

CMOS SINGLE CHIP MICROCOMPUTER
S5U1C88000Q Manual

S1C63/S1C88 Family Embedded System Simulator Package Ver.7

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. This material or portions thereof may contain technology or the subject relating to strategic products under the control of the Foreign Exchange and Foreign Trade Law of Japan and may require an export license from the Ministry of Economy, Trade and Industry or other approval from another government agency.

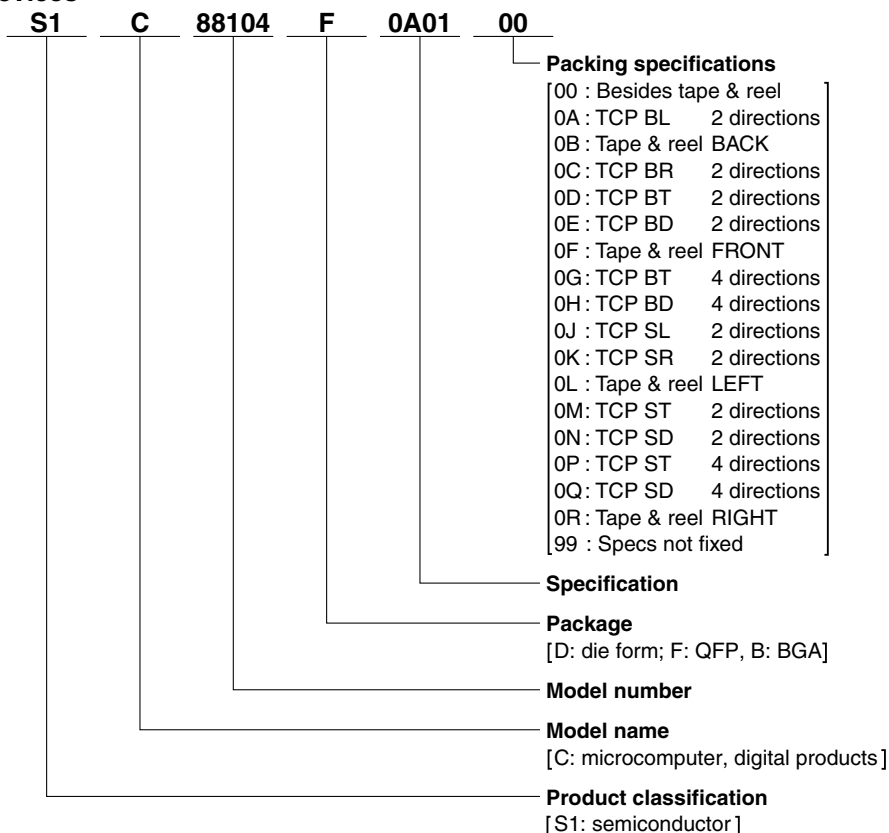
Windows 2000, Windows XP and Windows vista are registered trademarks of Microsoft Corporation, U.S.A.

PC/AT and IBM are registered trademarks of International Business Machines Corporation, U.S.A.

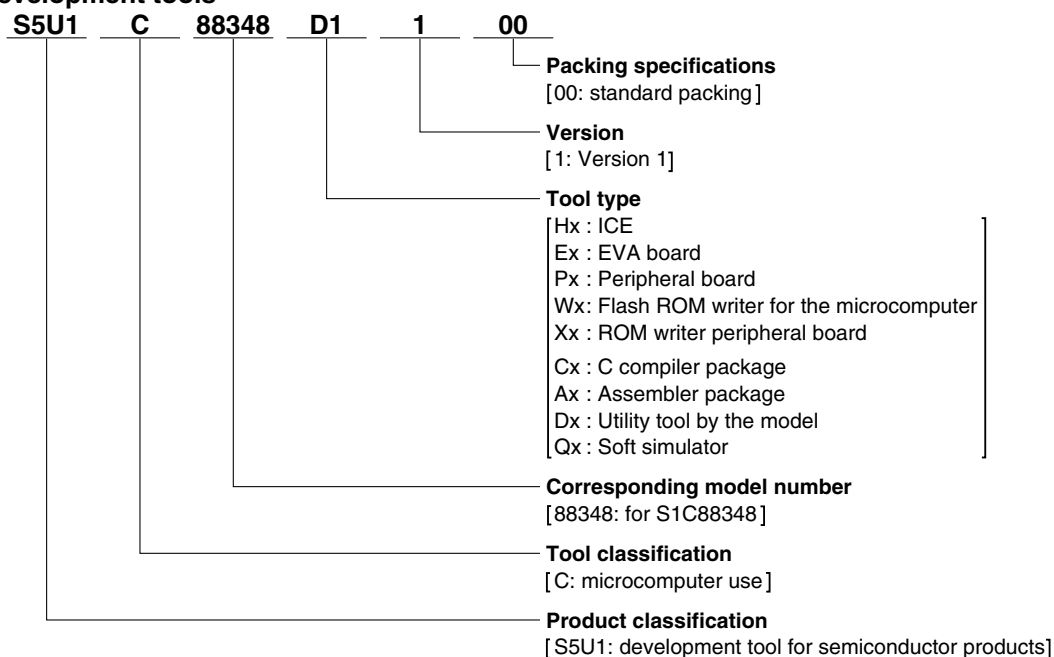
All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

Configuration of product number

Devices



Development tools



CONTENTS

1	OVERVIEW OF THE SIC63/SIC88 FAMILY SIMULATOR PACKAGE	1
1.1	Overview	1
1.2	Overview of Tools	2
2	INSTALLATION	3
2.1	Operating Environment	3
2.2	Installation Method	4
2.3	Directories and Files after Installation	5
3	SIC63 FAMILY SIMULATOR	7
3.1	Overview	7
3.2	Software Development Flow	8
3.3	Overview of Simulation Functions and Operations	9
3.4	Input/Output Files	11
3.5	Starting and Terminating the Simulator	13
3.6	Windows	14
3.6.1	Basic Structure of Window	14
3.6.2	[Command] Window	15
3.6.3	[LCD] Window	16
3.6.4	[I/O Terminal] Window	18
3.6.5	[Source] Window	21
3.6.6	[Data] Window	23
3.6.7	[Register] Window	24
3.6.8	[Trace] Window	25
3.7	Menu	26
3.8	Tool Bar	29
3.9	Method for Executing Commands	30
3.9.1	Entering Commands from Keyboard	30
3.9.2	Executing from Menu or Tool Bar	32
3.9.3	Executing from a Command File	33
3.9.4	Log File	34
3.10	Debug Functions	35
3.10.1	Loading Files	35
3.10.2	Source Display and Symbolic Debugging Function	36
3.10.3	Displaying and Modifying Program, Data, Option Data and Register	38
3.10.4	Executing Program	40
3.10.5	Break Functions	44
3.10.6	Trace Functions	47
3.11	Command List	49
3.12	Component Mapping File (.cmp)	50
3.13	IOT File (.iot)	51
3.14	Simulator Project File (.spj)	54
3.15	Restrictions	55
3.16	Simulator Messages	56
4	SIC88 FAMILY SIMULATOR	57
4.1	Overview	57
4.2	Software Development Flow	57
4.3	Overview of Simulation Functions and Operations	59
4.4	Input/Output Files	61
4.5	Starting and Terminating the Simulator	63
4.6	Windows	64

4.6.1	Basic Structure of Window	64
4.6.2	[Command] Window	65
4.6.3	[LCD] Window	67
4.6.4	[I/O Terminal] Window	69
4.6.5	[Source] Window	72
4.6.6	[Dump] Window	77
4.6.7	[Register] Window	78
4.6.8	[Symbol] Window	78
4.6.9	[Watch] Window	78
4.6.10	[Trace] Window	79
4.7	Menu	80
4.8	Tool Bar	83
4.9	Method for Executing Commands	84
4.9.1	Entering Commands from Keyboard	84
4.9.2	Executing from Menu or Tool Bar	86
4.9.3	Executing from a Command File	87
4.9.4	Log File	88
4.10	Debug Functions	89
4.10.1	Loading Files	89
4.10.2	Source Display and Symbolic Debugging Function	91
4.10.3	Displaying/Modifying Memory and Register Data	93
4.10.4	Executing Program	95
4.10.5	Break Functions	98
4.10.6	Trace Functions	102
4.10.7	Coverage	103
4.11	Command List	104
4.12	Component Mapping File (.cmp)	105
4.13	IOT File (.iot)	107
4.14	Simulator Project File (.spj)	110
4.15	Restrictions	111
4.16	Simulator Messages	112
5	LCD PANEL CUSTOMIZE UTILITY	113
5.1	Overview	113
5.2	Input/Output Files	113
5.3	Starting and Exiting	113
5.4	Window	114
5.5	Menus and Toolbar	115
5.5.1	Menus	115
5.5.2	Toolbar Buttons	117
5.6	Creating LCD Panel Data	118
5.6.1	Creating a New Panel and Setting the Panel Size	118
5.6.2	Creating Icons	119
5.6.3	Creating a Dot (pixel) Matrix	123
5.7	Precautions	125
6	BITMAP UTILITY	126
6.1	Overview	126
6.2	Input/Output Files	126
6.3	Starting and Exiting	126
6.4	Window	127
6.5	Menus and Toolbar	129
6.5.1	Menus	129
6.5.2	Standard Toolbar Buttons	131
6.5.3	Bitmap Edit Toolbar Buttons	132

6.6	<i>Creating Bitmap Data</i>	133
6.6.1	<i>New Data Wizard</i>	133
6.6.2	<i>Creating Bitmap Images</i>	138
6.6.3	<i>Editing Functions</i>	140
6.7	<i>Assembly Source File</i>	146
6.8	<i>Precautions</i>	149
7	<i>PORT SETTING UTILITY</i>	150
7.1	<i>Overview</i>	150
7.2	<i>Input/Output Files</i>	150
7.3	<i>Starting and Terminating PrtUtil</i>	150
7.4	<i>Window</i>	151
7.5	<i>Menu</i>	152
7.6	<i>Creating Port Setting Data</i>	153
7.6.1	<i>Creating New Data</i>	153
7.6.2	<i>Editing the Existing Data</i>	153
7.6.3	<i>Setting Key Matrix Data</i>	154
7.6.4	<i>Setting Push Key Data</i>	158
7.6.5	<i>Setting Target Key Names</i>	160
7.6.6	<i>Printing</i>	162
7.7	<i>Precautions</i>	163
7.8	<i>Port Setting File (.prt)</i>	164
8	<i>LOGIC ANALYZER</i>	166
8.1	<i>Overview</i>	166
8.2	<i>Input Files</i>	166
8.3	<i>Starting and Terminating LogAna</i>	166
8.4	<i>Menus and Toolbar</i>	167
8.4.1	<i>Menus</i>	167
8.4.2	<i>Toolbar Buttons</i>	167
8.5	<i>Using LogAna</i>	168
8.6	<i>Precautions</i>	169
9	<i>ROM DATA SETTING UTILITY</i>	170
9.1	<i>Overview</i>	170
9.2	<i>Input/Output Files</i>	170
9.3	<i>Using Rom88</i>	170
9.4	<i>Precautions</i>	171

1 OVERVIEW OF THE S1C63/S1C88 FAMILY SIMULATOR PACKAGE

1.1 Overview

The S1C63/S1C88 Family Embedded System Simulator Package is a development tool package for the S1C63 Family and S1C88 Family microcomputers. The simulator included in this package allows you to debug S1C63/S1C88 programs created using just a PC, without an in-circuit emulator (ICE) or other dedicated hardware. In addition to providing general debugging functions, the simulator simulates push-buttons or a key matrix that use I/O ports and LCD displays. The package includes utilities for creating bitmap and LCD panel data.

1.2 Overview of Tools

The following provides an overview of the software tools included in the package.

Simulator (Sim63.exe, Sim88.exe)

This software simulates device operations and debugging on a PC. Various functions are executed using commands entered from the keyboard or from files. Frequently-used break and single-step commands are registered to the toolbar to minimize repetitive keyboard tasks. Program code, register contents, and command execution results can be displayed in a multi-window screen to facilitate debugging. You can also simulate push-buttons or key matrix that use I/O ports and LCD displays. A different simulator is provided for each of the S1C63 Family (Sim63) and S1C88 Family (Sim88) microcomputers.

LCD panel customize utility (LcdUtil.exe)

Common to S1C63/S1C88 Family

This utility creates a panel layout and COM/SEG port assignment data required by the simulator (Sim63/Sim88) to simulate a monochrome LCD panel display. Icons and other display objects are loaded from bitmap files (.bmp) and may be created with general-purpose paint software, enabling simulation of the actual product screen.

Bitmap utility (BmpUtil.exe)

Common to S1C63/S1C88 Family

This utility creates bitmap image data (e.g., character data) for the dot matrix LCD display. The output data is created in the assembly source format with specified labels assigned to it to allow you to assemble data without modifications and link it to the program.

Port setting utility (PrtUtil.exe)

Common to S1C63/S1C88 Family

This utility creates port setting data for the simulator (Sim63/Sim88) to simulate target key events using the PC keyboard.

Logic analyzer (LogAna.exe)

Common to S1C63/S1C88 Family

This utility loads a log file created in the [I/O Terminal] window of the simulator (Sim63/Sim88) and displays the I/O port status recorded in the file as a waveform.

ROM data setting utility (Rom88.exe)

Only for S1C88 Family use

This utility loads a ROM image data file in binary format to the simulator (Sim88). It can also save the memory data patched in the Sim88 [Dump] window to a ROM image data file.

2 *INSTALLATION*

This chapter describes the installation and operating environment for the tools included in the package.

2.1 *Operating Environment*

The following operating environment is required to use the S1C63/S1C88 Family Embedded System Simulator Package.

Personal computer

An IBM PC/AT or compatible, having a 400-MHz Pentium or faster CPU and 64 MB or more of RAM.

