to transfer a program to Visual Memory:

- 1) An RS-232C cross cable
- 2) A communications program that runs under Windows
- 3) A debugger, such as CodeScape
- 4) GD Workshop
- 5) Dev.Box (Set 5.2X or later)



Note that items 1) and 2) are not provided in our SDK, and must be obtained separately.

Reference	For details on the transfer method, refer to chapter 3, "Memory Card Utility."
------------------	--



index

Symbol	
+5V Supply Signal	20
—	
Access Reference Monitor	42, 46
Address ON/OFF	43
APPFILENAME.ENV	74
APPFILENAME.H00	74
APPFILENAME.MAP	74
—	
Base Timer	58
Break	28
Break Button	31
Break by Address Comparison	42
Break Control	28, 42
Break Mode	43
Break On SFR Invalid Access	68
Break On Warning Above	68
Breakpoints	43
Buttons Connected to Port 3	20
—	
Checking Operation	
on the actual machine	12
Clear Console	29
Client	71
Client mode	61
Client settings	61
Color Button	70
Connecting two Simulators	
through TCP communications	71
Console	61, 70
Console buffer	34
Console Lines	70
Control Signals Connected to Port 7	20
Conversion to a HEX file	22
CPU Control	48
CPU Loop Count	69
CPU Register Display	30

—	
Data Monitor	62
Default Button	70
DEFAULT.ENV	73
Differences in Operating Speeds	5
Disassemble	28, 32
Disassemble Lines	69
Display by STAD Checkbox	60
Display Hints	68
Display Start Address	38
Display When an Interrupt Is Received	42
Dump Function	35
—	
E2H86K.EXE	22
Environment Settings Window	66
EVA	22
External INT	56
—	
FBIOS.SBF	21, 73
Flash Memory	18
FLASH#0	38
Font Button	70
Full-size BIOS	21
—	
GAME.BIN	14
Get Screen Button	59
Grid Button	59
Group ON/OFF	42
—	
HEX	22
Hexadecimal Input Pad	28, 64

—	
---	--

I/O Ports	19
Initial Panel Location	68
INT Control	50
Interrupt Controller	19
Interrupt Handler	63
Interrupt Report	45

—	
---	--

LCD	49
LCD Bit Image Display	39
LCD Controller	19
LCD resolution	39
LCD Snapshot	28
LCD Snapshot Window	59
LCDC	19
Load Application	66
Load System File.....	67
Loading and Executing Applications.....	22
Loading the System BIOS.....	21

—	
---	--

Main Program	63
Main Window	26
MAP File.....	22
Memory Control	28
Memory Control Window.....	35
Memory Fetch Break	44
Memory Fetch Break With Range Specification	44
Memory Usage Rate	37
Minimum Size	66

—	
---	--

Network Monitor	28
Network Monitor Window	60
Notes Concerning Startup for the First Time.....	13

—	
---	--

Open Application	26
Open FLASH#1	27
Open RAM File.....	27
Open System File	26
Operating clock for the virtual CPU.....	17
Operating Clock for Timers.....	19
Operating Environment.....	11
Operating speed of the virtual CPU	5
Others	68

—	
---	--

P70.....	20
P71	20
P72.....	20
P73.....	20
Port 1	19
Port 3	19
Port 7	19
PORT1.....	54
PORT3.....	55
PORT7.....	55
Print	27
PWM output.....	69
PWM.WAV	73

—	
---	--

QBIOS.SBF	21, 73
Quick start BIOS	21

—	
---	--

RAM Area	18
RAM#0	36
RAM#1	36
Register Dump	31
Re-open Application.....	26
Reset	28
Reset Button.....	31
Resuming program execution	22
ROM Area.....	18
Run After Loading	66
Run Button	31
Run/Continue	28

—	
---	--

Save Button.....	59
Save Console to File.....	27

index

Save FLASH#1	27	Tracing an interrupt handler only	63
Save RAM File	27	Trigger	63
Save Setting On Exit	68	TRR Fetch Break	43
SBF	21	TRR Invalid Address	67
Scroll Bars	70		
Serial Interface	19	Update Button	35
Server	71	Usr button	31
Server mode	61		
Server settings	61	Virtual CPU	17
Settings	66	Visual Memory has not been formatted	13
SFR	40	VMS Button Configuration	69
SFR Display	28	VMS Special	57
SIO	19, 53	VMS.ENV	73
Skipping the clock setting procedure	73	VMS.INI	73
Sound Simulation	69	Voltage Drop Signal	20
Special Function Register		VTRBF	41
Control Window	47		
Special Function Register Window	28	Warning Messages	75
Special Function Registers	40	Warning Specifications	67
Stack Guard	67	Work RAM	18, 41
Startup Settings	66	Work Settings	68, 69
Status Lamp	33		
Step Button	31	XRAM	18, 39
Step Execution	28	XRAM Invalid Address	67
Stopping a program	22		
Stopping Client operation	61	Zoom Button	59
Stopping Server Operation	61		
Symbol names for disassembly	22		
Symbol Search	65		
Synchronous Display Function	35		
Sys button	31		
System Console	34		
System File Setting	67		
System Files	73		
TCP communication	19		
Timer	19		
Timer 0	51		
Timer 1	52		
Timer Delay Count	69		
Timer Restrictions	19		
Toolbar	29		
Trace Area	63		
Trace Console	63		
Trace Level	63		
Trace Mode Checkbox	62		
Trace Panel	28, 62		