- * As an index of performance when using the personal computer with the condition above, Sim63 can simulate real-time execution of an application with a 0.5 MHz OSC3 clock, and Sim88 can simulate an application with a 1 MHz OSC3 clock.

Display

Minimum 800 × 600 display resolution (SVGA)

Hard disk

Installation of this package requires a minimum of 50 MB of free hard disk space (more space is highly recommended).

System software

To operate, the package requires that Windows be pre-installed. (For details, refer to Readme_E.txt of S5U1C88000Q.)

Other

In addition to this package, development of S1C63 Family applications requires the S1C63 Family Assembler Package. For S1C88 Family applications, the S1C88 Family Integrated Tool Package is required.

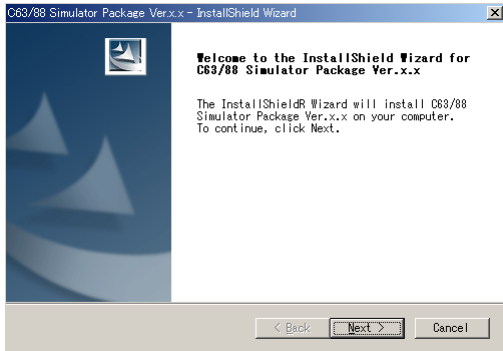
2.2 Installation Method

To install the development tools, use the installer (Setup.exe).

To install the tools



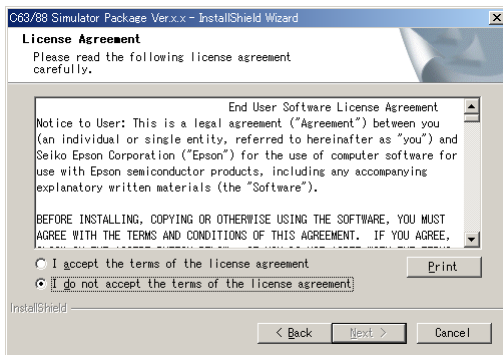
- (1) Start Microsoft Windows. If the OS is already active, close active programs.
- (2) Download the S5U1C88000Q compressed file from the user's site of Seiko Epson, and decompress it into an arbitrary folder.
- (3) Double-click Setup.exe.



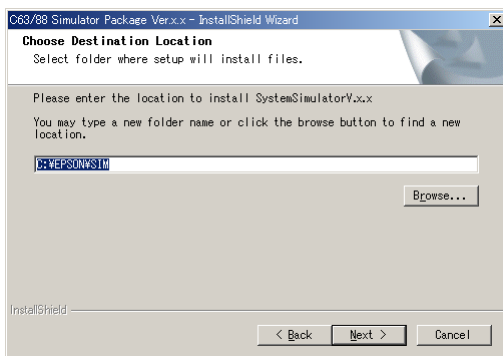
Welcome to ...

The install wizard starts and displays the welcome dialog box.

- (4) Click on the [Next>] button to proceed. The End User Software License Agreement window appears. Be sure to read the license agreement before installing the software.



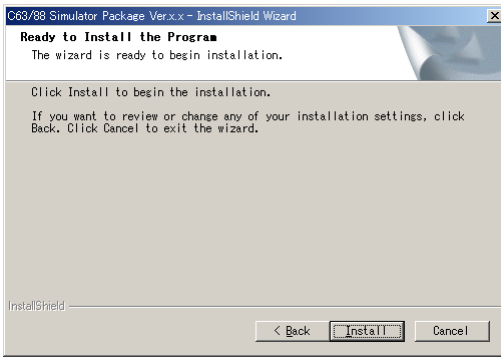
- (5) If you accept the terms of the license agreement, select "I accept the terms of the license agreement", and click the [Next>] button. If you do not accept the terms, select the [Cancel] button to exit the installer.



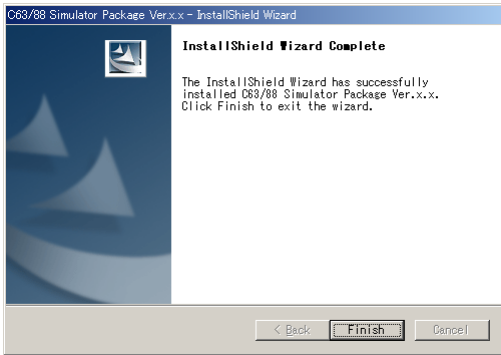
Choose Destination Location

A dialog box for specifying the folder in which to install the SIC63/SIC88 Family embedded system simulator package appears.

- (6) Check the installation directory that is displayed. To install in another folder, click [Browse...] to bring up the [Choose Folder] dialog box. Select the installation folder from the list in the dialog box, or enter the folder path in the [Path] text box and click the [OK] button.
- (7) Click the [Next>] button.



- (8) The installation confirmation window appears. To start installation, click the [Install] button.



InstallShield Wizard Complete

- (9) Click [Finish] to terminate the installer.

To end installation

All dialog boxes that appear during installation have a [Cancel] button. To prematurely terminate installation, click [Cancel] in the dialog box when it is displayed.

To uninstall

To uninstall the installed tools, use "Add/Remove Programs" on the Control Panel.

2.3 Directories and Files after Installation

The installer copies the following files in the specified directory (default is "C:\EPSON\SIM\");

[EPSON\SIM]	
README_E.TXT	...Readme_E text file (English)
README_J.TXT	...Readme_J text file (Japanese)
[S1C63]	...S1C63 Family simulator folder
SIM63.EXE	...Simulator
xxxxxxx.INI	...INI file
xxxxxxx.PAR	...Parameter file
xxxxxxx.BMC	...Component file
:	
.....	...Other related files
[SAMPLE]	...Sample folder
README_E.TXT	...Explanation of sample programs (English)
README_J.TXT	...Explanation of sample programs (Japanese)
SAMPLEx.xxx	...Sample files
:	

2 INSTALLATION

[S1C88]	...S1C88 Family simulator folder
SIM88.EXE	...Simulator
xxxxxxx.INI	...INI file
xxxxxxx.PAR	...Parameter file
xxxxxxx.BMC	...Component file
:	
.....	...Other related file
[SAMPLE]	...Sample folder
README_E.TXT	...Explanation of sample programs (English)
README_J.TXT	...Explanation of sample programs (Japanese)
SAMPLEx.xxx	...Sample files
:	
[UTILITY]	...Utility folder
BMPUTIL.EXE	...Bitmap utility
LCDUTIL.EXE	...LCD panel customize utility
LOGANA.EXE	...Logic analyzer
PRTUTIL.EXE	...Port setting utility
ROM88.EXE	...ROM data setting utility
:	
[DOC]	
[ENGLISH]	...Document folder (English)
REL_xxx_E.TXT	...Release note
TBD_E.PDF	...Manuals (PDF)
:	
[JAPANESE]	...Document folder (Japanese)
REL_xxx_J.TXT	...Release note
TBD_J.PDF	...Manuals (PDF)
:	

Online manual in PDF format

The online manuals are provided in PDF format, so Adobe Acrobat Reader Ver. 4.0 or later is needed to read it.

Files for future release models

The files for future release models will be provided in the Microcomputer User's Site of Seiko Epson. Refer to the Readme file included in the package for installation.

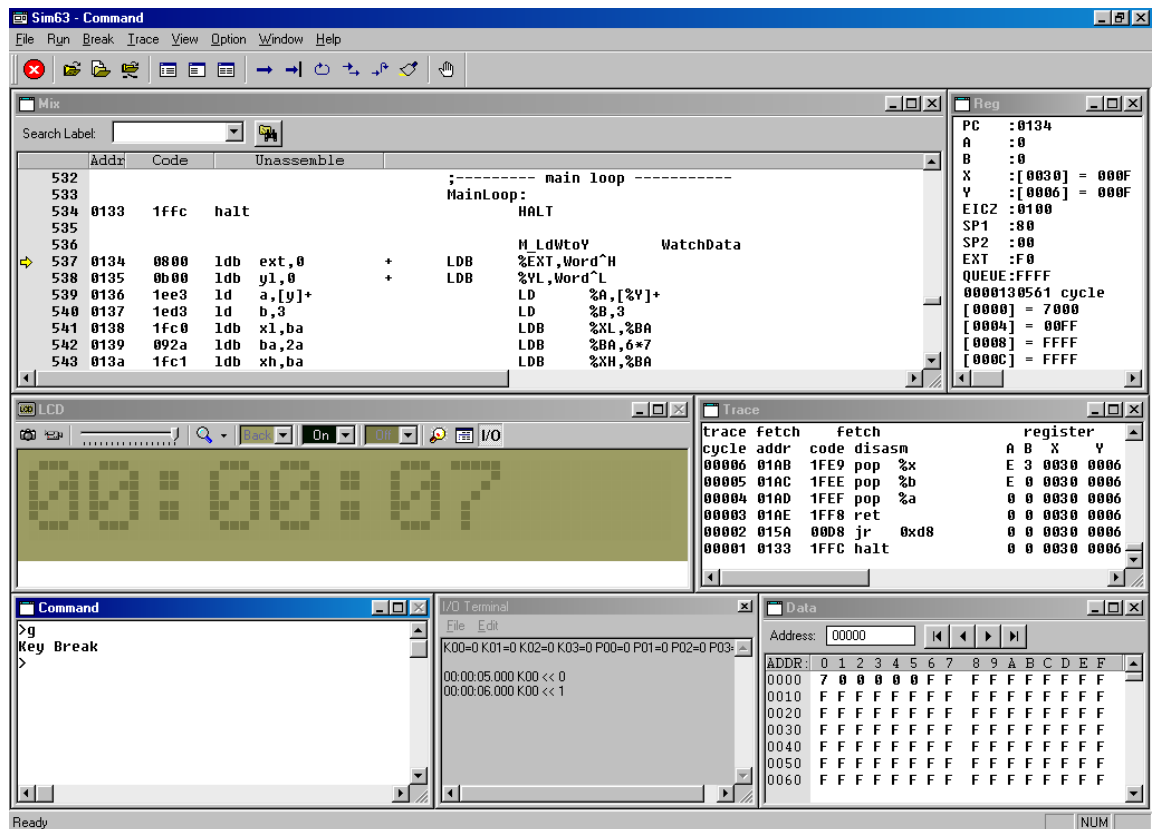
3 S1C63 FAMILY SIMULATOR

3.1 Overview

The S1C63 Family Simulator is a development tool for the S1C63 Family of 4-bit single-chip microcomputers. The simulator included in this package allows you to debug software created with the S1C63 assembler using just a PC, without an in-circuit emulator (ICE) or other dedicated hardware. In addition to providing general debugging functions, the simulator simulates push-buttons or a key matrix that use I/O ports, serial and general port input/output, A/D conversion and LCD displays. The package includes utilities for creating bitmap and LCD panel data.

The simulator has the following features and functions:

- Operations including LCD panel display can be simulated with a PC alone without any debugging hardware.
- Various data can be referenced at the same time using multiple windows.
- Frequently used commands can be executed from tool bars and menus using a mouse.
- Also available are source display and symbolic debug functions which correspond to assembly source codes.
- Consecutive program execution and two types of single-stepping are possible.
- Five break functions are supported.
- Trace function.
- An automatic command execution function using a command file.



3.2 Software Development Flow

Figure 3.2.1 shows the typical software development flow for the S1C63 Family. The items in bold indicate tools and related files provided in this package.

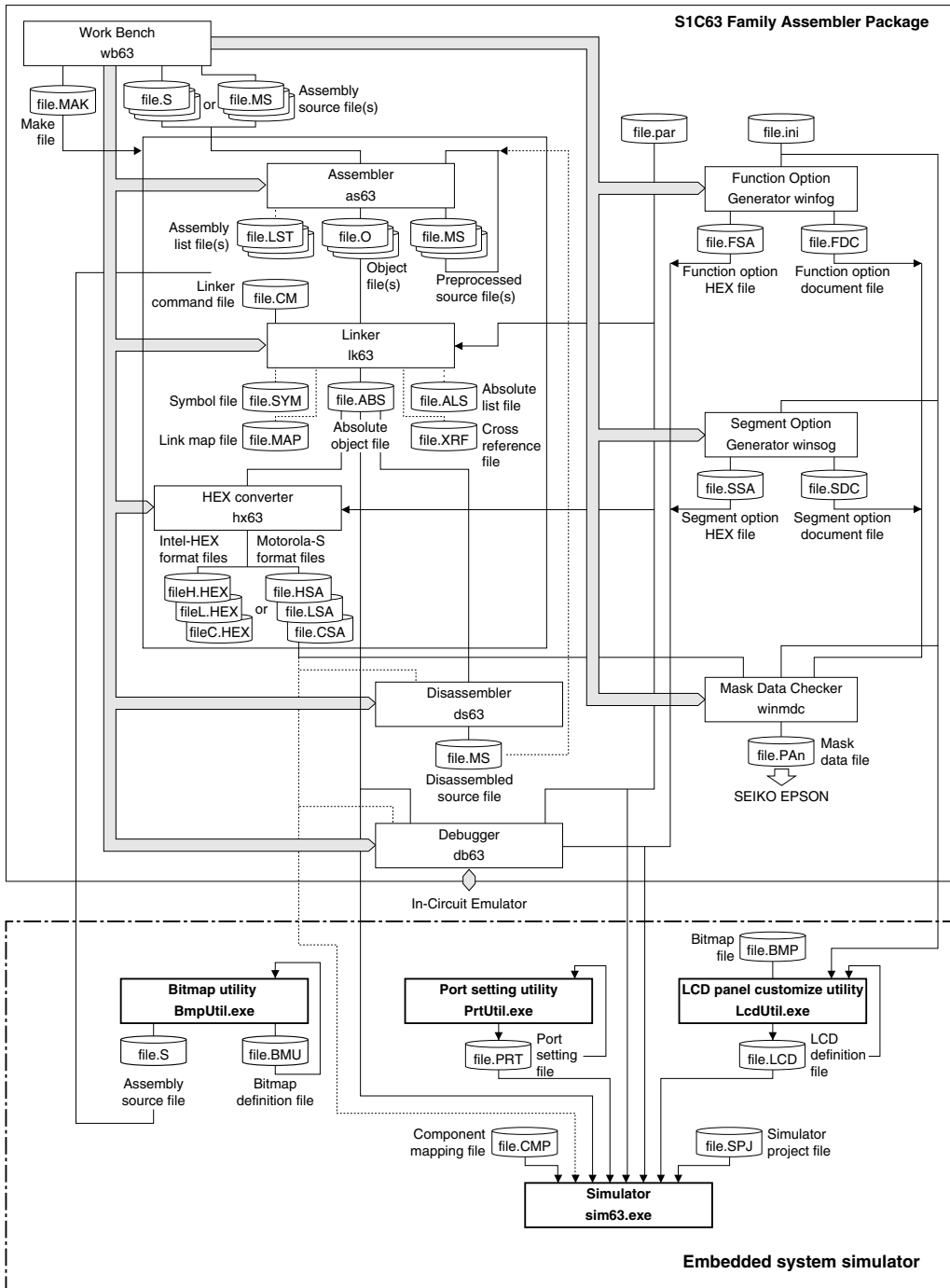


Fig. 3.2.1 Software development flow

Note: In addition to this package, software development for the S1C63 Family requires the S1C63 Family Assembler package as shown above.

3.3 Overview of Simulation Functions and Operations

This section explains how the target application is simulated on a PC with the simulator sim63.

Supported MPUs

For the MPU models supported by the simulator, please refer to the "Supported MCU models and peripherals" described in the `Readme_E.txt`.

Target applications

This simulator is best suited to simulating applications that require key input and LCD display functions, such as watches, pocket calculators, electronic pocketbooks, and portable games.

OSC1 clock operation can be executed in real time. For OSC3 operation, real-time execution is possible in the range of 500 kHz to 1 MHz. (With PCs running on a 400-MHz Pentium or equivalent or faster and 64 MB RAM, real-time execution for approximately 500 kHz is possible.) However, for reasons involving instruction-level simulation accuracy, it is not possible to simulate high-accuracy timing tasks such as those for control systems.

Support for external devices is limited to monochrome LCD panels, backlight, keys, and a key matrix.

Note: Simulation on a PC is subject to some other limitations. The formal release version will support a greater number of external devices. See Section 3.15, "Restrictions", and the `Readme_E.txt` file.

Entering schematic information into the simulator

To set the information required to simulate the operation of external devices, load data from files.

Mapping of external devices and the internally-generated clock frequency

Create a component mapping file (`file.cmp`) by writing addresses to which external devices (e.g. backlight control bit) are mapped, then load the file into the simulator. This allows simulated control of these external devices. Use the component mapping file to set simulation conditions such as OSC1/OSC3 oscillation clock frequencies. For more information on creating this file, see Section 3.12, "Component Mapping File (.cmp)".

Entering key and key matrix specifications

Create a port setting file (`.prt`) containing a description of the relationship between key input ports and the I/O ports comprising the key matrix and target and PC keyboards, then load the file into the simulator. This allows simulated key input with the PC keyboard. For more information on creating this file, see Chapter 7, "Port Setting Utility".

Entering LCD design

Create an LCD panel definition file (`file.lcd`) containing a record of LCD panel layout and SEG/COM port assignments, then load the file into the simulator. This allows simulated display on LCD panels. For more information on creating the LCD panel definition file, see Chapter 5, "LCD Panel Customize Utility".

These files normally are loaded when the simulator starts.

Loading and executing the target program

Use the simulator if the lf command to load the target program and the g (or gr, s, n) command to run it. To simulate the LCD display or external inputs/outputs during program execution, use the simulator's [LCD] and [I/O Terminal] windows.

LCD display: Displayed in the [LCD] window as on an actual LCD.

Key input: Activate the [LCD] window and type the appropriate key on the PC keyboard. You can also set and verify key input status in a dedicated window.

SVD evaluation: Manipulate the SVD slide bar on the [LCD] window to simulate SVD function.

Inputs/outputs of the serial interface and general ports, and A/D conversion:

Activate the [I/O Terminal] window and load an IOT file in which an input sequence is described to simulate the input/output functions.

3.4 Input/Output Files

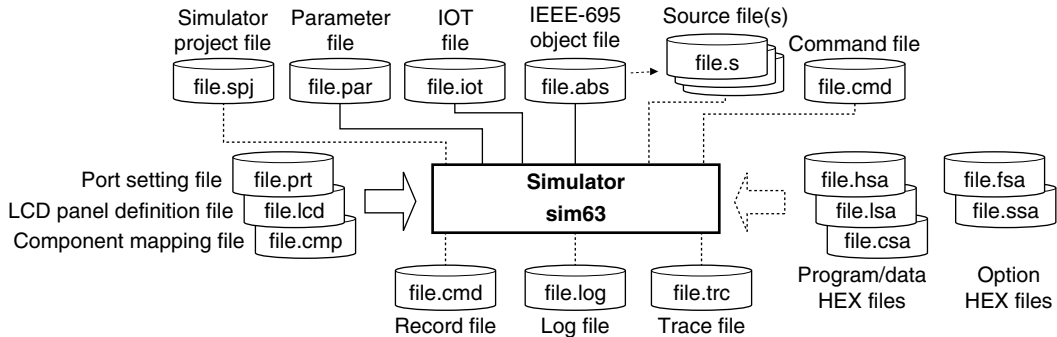


Fig. 3.4.1 Input/output files

Parameter file (file_name.par)

This file contains memory information on each microcomputer model and is indispensable for starting the simulator. This file is included in the S1C63 Family Assembler Package.

Absolute object file (file_name.abs)

This is an object file generated by the linker. This file is read into the simulator by the `lf` command. By reading a file in the IEEE-695 format that contains debug information, source display and symbolic debugging can be performed.

Source file (file_name.s)

This is the source file of the above object file. It is read when the simulator performs source display.

Program HEX file (file_name.hsa, file_name.lsa)

This is a load image file (HEX file in Motorola S2 format) of the code ROM, and is read into the simulator by the `lo` command. The file ".hsa" corresponds to the 5 high-order bits of the program code and the file ".lsa" corresponds to the 8 low-order bits of the program code. These files are generated for the purpose of creating mask data from an object file in the IEEE-695 format by the Hex converter. Unlike files in the IEEE-695 format, these files cannot be used for source display or symbolic debugging, but can be used to check the operation of final program data.

Data ROM HEX file (file_name.csa)

This is a load image file (HEX file in Motorola S2 format) of the data ROM, and is read into the simulator by the `lo` command. This file is generated for the purpose of creating mask data from an object file in the IEEE-695 format by the Hex converter. When an absolute object file in the IEEE-695 format is loaded, it is not necessary to load this file.

Function option HEX file (file_name.fsa)

This is the mask option setup file in Motorola S2 format that is generated by the function option generator. This file is read by the `lo` command.

Segment option HEX file (file_name.ssa)

This is the mask option setup file in Motorola S2 format that is generated by the segment option generator. This file is read by the `lo` command. Some models have no segment option, but a HEX file is provided for setting up the simulator.

Simulator project file (file_name.spj)

This file is used to specify a parameter file, LCD panel definition file, component mapping file and port setting file at the simulator start up. Enter file names using an editor to create this file. The simulator can be started up if this file does not exist by selecting the files from a dialog box.

IOT file (file_name.iot)

This is the text file in which the input data used to simulate serial/general port inputs/outputs and A/D conversions are described.

LCD panel definition file (file_name.lcd)

This file includes an LCD panel layout bitmap and SEG/COM port allocation information. Create this file using the LCD panel customizing utility (LcdUtil).

Component mapping file (file_name.cmp)

This is the text file that sets the addresses where the backlight are mapped.

Port setting file (file_name.prt)

This is the text file in which push keys, key-matrix configuration and the corresponding between the keys and ports are described. Create this file using the port setting utility (PrtUtil).

Command file (file_name.cmd)

This text file contains a description of debug commands to be executed successively. By writing a series of frequently used commands in this file, the time and labor required for entering commands from the keyboard can be saved. The command described in the file are read and executed using the com or cmw command.

Log file (file_name.log)

This text file contains the executed commands and execution results. Output of this file can be controlled by the log command.

Record file (file_name.cmd)

This text file contains the executed commands. Output of this file can be controlled by the rec command. This command can be used as a command file.

Trace file (file_name.trc)

This text file contains the specified range of trace information. Output of this file can be controlled by the tf command.

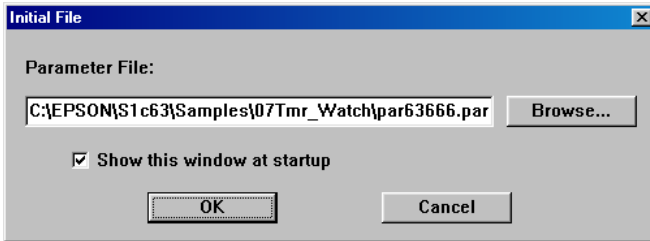
3.5 Starting and Terminating the Simulator



Sim63.exe

Double-clicking this icon to start the simulator.

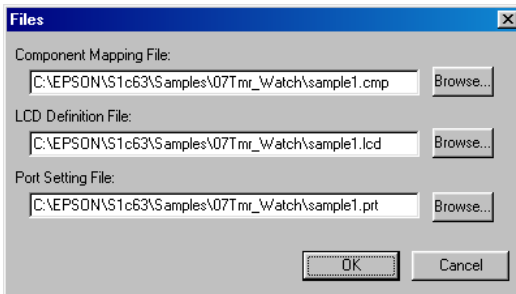
The dialog box shown below appears at the first time the simulator starts up. Enter the parameter file name or simulator project file name to the text box, or choose it using the [Browse...] button (see Section 3.14 for the simulator project file).



If the [Show this window at startup] check box is deselected, this dialog box will not appear from the next startup of the simulator and the same file will be selected. After this, use the par command ([File | Load Parameter File...]) to redisplay this dialog box for changing the file.

When the check box is left on, the currently selected file name will appear in the text box at the next startup of the simulator allowing choose of the file by clicking the [OK] button only.

When a parameter file is selected, the dialog box shown below appears.



Enter the component mapping file name, LCD panel definition file name and port setting file name to the respective text boxes or choose them using the [Browse...] button. This dialog box will appear at the next start-up of the simulator even if the [Show this window at startup] is deselected. However, the selected file names will appear in the text boxes at the next startup of the simulator allowing choice of the files by clicking the [OK] button only.

This dialog box will not appear when a simulator project file is selected.

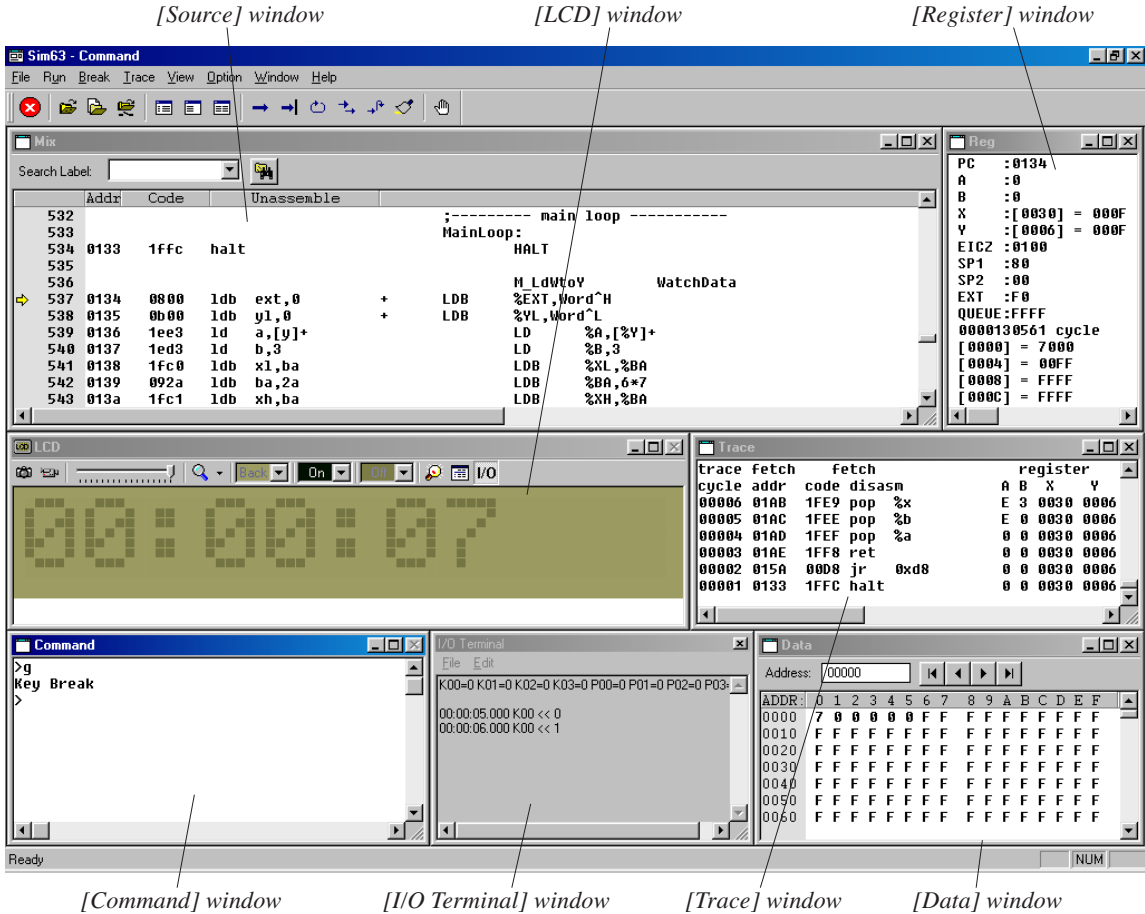
Select [Exit] from the [File] menu to terminate the simulator.

3.6 Windows

This section describes the types of windows used by the simulator.

3.6.1 Basic Structure of Window

The diagram below shows the window structure of the simulator.



Features common to all windows

(1) Open/close and activating a window

All windows except [Command] and [LCD] can be closed or opened.

To open a window, select the window name from the [View] menu. When a command is executed, the corresponding window opens if the command uses the window for displaying the executed results.

To close a window, click the [Close] box on the window.

The opened windows are listed in the [Window] menu. Selecting one from the list activates the selected window. It can also be done by simply clicking on an inactive window. Furthermore, pressing [Ctrl]+[Tab] switches the active window to the next open window.

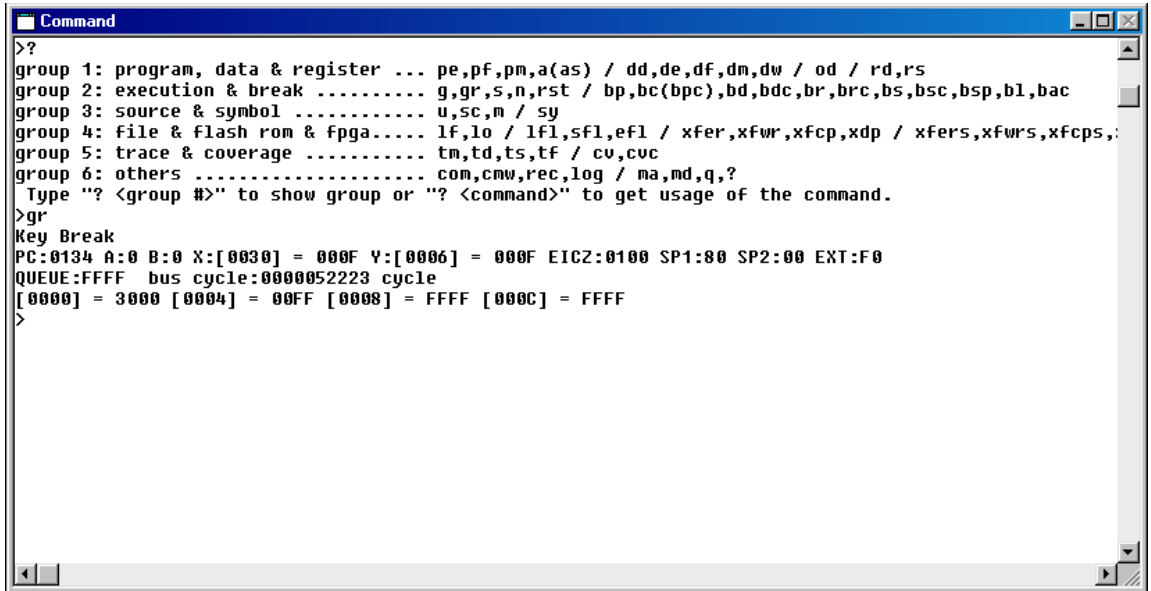
(2) Resizing and moving a window

Each window can be resized as needed by dragging the boundary of the window with the mouse. The [Minimize] and [Maximize] buttons work in the same way as in general Windows applications. Each window can be moved to the desired display position by dragging the window's title bar with the mouse. However, windows can only be resized and moved within the range of the application window.

(3) Other

The opened windows can be cascaded or tiled using the [Window] menu.

3.6.2 [Command] Window



```

Command
>?
group 1: program, data & register ... pe,pf,pm,a(as) / dd,de,df,dm,dw / od / rd,rs
group 2: execution & break ..... g,gr,s,n,rst / bp,bc(bpc),bd,bdc,br,brc,bs,bsc,bsp,b1,bac
group 3: source & symbol ..... u,sc,m / sy
group 4: file & flash rom & fpga.... lf,lo / lf1,sf1,ef1 / xfer,xfwr,xfcp,xdp / xfers,xfwrs,xfcps,:
group 5: trace & coverage ..... tm,td,ts,tf / cv,cvc
group 6: others ..... com,cmw,rec,log / ma,md,q,?
Type "? <group #>" to show group or "? <command>" to get usage of the command.
>gr
Key Break
PC:0134 A:0 B:0 X:[0030] = 000F Y:[0006] = 000F EIC2:0100 SP1:80 SP2:00 EXT:F0
QUEUE:FFFF bus cycle:0000052223 cycle
[0000] = 3000 [0004] = 00FF [0008] = FFFF [000C] = FFFF
>

```

The [Command] window is used to do the following:

(1) Entering debug commands

When the prompt ">" appears in the [Command] window, the system will accept a command entered from the keyboard.

(2) Displaying debug commands selected from menus or tool bar

When a command is executed by selecting the menu item or tool bar button, the executed command line is displayed in the [Command] window.

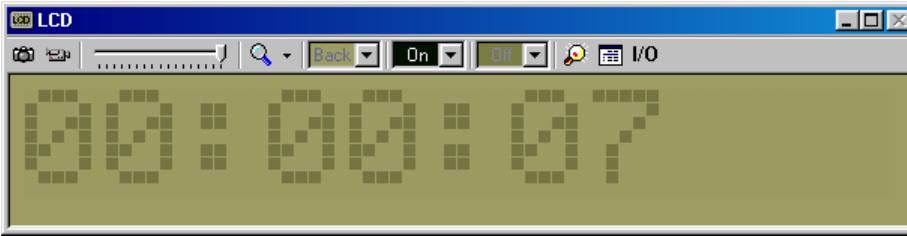
(3) Displaying command execution results

The [Command] window displays command execution results. However, some command execution results are displayed in the [Source], [Data], [Register], or [Trace] windows. The contents of these execution results are displayed when their corresponding windows are open. If the corresponding window is closed, the execution result is displayed in the [Command] window.

When writing to a log file, the content of the write data is displayed in the window. (Refer to the description for log command.)

Note: The [Command] window cannot be closed.

3.6.3 [LCD] Window



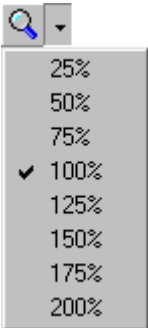
Note: The [LCD] window cannot be closed.

The [LCD] window has the following functions:

(1) LCD display simulation

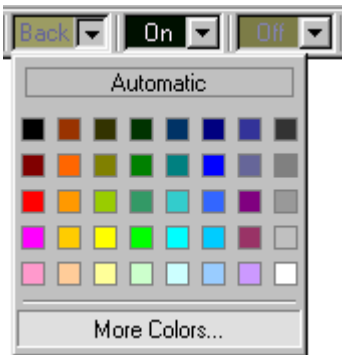
Displays the LCD panel defined in the LCD panel definition file. The icons and dot matrix change their display status according to the program being executed.

Furthermore, the panel size and the color can be set using the following controls.



Scaling button & drop-down list

The panel size is magnified in 25% steps every time the button is clicked. If the size has reached 200%, the next click reduces the size to 25%. The drop-down list allows direct selection of the magnification rate.



[Back], [On], [Off] drop-down lists

Set the LCD panel color.

[Back] Background color

[On] Dot color when it is on

[Off] Dot color when it is off



[Backlight] button

When this button is clicked, the dialog box shown below appears allowing registration of up to 4 backlight colors. The color can be adjusted using the RGB sliders and selecting the lower left check box turns the backlight on.



(2) Capturing the panel image



Camera button

Clicking this button captures the LCD panel image at that point and then transfers it to the clipboard as a bitmap image. The captured image can be pasted to paint software to print and/or saved to a file.

Since this button is invalid during program execution, program execution must be broken at the desired image before it can be captured.



Video button

Clicking this button enables the panel image recording function (can be saved as an AVI file).

When starting the program execution in this status, two dialog boxes to specify a file name and a video compression format appear sequentially. Enter a file name and choose an available video compression format. The recording continues until the program execution is broken.

The generated file can be played back with the Windows standard medium player.

This button cannot be cancelled after it has been clicked once. To cancel the recording, click on the [Cancel] button in the file name input dialog box. The program will be executed, but the recording is cancelled.

Note: The recording operation reduces the program execution speed.

(3) Key entry simulation

If the program being executed is waiting a key entry, key entry operation can be simulated using the PC keyboard after activating the [LCD] window.

The correspondence between PC keys and ports should be defined in the port setting file.

This definition list can be displayed using the [Key List] button.



[Key List] button

Port	Target Key	PC Key
<input checked="" type="checkbox"/> K00	Q	Q
<input type="checkbox"/> K01	W	W
<input type="checkbox"/> K02	E	E
<input type="checkbox"/> K03	R	R
<input type="checkbox"/> K10	T	T
<input type="checkbox"/> K11	Y	Y
<input type="checkbox"/> K12	U	U
<input type="checkbox"/> K13	I	I

The displayed contents are port name, target key name and PC key name, respectively from the left. The correspondence between PC keys and target keys can be verified here.

The symbol on the left indicates the port status; H indicates that the port is high level and L indicates low level. The input level is different by the port specification of the model.

The key entry status can also be set in break status. The key entry status is maintained during the next program execution until the user operates the key. This allows setting of the key entry status during single-step operation.

The input port is fixed at active level if the pressed key is directly connected to the input port. In the case of a key matrix, the input port goes active level only when the corresponding output port goes active level.

(4) SVD simulation

This slider changes the detection level of the SVD to 17 steps.



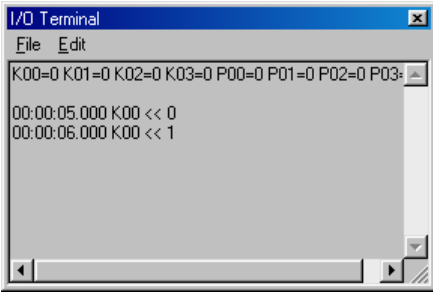
(5) Simulating serial/general-purpose port input/output and A/D conversion

Use the [I/O] button to display the [I/O Terminal] window and to load an IOT file. This allows simulation of serial interface input/output, general-purpose port input/output, and A/D conversion. For more information, refer to the next section.



[I/O] button

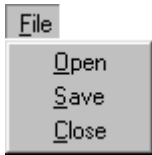
3.6.4 [I/O Terminal] Window



The [I/O Terminal] window provides an I/O text terminal function to display the input/output status of the general-purpose ports, serial interface, and A/D converter by simulating their inputs/outputs.

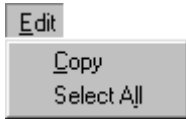
(1) Menu

[File] menu



- Open Loads an IOT file.
- Save Saves the logs displayed in the window to a text file.
- Close Closes the IOT file (halts I/O simulation).

[Edit] menu



- Copy Copies a data range selected within the window to the clipboard.
- Select All All selects all logs in the window.

* The Copy/Select All menu can also be displayed by right-clicking anywhere in the window.

(2) IOT file

The IOT file is a text file that contains a description of the following specifications. It is created in a general-purpose editor.

- a. Specify the general-purpose ports (Rxx, Pxx) to monitor the input/output status.
- b. Specify the input timing to and the input level of general-purpose ports (Kxx, Pxx).
- c. Specify the power supply voltage and the reference voltage of the A/D converter.
- d. Specify the A/D input voltage.
- e. Specify the serial interface input data.

Example:

```
[General I/O]
watch P00, P01, P02, P10, P11, P20  ← a
5.0 K00=1                               ← b  The first value indicates the time in seconds from a reset.
5.5 K00=0                               1 / H = High, 0 / L = Low
8.0 K00=H
8.5 K00=L

[A/D Converter]
AVdd=3.0                                 ← c  The values indicate voltages (V).
AVss=0.2
AVref=2.9

[A/D CH2]
2.40, 2.39, 2.38, 2.37, 2.36             ← d  The values indicate voltages (V).
2.00, 2.10, 2.20, 2.30
0.5
0.65
0.80

[Serial CH1]
"aBcDeFg"                                ← e  Specify the input data using a character string
0x25, 0x20, 0x41                        or hexadecimal value.
```

Load this file to simulate the status of each input pin. For detailed information on this file, refer to Section 3.13, "IOT File (.iot)".

(3) Starting and finishing input/output simulation

Follow the procedure given below to start input/output simulation:

1. Display the [I/O Terminal] window.
2. Load the IOT file by selecting [Open] from the [File] menu.
3. Run the target program from a reset state.

Although the IOT file can be loaded while running the target program, the simulator must be reset before input/output status can be simulated.

To terminate input/output simulation, select [Close] from the [File] menu.

(4) Simulation and log display

When the specified input/output ports change state (from 1 to 0 or vice versa), or the serial interface receives input data or sends output data, or the A/D converter receives analog signals as input, the pertinent information is displayed in the window.

Example:

```

P00=1 R01=1
AVdd=3.000 AVss=0.200 AVref=2.900 ] Initial data

00:00:01.029 S2 << 31h(1)
00:00:02.033 S2 << 32h(2)
00:00:03.036 S1 >> 33h(3)
00:00:04.040 S1 >> 34h(4)
00:00:05.000 P30 << 0 P31 << 0
00:00:06.016 AD0 << 2.400
00:00:07.020 AD0 << 2.390
00:00:08.023 AD0 << 2.380
00:00:09.027 AD0 << 2.370
00:00:05.000 R01 >> 1 ] Log data

```

Immediately after a reset, the simulator displays the initial data. The contents displayed here are the initial values of the general-purpose input/output ports specified with a "Watch" statement in the IOT file's [General I/O] section and the "AVdd", "AVss", and "AVref" values specified in the [A/D Converter] section. These values are not displayed unless they are specified.

Displayed following the above is data input or output during the simulation. The numeric value appearing at the beginning of each line indicates the elapsed time (in hours, minutes, and seconds) since the simulator was reset. The symbols "<<" and ">>" denote input and output, respectively. The data for each port and the contents of processing are described below.

General-purpose input/output ports:

When input or output on the ports specified with a Watch statement changes state, the simulator displays the port names (Kxx, Pxx, Rxx) and the value after the change (0 or 1). Changes in input occur synchronously with the timing written in the IOT file. If interrupts are enabled at this time, an input interrupt is generated according to the interrupt conditions set.

A/D converter:

Each time the target program performs an A/D conversion, the data written in the IOT file is loaded in sequence from the top (left) to simulate the A/D conversions performed. At this time, the simulator displays the channel numbers (ADx) and the input voltages (V) loaded from the file. If interrupts are enabled, an A/D conversion interrupt is generated after a certain period of time has elapsed (sampling time + A/D conversion time) after the start of input. If a converted channel has no specified data, the input on that channel is read out as level 0 voltage (equivalent to AVss). If A/D conversion is performed more than a specified data count, the last data specified is read out repeatedly. If a value smaller than AVss was specified, it is read out as level 0 (equivalent to AVss); if a value larger than AVref was specified, it is read out as the maximum level (equivalent to AVref).

Serial interface:

As the target program performs input/output via the serial interface, the simulator displays the channel numbers (S1, S2) and the input/output values (hexadecimal values and ASCII characters) for each byte transferred. For input, the input data written in the IOT file is read out sequentially from the top (left). If interrupts are enabled, a serial interface interrupt is generated synchronously with the timing at which the program finishes sampling the input.

(5) Saving logs

The logs displayed in the [I/O Terminal] window can be saved to text files using [Save] from the [File] menu.

Furthermore, a range of logs selected by dragging the mouse in the [I/O Terminal] window can be copied to the clipboard using [Copy] from the [Edit] menu. The copied log can be pasted from the clipboard into any document by using an editor. To copy all logs, use [Select All] from the [Edit] menu to select all logs before copying to the clipboard.

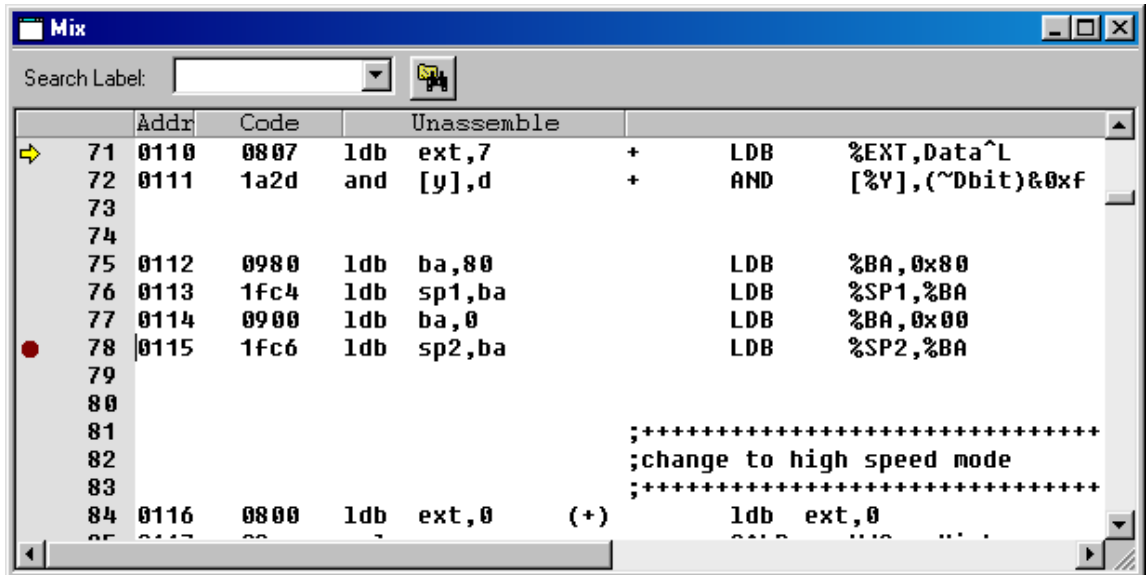
Precaution

- In development environments where the S1C63/88 simulator is already installed, the components corresponding to the serial interface must be added to the component map file (.cmp). When using a model that does not support the serial interface, add NullDev.bmc to the component map file. When using the A/D converter, make any necessary corrections to the component map file.

Example: 63666.cmp	Example: 63458.cmp
[Internal]	[Internal]
CPU=CPU.bmc	CPU=CPU.bmc
LCD=LcdDrv63.bmc	LCD=LcdDrv63.bmc
K/P/R port=KPRport.bmc	K/P/R port=KPRport.bmc
SVD=SVD63.bmc	SVD=SVD63.bmc
Sound=Sound63.bmc	Sound=Sound63.bmc
Serial=Serial63666.bmc ← Add	Serial=Serial63.bmc ← Add
Adc=Adc63.bmc ← Add	Adc=NULLDev.bmc ← Add

- The time at which the simulator simulates input/output operations is calculated from its instruction execution cycles. This time differs slightly from the actual MCU (actual time).

3.6.5 [Source] Window



The [Source] window displays the contents of (1) to (3) listed below. This window also allows breakpoints to be set and words or labels to be found.

(1) Unassembled codes and source codes

You can choose one of the following three display modes:



[Mix] button

1. Mix mode

(selected by the [Mix] button or entering the m command)

In this mode, the window displays the addresses, codes, unassembled contents, and corresponding source line numbers and source statements. (See the diagram above.)



[Source] button

2. Source mode

(selected by the [Source] button or entering the sc command)

In this mode, the window displays the source line numbers and source statements.



[Unassemble] button

3. Unassemble mode

(selected by the [Unassemble] button or entering the u command)

In this mode, the window displays the addresses, codes, and unassembled contents. This format is selected when the debugger starts up.

Note: The m, sc and u commands can update the [Source] window if the window is already opened. If the [Source] window is closed, the program code is displayed in the [Command] window.

All program code in the 64K address space can be referenced by scrolling the window. When a break occurs, the display content is updated so that the address line to be executed next is displayed, with an arrow mark at the beginning of the line for identification.

Use the scroll bar or arrow keys to scroll the window. Or enter a command to display the program code beginning with a specified position.

* Display of source line numbers and source statements

The source line numbers and source statements can only be displayed when the IEEE-695 absolute object file including debugging information for the source display is loaded. Furthermore, the source statements that are actually displayed from this file are those which have had the -g option specified by the assembler.

3 S1C63 FAMILY SIMULATOR

* Updating of display

When a program is loaded and executed (g, gr, s, n, or rst command), or the memory contents are changed (a (as), pe, pf, or pm command), the display contents are updated. In this case the [Source] window updates its display contents so that the current PC address can always be displayed. The display contents are also updated when the display mode is changed.

(2) Current PC

The current PC (program counter) address line is indicated by an arrow mark at the beginning of the line. (Address 0x0110 in the diagram)

(3) PC breakpoint

The address line where a breakpoint is set is indicated by a red ● mark at the beginning of the line. (Address 0x0115 in the diagram)

(4) Break setting at the cursor position

Place the cursor at an address line where a breakpoint is to be set (not available for a source-only line).



[Break] button

Then click on the [Break] button. A PC breakpoint will be set at that address.

If the same is done at the address line where a PC breakpoint has been set, the breakpoint will be cleared.

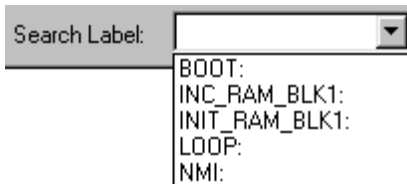


[Go to Cursor] button

If the [Go to Cursor] button is clicked, the program will execute beginning with the current PC position, and program execution breaks at the line where the cursor is located.

(5) Finding labels and words

Any labels and words can be found using the [Search Label] pull-down list box or the [Find] button on the [Source] window.



[Search Label] pull-down list box



[Find] button

3.6.6 [Data] Window

ADDR:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
FF00	0	0	*	*	0	0	0	0	*	*	*	*	*	*	*	*
FF10	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FF20	0	0	F	*	0	0	F	*	*	*	*	*	*	*	*	*
FF30	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*
FF40	0	F	0	*	0	F	0	*	*	*	*	*	*	*	*	*
FF50	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FF60	0	4	0	*	*	*	*	*	*	*	*	*	0	0	0	0
FF70	0	0	0	0	0	0	0	*	0	0	0	0	0	*	*	*
FF80	A	0	0	A	0	0	0	*	*	*	*	*	*	*	*	*
FF90	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*
FFA0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FFB0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FFC0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FFD0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FFE0	*	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*
FFF0	*	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*

(1) Displaying data memory contents

The [Data] window displays the memory dump results in hexadecimal numbers.

The display area is the entire 64K-word data memory space (RAM, data ROM, I/O). The contents of all addresses from 0x0000 to 0xffff can be displayed by scrolling the window. The contents of unmapped addresses in each microcomputer model are indicated by an "*".

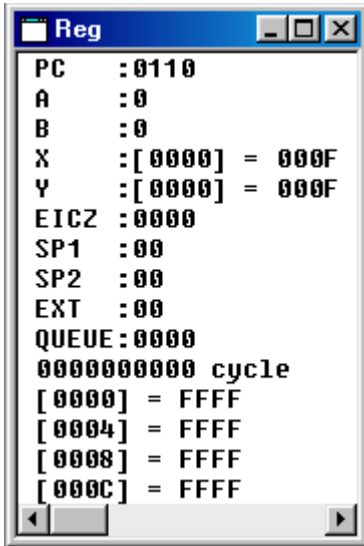
* Updating of display

The display contents of the [Data] window are updated automatically when memory contents are modified with a command (de, df, or dm command), or by direct modification. After executing the program (g, gr, s, n, or rst command), the display contents are also updated. To refresh the [Data] window manually, execute the dd command or click the vertical scroll bar.

(2) Direct modification of data memory contents

The [Data] window allows direct modification of data memory contents. To modify data on the [Data] window, place the cursor at the front of the data to be modified or double click the data, and then type a hexadecimal character (0-9, a-f). Data in the address will be modified with the entered number and the cursor will move to the next address. This allows successive modification of a series of addresses.

3.6.7 [Register] Window



(1) Displaying register contents

The [Register] window displays the contents of the PC, A register, B register, X register and its memory, Y register and its memory and flags (E, I, C, Z), stack pointers (SP1, SP2), EXT register, and QUEUE register.

(2) Execution cycle counter

This counter calculates and indicates the number of executed cycles or execution time since the CPU was reset.

(3) Monitor data

The debugger allows you to specify four addresses in RAM and monitor the memory contents at these addresses. The [Register] window displays the contents of these four watch data addresses (4 words each beginning from the specified address). When the debugger starts up, addresses 0, 4, 8, and C are initially set as the watch data addresses. The contents are arranged sequentially from left to right in order of their addresses as they are displayed on the screen.

*** Updating of display**

The display is updated when registers are dumped (rd command), when watch data addresses are set (dw command), when register data is modified (rs command), when the CPU is reset (rst command), or after program execution (g, gr, s, or n command) is completed.

(4) Direct modification of register contents

The [Register] window allows direct modification of register contents. To modify data on the [Register] window, select (highlight) the data to be modified and type a hexadecimal number (0-9, a-f), then press [Enter]. The register data will be modified with the entered number.

3.6.8 [Trace] Window

trace cycle	fetch addr	fetch code	disasm	register	flag	data	trace
				A B X Y	EICZ	addr data SP	in
00015	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00014	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21 r0	
00013	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00012	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00011	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21 r0	
00010	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00009	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00008	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21 r0	
00007	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00006	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00005	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21 r0	
00004	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00003	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00002	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21 r0	
00001	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--

After the trace function is turned on by the tm command, the simulator samples trace information while the target program is running. The trace data buffer has a capacity for 8192 instructions (overwritten from the beginning if the capacity is exceeded), and its data can be displayed in the [Trace] window.

The following lists the trace contents:

- Traced cycle number
- Fetched address
- Fetched code and disassembled contents
- Register contents (A, B, X, Y, and flags)
- Memory access status (address, R/W, data, and SP1/SP2)

This window also displays the trace data search results by the ts command.

* Updating of display

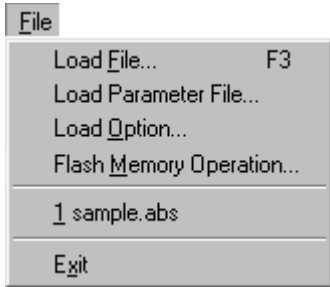
The contents of the [Trace] window are cleared when the target program is being executed. During this period, the [Trace] window does not accept scrolling and resizing operations.

After an program execution is terminated, this window displays the latest data traced during the execution. To specify a display start cycle, execute the td command.

3.7 Menu

This section outlines the menu bar available with the simulator. The menu bar has eight menus, each including frequently-used commands.

[File] Menu



Note: [Flash Memory Operation...] is invalid in Sim63.

[Load File...]

This menu item reads an object file in the IEEE-695 format into the simulator. It performs the same function when the lf command is executed.

[Load Parameter File...]

This menu item reads a parameter file into the simulator. It performs the same function when the par command is executed.

[Load Option...]

This menu item reads a program file, data file for the data ROM or an optional HEX file in Motorola S2 format into the simulator. It performs the same function when the lo command is executed.

[Exit]

This menu item quits the simulator. It performs the same function when the q command is executed.

[Run] Menu



[Go]

This menu item executes the target program from the address indicated by the current PC. It performs the same function when the g command is executed.

[Go to Cursor]

This menu item executes the target program from the address indicated by the current PC to the cursor position in the [Source] window (the address of that line). It performs the same function when the g <address> command is executed. Before this menu item can be selected, the [Source] window must be open and the address line where the program is to break must be clicked. Selecting a break address by clicking on the address line is valid for only the lines that have actual code, and is invalid for the source-only lines.

[Go from Reset]

This menu item resets the CPU and then executes the target program from the program start address (0x0110). It performs the same function when the gr command is executed.

[Step]

This menu item executes one instruction step at the address indicated by the current PC. It performs the same function when the s command is executed.

[Next]

This menu item executes one instruction step at the address indicated by the current PC. If the instruction to be executed is calr, calz or int, it is assumed that a program section until control returns to the next address constitutes one step and all steps of their subroutines are executed. This menu item performs the same function when the n command is executed.

[Command File...]

This menu item reads a command file and executes the debug commands written in that file. It performs the same function when the com or cmw command is executed.

[Reset CPU]

This menu item resets the CPU. It performs the same function when the rst command is executed.

[Break] Menu**[Breakpoint Set...]**

This menu item displays, sets or clears PC breakpoints using a dialog box. It performs the same function as executing the bp command.

[Data Break...]

This menu item displays, sets or clears data break conditions using a dialog box. It performs the same function as executing the bd command.

[Register Break...]

This menu item displays, sets or clears register break conditions using a dialog box. It performs the same function as executing the br command.

[Sequential Break...]

This menu item displays, sets or clears sequential break conditions using a dialog box. It performs the same function as executing the bs command.

[Stack Break...]

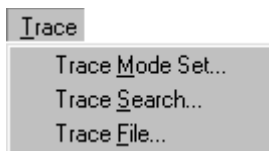
This menu item displays or sets stack break conditions using a dialog box. It performs the same function as executing the bsp command.

[Break List]

This menu item displays the all break conditions that have been set. It performs the same function as executing the bl command.

[Break All Clear]

This menu item clears all break conditions. It performs the same function as executing the bac command.

[Trace] Menu**[Trace Mode Set...]**

This menu item sets a trace mode ON and OFF. It performs the same function as executing the tm command.

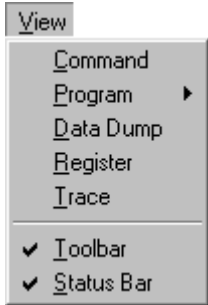
[Trace Search...]

This menu item searches trace information from the trace memory under the condition specified using a dialog box. It performs the same function as executing the ts command.

[Trace File...]

This menu item saves the specified range of the trace information displayed in the [Trace] window to a file. It performs the same function as executing the tf command.

[View] Menu



[Command]

This menu item activates the [Command] window.

[Program]



This menu item opens or activates the [Source] window and displays the program from the current PC address in the display mode selected from the sub menu items. These sub menu items perform the same functions as executing the u, sc, and m command, respectively.

[Data Dump]

This menu item opens or activates the [Data] window and displays the data memory contents from the memory start address.

[Register]

This menu item opens or activates the [Register] window and displays the current values of the registers.

[Trace]

This menu item opens or activates the [Trace] window and displays the trace data sampled in the trace memory.

[Toolbar]

This menu item shows or hides the toolbar.

[Status Bar]

This menu item shows or hides the status bar.

[Option] Menu



[Log...]

This menu item starts or stops logging using a dialog box. It performs the same function as executing the log command.

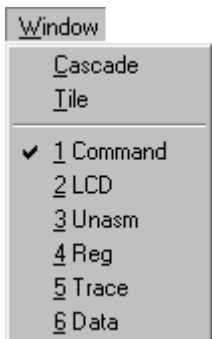
[Record...]

This menu item starts or stops recording of a command execution using a dialog box. It performs the same function as executing the rec command.

[Mode Setting...]

This menu item sets the simulator operating modes using a dialog box. It performs the same functions as executing the md command.

[Window] Menu



[Cascade]

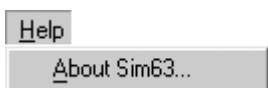
This menu item cascades the opened windows.

[Tile]

This menu item tiles the opened windows.

This menu shows the currently opened window names. Selecting one activates the window.

[Help] Menu



[About Sim63...]

This menu item displays an About dialog box for the simulator.

3.8 Tool Bar

The tool bar has 15 buttons, each one assigned to a frequently used command.



[Key Break] button

This button forcibly breaks execution of the target program. This function can be used to cause the program to break when the program has fallen into an endless loop.



[Load File] button

This button reads an object file in the IEEE-695 format into the simulator. It performs the same function when the lf command is executed.



[Load Parameter] button

This button reads a parameter file into the simulator. It performs the same function when the par command is executed.



[Load Option] button

This button reads a program file, data file for the data ROM or an optional HEX file in Motorola S2 format into the simulator. It performs the same function when the lo command is executed.



[Source] button

This button switches the display of the [Source] window to the source mode. The [Source] window opens if it is closed. This button performs the same function when the sc command is executed.



[Unassemble] button

This button switches the display of the [Source] window to the unassemble mode. The [Source] window opens if it is closed. This button performs the same function when the u command is executed.



[Mix] button

This button switches the display of the [Source] window to the mix mode (unassemble & source). The [Source] window opens if it is closed. This button performs the same function when the m command is executed.



[Go] button

This button executes the target program from the address indicated by the current PC. It performs the same function when the g command is executed.



[Go to Cursor] button

This button executes the target program from the address indicated by the current PC to the cursor position in the [Source] window (the address of that line). It performs the same function when the g <address> command is executed. Before this button can be selected, the [Source] window must be open and the address line where the program is to break must be clicked. Selecting a break address by clicking on the address line is valid for only the lines that have actual code, and is invalid for the source-only lines.



[Go from Reset] button

This button resets the CPU and then executes the target program from the program start address (0x110). It performs the same function when the gr command is executed.



[Step] button

This button executes one instruction step at the address indicated by the current PC. It performs the same function when the s command is executed.



[Next] button

This button executes one instruction step at the address indicated by the current PC. If the instruction to be executed is calr, calz or int, it is assumed that a program section until control returns to the next address constitutes one step and all steps of their subroutines are executed. This button performs the same function when the n command is executed.



[Reset] button

This button resets the CPU. It performs the same function when the rst command is executed.



[Break] button

Use this button to set and clear a breakpoint at the address where the cursor is located in the [Source] window. This function is valid only when the [Source] window is open. Note that selecting a break address by clicking on the address line is valid for only the lines that have actual code and is invalid for the source-only lines.



[Help] button

By clicking on this button, a help window appears on the screen, displaying the contents of help topics.

3.9 Method for Executing Commands

All debug functions can be performed by executing debug commands. This section describes how to execute these commands.

3.9.1 Entering Commands from Keyboard

Select the [Command] window (by clicking somewhere on the [Command] window). When the prompt ">" appears on the last line in this window and a cursor is blinking behind it, the system is ready to accept a command from the keyboard. Input a debug command at the prompt position. The commands are not case-sensitive; they can be input in either uppercase or lowercase.

General command input format

>command [parameter [parameter ... parameter]] ␣

- A space is required between a command and parameter.
- Space is required between parameters.

Use the arrow keys, [Back Space] key, or [Delete] key to correct erroneous input.

When you press the [Enter] key after entering a command, the system executes that command. (If the command entered is accompanied by guidance, the command is executed when the necessary data is input according to the displayed guidance.)

Input example:

>g␣ (Only a command is input.)
 >com test.cmd␣ (A command and parameter are input.)

Command input accompanied by guidance

For commands that cannot be executed unless a parameter or the commands that modify the existing data are specified, a guidance mode is entered when only a command is input. In this mode, the system brings up a guidance field, so input a parameter there.

Input example:

>lf␣
 File name ? :test.abs␣ ← Input data according to the guidance (underlined part).
 >

• Commands requiring parameter input as a precondition

The If command shown in the above example reads a program file into the simulator. Commands like this that require an entered parameter as a precondition are not executed until the parameter is input and the [Enter] key pressed. If a command has multiple parameters to be input, the system brings up the next guidance, so be sure to input all necessary parameters sequentially. If the [Enter] key is pressed without entering a parameter in some guidance session of a command, the system assumes the command is canceled and does not execute it.

• Commands that replace existing data after confirmation

The commands that rewrite memory or register contents one by one provide the option of skipping guidance (do not modify the contents), returning to the immediately preceding guidance, or terminating during the input session.

[Enter] key Skips input.
 [^] key Returns to the immediately preceding guidance.
 [q] key Terminates the input session.

Input example:

>de␣ ← Command to modify data memory.
 Data enter address ? :0␣ ← Inputs the start address.
 0000 A:1␣ ← Modifies address 0x0000 to 1.
 0001 A:^␣ ← Returns to the immediately preceding address.
 0000 1:0␣ ← Inputs address 0x0000 back again.
 0001 A:␣␣ ← Skips address 0x0001 by pressing [Enter] alone.
 0002 A:␣␣
 0001 A:q␣ ← Terminates the input session.
 >

Numeric data format of parameter

For numeric values to be accepted as a parameter, they must be input in hexadecimal numbers for almost all commands. However, some parameters accept decimal or binary numbers.

The following characters are valid for specifying numeric data:

Hexadecimal: 0–9, a–f, A–F, *

Decimal: 0–9

Binary: 0, 1, *

("*" is used to mask bits when specifying a data pattern.)

Specification with a symbol

For address specifications, symbols defined in the source can also be used. However, it is necessary to load an absolute object file that contains debug information.

Symbols should be used as follows:

Global symbol	@<symbol name>	e.g. @RAM_BLK1
Local symbol	@<symbol name>@<source file name>	e.g. @LOOP@main.s

Successive execution using the [Enter] key

The commands listed below can be executed successively by using only the [Enter] key after executing once. Successive execution here means repeating the previous operation or continuous display of the previous contents.

Execution commands: g (go), s (step), n (next), com (execute command file)














Display commands: sc (source), m (mix), u (unassemble), dd (data memory dump),
td (trace data display), sy (symbol list), ma (map information)

The successive execution function is terminated when some other command is executed.

3.9.2 Executing from Menu or Tool Bar

The menu and tool bar are assigned frequently-used commands as described in Sections 3.7 and 3.8. A command can be executed simply by selecting desired menu command or clicking on the tool bar button. Table 3.9.2.1 lists the commands assigned to the menu and tool bar.

Table 3.9.2.1 Commands that can be specified from menu or tool bar

Command	Function	Menu	Button
lf	Load IEEE-695 absolute object file	[File Load File...]	
par	Load parameter file	[File Load Parameter File...]	
lo	Load Motorola S2 file	[File Load Option...]	
g	Execute program successively	[Run Go]	
g <address>	Execute program to <address> successively	[Run Go to Cursor]	
gr	Reset CPU and execute program successively	[Run Go after Reset]	
s	Single step execution	[Run Step]	
n	Step execution with skip subroutine	[Run Next]	
com	Load and execute command file	[Run Command File...]	–
cmw	Load and execute command file with wait	[Run Command File...]	–
rst	Reset CPU	[Run Reset CPU]	
bp, bc (bpc)	Set/clear PC breakpoint	[Break Breakpoint Set...]	
bd, bdc	Set/clear data break	[Break Data Break...]	–
br, brc	Set/clear register break	[Break Register Break...]	–
bs, bsc	Set/clear sequential break	[Break Sequential Break...]	–
bsp	Set stack break	[Break Stack Break...]	–
bl	Break list	[Break Break List]	–
bac	Clear all break conditions	[Break Break All Clear]	–
tm	Set trace mode	[Trace Trace Mode Set...]	–
ts	Search trace information	[Trace Trace Search...]	–
tf	Save trace information to file	[Trace Trace File...]	–
u	Unassemble display	[View Program Unassemble]	
sc	Source display	[View Program Source Display]	
m	Mix display	[View Program Mix Mode]	
dd	Dump data memory	[View Data Dump]	–
rd	Display register values	[View Register]	–
td	Display trace information	[View Trace]	–
log	Turn log output on or off	[Option Log...]	–
rec	Record commands to a command file	[Option Record...]	–
md	Set modes	[Option Mode Setting...]	–

3.9.3 Executing from a Command File

Another method for executing commands is to use a command file that contains descriptions of a series of debug commands. By reading a command file into the simulator the commands written in it can be executed.

Creating a command file

Create a command file as a text file using an editor.

Although there are no specific restrictions on the extension of a file name, Seiko Epson recommends using ".cmd".

Command files can also be created using the rec command. The rec command creates a command file and saves the executed commands to the file.

Example of a command file

The example below shows a command group that loads a program file, sets a breakpoint and then executes the program.

Example: File name = start.cmd

```
lo test.fsa
lo test.ssa
lf test.abs
bp 0004d7
g
```

A command file to write the commands that come with a guidance mode can be executed. In this case, be sure to break the line for each guidance input item as a command is written.

Reading in and executing a command file

The simulator has the com and cmw commands available that can be used to execute a command file. The com command reads in a specified file and executes the commands in that file sequentially in the order they are written.

The cmw command performs the same function as the com command except that each command is executed at intervals specified by the md command (1 to 256 seconds).

Example: com start.cmd

```
cmw test.cmd
```

The commands written in the command file are displayed in the [Command] window.

Restrictions

Another command file can be read from within a command file. However, nesting of these command files is limited to a maximum of five levels. An error is assumed and the subsequent execution is halted when the com or cmw command at the sixth level is encountered.

3.9.4 Log File

The executed commands and the execution results can be saved to a file in text format that is called a "log file". This file allows verification of the debug procedures and contents. The contents displayed in the [Command] window are saved to this file.

Command example

```
>log tst.log
```

After the simulator is set to the log mode by the log command (after it starts outputting to a log file), the log command toggles (output turned on in log mode \cap output turned off in normal mode). Therefore, you can output only the portions needed can be output to the log file.

Display of [Command] window in log mode

The contents displayed in the [Command] window during log mode differ from those appearing in normal mode.

- (1) When executing a command when each window is open
(When the window that displays the command execution result is opened)
Normal mode: The contents of the relevant display window are updated. The execution results are not displayed in the [Command] window.
Log mode: The same contents as those displayed in the relevant window are also displayed in the [Command] window. However, changes made to the relevant window by scrolling or opening it are not reflected in the [Command] window.
- (2) When executing a command while each window is closed
When the relevant display window is closed, the execution results are always displayed in the [Command] window regardless of whether operation is in log mode or normal mode.





3.10 Debug Functions

This section outlines the debug features of the simulator, classified by function.

3.10.1 Loading Files

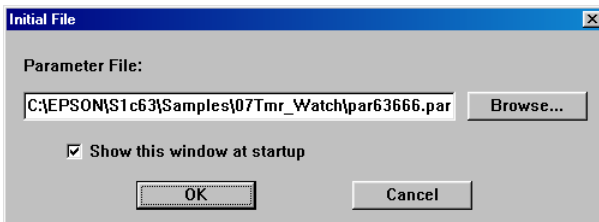
Table 3.10.1.1 lists the files read by the simulator and the load commands.

Table 3.10.1.1 Files and load commands

File	Type	Generation tool	Command	Menu	Button
1. Parameter file	.par	–	par	[File Load Parameter File...]	
2. LCD panel definition file	.lcd	LcdUtil	–	–	–
3. Component mapping file	.cmp	–	–	–	–
4. Port setting file	.prt	–	–	–	–
5. Simulator project file	.spj	–	par	[File Load Parameter File...]	
6. IEEE-695 absolute object file	.abs	lk63	lf	[File Load File...]	
7. Program HEX file	.hsa .lsa	hx63	lo	[File Load Option...]	
8. Data ROM HEX file	.csa				
9. Function option file	.fsa	fog63xxx or winfog			
10. Segment option file	.ssa	sog63xxx or winsog			
11. Command file	.cmd	–	com/cmw	[Run Command File...]	–

Files 1 to 4 are required for starting the simulator.

Either a parameter file (1) or a simulator project file (5) must be selected in the dialog box shown below at the first time the simulator starts up.

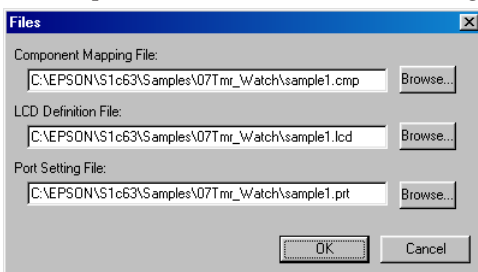


Enter the parameter file name or simulator project file name to the text box, or choose it using the [Browse...] button.

If the [Show this window at startup] check box is deselected, this dialog box will not appear from the next startup of the simulator and the same file will be selected.

Loading a parameter file resets the simulator. The memory mapping information set by the parameter file can be displayed using the ma command.

When a parameter file is selected, the dialog box shown below appears to select files from 2 to 4.



When a simulator project file is selected, this dialog box does not appear because the file names (2 to 4) are specified in the file.

Enter a component mapping file name, LCD panel definition file name and a port setting file name to the respective text boxes, or chose using the [Browse...] button.

These files are selected once, the file names will appear in the text boxes at the next startup of the simulator allowing choice of the files by clicking the [OK] button only.

The lf command loads an absolute object file (.abs) in the IEEE-695 format. To perform source display and symbolic debugging, load an object file that contains the debugging information in this format. Furthermore, the source files must be located in the original directory.

The lo command loads a program/data files (.hsa, .lsa, .csa) or an option HEX file (.fsa, .ssa) in the Motorola S2 format. When files in this format are loaded, the [Source] window can display the program code only in unassembled mode.

For the command file, refer to Section 3.9.3.




3.10.2 Source Display and Symbolic Debugging Function

The simulator allows program debugging while displaying the assembly source statements. Address specification using a symbol name is also possible.

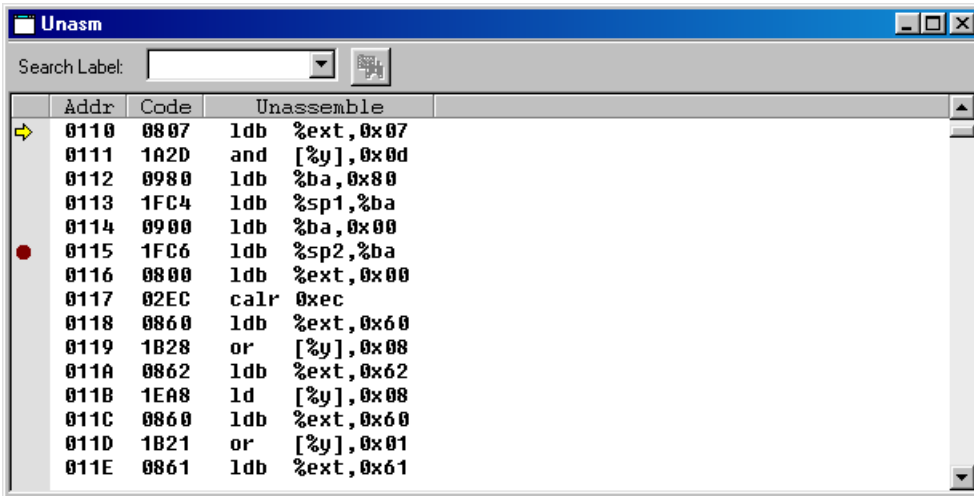
Displaying program code

The [Source] window displays the program in the specified display mode. The display mode can be selected from among the three modes: Unassemble mode, Source mode, Mix mode.

Table 3.10.2.1 Commands/menu items/tool bar buttons to switch display mode

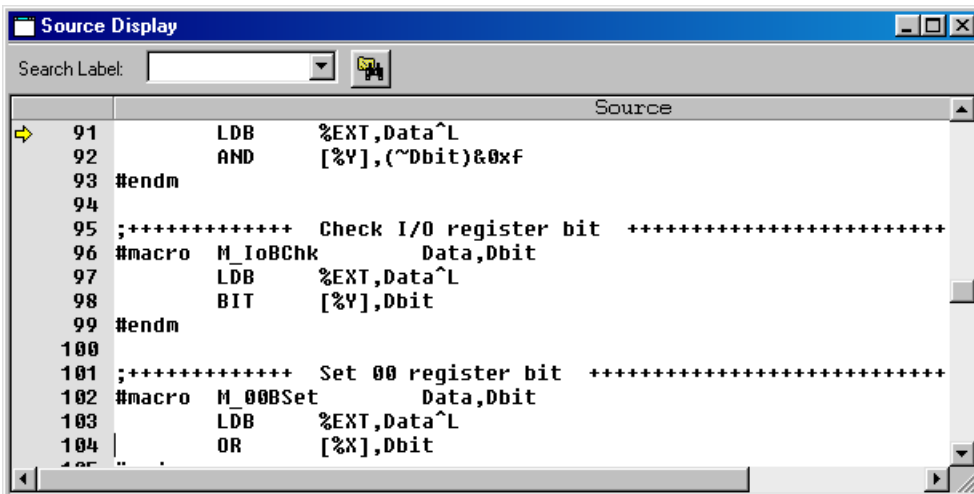
Function	Command	Menu	Button
Unassemble display mode	u	[View Program Unassemble]	
Source display mode	sc	[View Program Source Display]	
Mix display mode	m	[View Program Mix Mode]	

(1) Unassemble mode



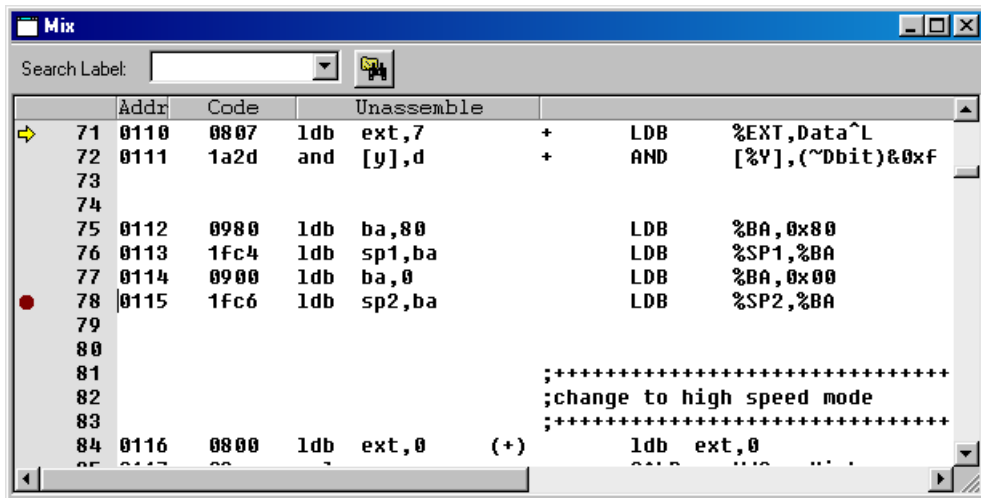
In this mode, the simulator displays the program codes after unassembling into mnemonics.

(2) Source mode



In this mode, the source that contains the code at the current PC address is displayed like an editor screen. This mode is available only when an absolute object file that contains source debugging information has been loaded.

(3) Mix mode



In this mode, both unassembled codes and sources are displayed like an absolute list. This mode is available only when an absolute object file that contains source debugging information has been loaded.

Refer to Section 3.6.5, "[Source] Window" for details about the display contents.

Symbol reference

When debugging a program after reading an object file in IEEE-695 format, the symbols defined in the source file can be used to specify an address. This feature can be used when entering a command having <address> in its parameter from the [Command] window or a dialog box.

(1) Referencing global symbols

Follow the method below to specify a symbol that is declared to be a global symbol/label by the .global or .comm pseudo-instruction.

@<symbol>

Example of specification:

```
>m @BOOT
```

```
>de @RAM_BLK1
```

(2) Referencing local symbols

Follow the method below to specify a local symbol/label that is used in only the defined source file.

@<symbol>@<file name>

The file name here is the source file name (.s) in which the symbol is defined.

Example of specification:

```
>bp @SUB1@test.s
```

(3) Displaying symbol list

All symbols used in the program and the defined addresses can be displayed in the [Command] window.

Table 3.10.2.2 Command to display symbol list

Function	Command	Menu	Button
Displaying symbol list	sy	—	—

3.10.3 Displaying and Modifying Program, Data, Option Data and Register

The simulator has functions to operate on the program memory, data memory, and registers, as well as option data. Each memory area is set to the simulator according to the map information that is given in a parameter file.

Operating on program memory area

The following operations can be performed on the program memory area:

Table 3.10.3.1 Commands to operate on program memory

Function	Command	Menu	Button
Entering/modifying program code	pe	–	–
In-line assemble	a (as)	–	–
Rewriting specified area	pf	–	–
Copying specified area	pm	–	–

(1) Entering/modifying program code

The program code at a specified address is modified by entering hexadecimal data.

(2) In-line assemble

The program code at a specified address is modified by entering a mnemonic code.

(3) Rewriting specified area

An entire specified area is rewritten with specified code.

(4) Copying specified area

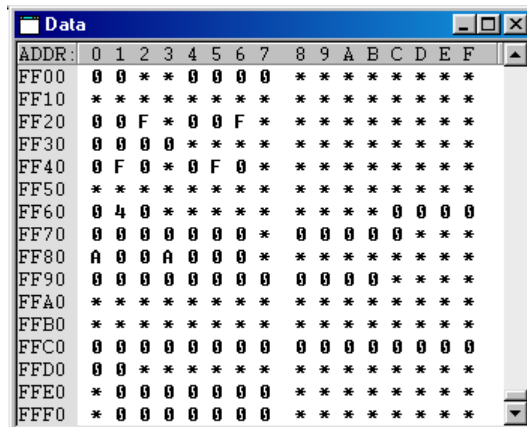
The content of a specified area is copied to another area.

Operating on data memory area

The following operations can be performed on the data memory areas (RAM, data ROM, display memory, I/O memory):

Table 3.10.3.2 Commands/menu item to operate on data memory

Function	Command	Menu	Button
Dumping memory data	dd	[View Data Dump]	–
Entering/modifying memory data	de	–	–
Rewriting specified area	df	–	–
Coping specified area	dm	–	–



(1) Dumping data memory

The contents of the data memory are displayed in hexadecimal dump format. If the [Data] window is opened, the contents of the [Data] window are updated; if not, the contents of the data memory are displayed in the [Command] window.

(2) Entering/modifying data

Data at a specified address is rewritten by entering hexadecimal data. Data can be directly modified on the [Data] window.

(3) Rewriting specified area

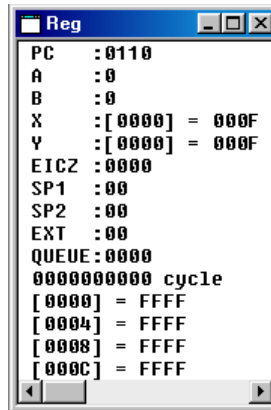
An entire specified area is rewritten with specified data.

(4) Copying specified area

The content of a specified area is copied to another area.

(5) Monitoring memory

Four memory locations, each with area to store 4 consecutive words, can be registered as watch data addresses. The registered watch data can be verified in the [Register] window. Addresses 0, 4, 8, and C are made the watch data addresses by default.



The memory content displayed at the left indicates data at a specified address, and the one displayed at the right indicates 4-word data at the high-order address.

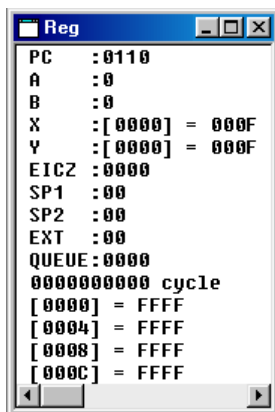
Monitor data

Operating registers

The following operations can be performed on registers:

Table 3.10.3.3 Commands/menu items to operate registers

Function	Command	Menu	Button
Displaying register values	rd	[View Register]	-
Modifying register value	rs	-	-



(1) Displaying registers

Register contents can be displayed in the [Register] or [Command] window.

Registers: PC, A, B, X and [X], Y and [Y], F, SP1, SP2, EXT, and QUEUE

(2) Modifying register values

The contents of the above registers can be set to any desired value. The register values can be directly modified on the [Register] window.

3.10.4 Executing Program

The simulator can execute the target program successively or execute instructions one step at a time (single-stepping).




Successive execution

(1) Types of successive execution

There are two types of successive execution available:

- Successive execution from the current PC
- Successive execution from the program start address (0x0110) after resetting the CPU

Table 3.10.4.1 Commands/menu items/tool bar buttons for successive execution

Function	Command	Menu	Button
Successive execution from current PC	g	[Run Go]	
		[Run Go to Cursor]	
Successive execution after resetting CPU	gr	[Run Go from Reset]	

(2) Stopping successive execution

Using the successive execution command (g), can specify up to two temporary break addresses that are only effective during program execution.

The temporary break address can also be specified from the [Source] window (one location only).

If the cursor is placed on an address line in the [Source] window and the [Go to Cursor] button clicked, the program starts executing from the current PC address and breaks before executing the instruction at the address the cursor is placed.

Except being stopped by this temporary break, the program continues execution until it is stopped by one of the following causes:

- Break conditions set by a break set up command are met.
- The [Key Break] button is clicked.
- A map break, etc. occurs.



[Key Break] button

* When the program does not stop, use this button to forcibly stop it.

(3) Simulation for LCD panel display and external input/output

The [LCD] window shows the LCD panel images according to the program sequence during the program execution. It also allows simulation of key inputs using the keyboard of the computer. These functions are configured with the component mapping file, LCD panel definition file and port setting file loaded at the startup of the simulator. See Section 3.6.3, "[LCD] Window", for more information. Furthermore, serial/general port inputs/outputs and A/D conversion can be simulated by reading an IOT file from the [I/O Terminal] window. See Section 3.6.4, "[I/O Terminal] Window", for details.

Single-stepping



(1) Types of single-stepping

There are two types of single-stepping available:

- Stepping through all instructions (STEP)
All instructions are executed one step at a time according to the PC, regardless of the type of instruction.
- Stepping through instructions except subroutines (NEXT)
The calr, calz and int instructions are executed under the assumption that one step constitutes the range of statements until control is returned to the next step by a return instruction. Other instructions are executed in the same way as in ordinary single-stepping.

In either case, the program starts executing from the current PC.

Table 3.10.4.2 Commands/menu items/tool bar buttons for single-stepping

Function	Command	Menu	Button
Single step execution	s	[Run Step]	
Single step execution except subroutines	n	[Run Next]	

When executing single-stepping by command input, the number of steps to be executed can be specified, up to 65,535 steps. When using menu commands or tool bar buttons, the program is executed one step at a time.

In the following cases, single-stepping is terminated before a specified number of steps is executed:

- When the [Key Break] button is clicked.
- When a map break or similar break occurs.

Single-stepping is not suspended by breaks set by the user such as a PC break or data break.



[Key Break] button * When the program does not stop, use this button to forcibly stop it.

(2) Display during single-stepping

In the initial simulator settings, the display is updated as follows:

The display contents of the [Register] window are updated every step. If the [Register] window is closed, its contents are displayed in the [Command] window. This default display mode can be switched over by the md command so that the display contents are updated at only the last step in a specified number of steps. The display of the [Source] and [Data] windows are updated after the specified number of step executions are completed.

(3) HALT and SLEEP states and interrupts

The CPU is placed in a standby mode when the halt or slp instruction is executed. An interrupt is required to cancel this mode. The simulator has a mode to enable or disable an external interrupt for use in single-step operation.

Table 3.10.4.3 External interrupt modes

	Enable mode	Disable mode
External interrupt	Interrupt is processed.	Interrupt is not processed.
halt and slp instructions	Executed as the halt instruction. Processing is continued by an external interrupt or clicking on the [Key Break] button.	The halt and slp instructions are replaced with a nop instruction as the instruction is executed.

In the initial settings, the simulator is set to the interrupt disable mode. The interrupt enable mode can also be set by using the md command.

(4) Key entry simulation during single-stepping

Key entry status can be set on the [Key List] window displayed by clicking on the [Key List] button on the [LCD] window and is maintained during single-stepping.

Measuring execution cycles

(1) Execution cycle counter

The simulator contains a 31-bit execution cycle counter allowing you to measure the number of bus cycles executed.

Up to 2,147,483,647 cycles can be measured by the execution cycle counter.

(2) Displaying measurement results

The measurement result is displayed in the [Register] window. This display is cleared during program execution and is updated after completion of execution. If the [Register] window is closed, the measurement result can be displayed in the [Command] window using the rd command. The execution results of single-stepping are also displayed here.

If the counter's maximum count is exceeded, the system indicates "over flow".

(3) Hold mode and reset mode

In the initial simulator settings, the execution cycle counter is set to hold mode. In this mode, the measured values are combined until the counter is reset.

The reset mode can be set by the md command. In this mode, the counter is reset each time the program is executed. In successive execution, the counter is reset when the program is made to start executing by entering the g command and measurement is taken until the execution is terminated (beak occurs). (The same applies for the gr command except that the counter is reset simultaneously when the CPU is reset. Consequently, the counter operates the same way in both hold and reset modes.)

In single-stepping, the counter is reset when the program is made to start executing by entering the s or n command and measurement is taken until execution of a specified number of steps is completed. The counter is reset every step if execution of only one step is specified or execution is initiated by a tool bar button or menu command.


(4) Resetting execution cycle counter

The execution cycle counter is reset in the following cases:

- When the CPU is reset with the rst command, [Reset CPU] in the [Run] menu, or the [Reset] button
- When the gr command or [Go from Reset] in the [Run] menu is executed
- When the execution cycle counter mode is switched over by the md command (between hold and reset modes)
- When program execution is started in reset mode

Resetting the CPU

Table 3.10.4.4 Commands/menu items/tool bar buttons for resetting CPU

Function	Command	Menu	Button
Reset CPU	rst	[Run Reset CPU]	
Successive execution after resetting CPU	gr	[Run Go from Reset]	

The CPU is reset when the gr command is executed, or by executing the rst command.

When the CPU is reset, the internal circuits are initialized as follows:

(1) Internal registers of the CPU

PC	... 0x0110
A, B	... 0xa
X, Y, QUEUE	... 0xaaaa
F	... 0b0000
SP1, SP2, EXT	... 0xaa

(2) The execution cycle counter is reset to 0.

(3) The [Source] and [Register] windows are redisplayed.

Because the PC is set to 0x0110, the [Source] window is redisplayed beginning with that address.

The [Register] window is redisplayed with the internal circuits initialized as described above.

The data memory contents are not modified.

3.10.5 Break Functions

The target program is made to stop executing by one of the following causes:

- Break command conditions are satisfied.
- The [Key Break] button is activated.
- A map break or similar break occurs.

Break by command

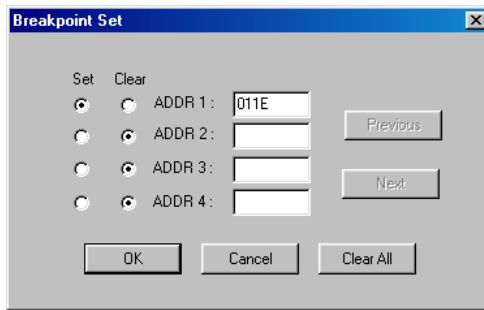
The simulator has five types of break functions that allow the break conditions to be set by a command. When the set conditions in one of these break functions are met, the program under execution is made to break.

(1) Break by PC

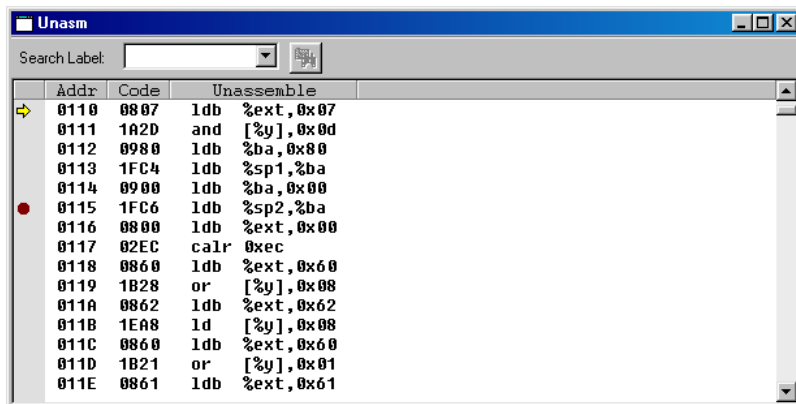
This function causes the program to break when the PC matches the set address. The program is made to break before executing the instruction at that address. The PC breakpoints can be set for multiple addresses.

Table 3.10.5.1 Commands/menu items/tool bar button to set breakpoints

Function	Command	Menu	Button
Set breakpoints	bp	[Break Breakpoint Set...]	
Clear break points	bc (bpc)	[Break Breakpoint Set...]	



The addresses that are set as PC breakpoints are marked with a ● as they are displayed in the [Source] window.



Using the [Break] button easily allows the setting and canceling of breakpoints.

Click on the address line in the [Source] window at where the program break is desired (after moving the cursor to that position) and then click on the [Break] button. A ● mark will be placed at the beginning of the line indicating that a breakpoint has been set there, and the address is registered in the breakpoint list. Clicking on the line that begins with a ● and then the [Break] button cancels the breakpoint you have set, in which case the address is deleted from the breakpoint list.

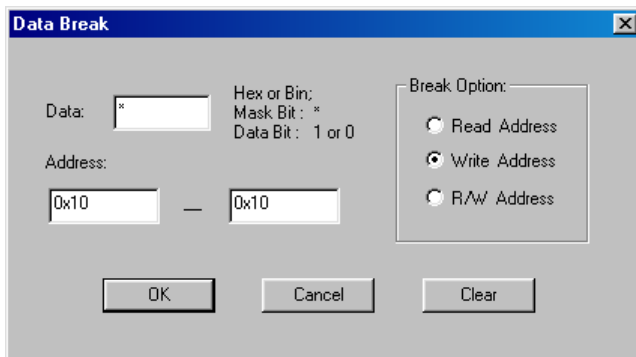
* The temporary break addresses that can be specified by the successive execution commands (g) do not affect the set addresses in the breakpoint list.

(2) Data break

This break function allows a break to be executed when a location in the specified data memory area is accessed. In addition to specifying a memory area in which to watch accesses, specification as to whether the break is to be caused by a read or write, as well as specification of the content of the data read or written. The read / write condition can be masked, so that a break will be generated for whichever operation, read or write, is attempted. Similarly, the data condition can also be masked in bit units. A break occurs after completing the cycle in which an operation to satisfy the above specified condition is performed.

Table 3.10.5.2 Commands/menu item to set data break

Function	Command	Menu	Button
Set data break condition	bd	[Break Data Break...]	–
Clear data break condition	bdc	[Break Data Break...]	–



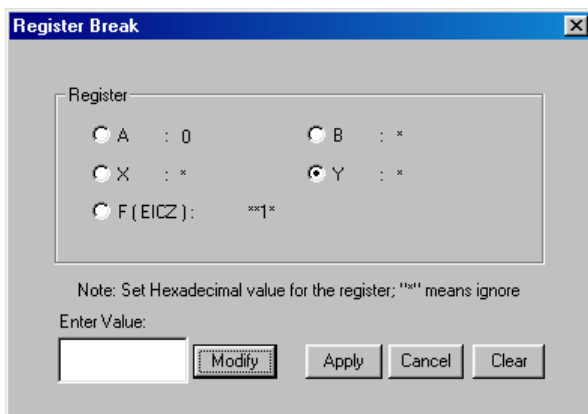
For example, if the program is executed after setting the data break condition as Address = 0x10, Data pattern = * (mask) and R/W = W, the program breaks after writing any data to the data memory address 0x10.

(3) Register break

This break function causes a break when the A, B, F, X, and Y register reach a specified value. Each register can be masked (so they are not included in break conditions). The F register can be masked in bit units. A break occurs when the above registers are modified to satisfy all set conditions.

Table 3.10.5.3 Commands/menu item to set register break

Function	Command	Menu	Button
Set register break condition	br	[Break Register Break...]	–
Clear register break condition	brc	[Break Register Break...]	–



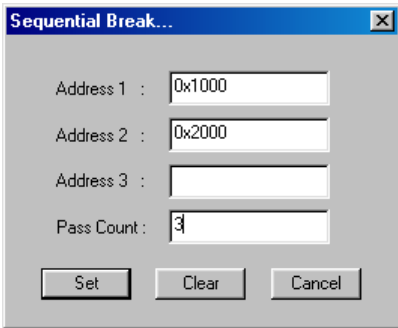
For example, if the program is executed after setting 0 for the data of the A register and "**1*" for the data of the F register (C flag = 1) and masking all others, the program breaks when the A register is cleared to 0 and the C flag is set to 1.

(4) Sequential break

This break function allows settings of up to three break addresses and the number of times the instructions of the last address to be executed. While passing through all addresses sequentially in the order set, the program executes instructions at the final specified address the directed number of times, and then fetches the instruction at that address one more time before it breaks.

Table 3.10.5.4 Commands/menu item to set sequential break

Function	Command	Menu	Button
Set sequential break condition	bs	[Break Sequential Break...]	-
Clear sequential break condition	bsc	[Break Sequential Break...]	-

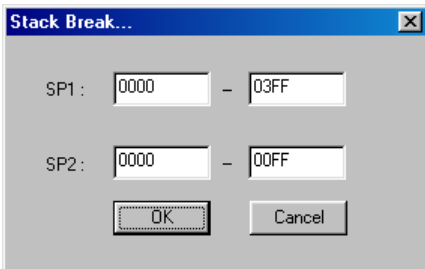


For example, if you execute the program after first setting a break address in two locations at addresses 0x1000 and 0x2000 and specifying 3 for the execution count using the *bs* command, the program executes address 0x2000 three times after executing address 0x1000 more than one time, and when the PC reaches 0x2000, it breaks before performing the 4th execution.

The execution count can be set up to 4,095.

(5) Accessing outside stack area

This break occurs when a location outside the stack area is accessed by stack pointer SP1 or SP2.



Before this function can be used, the SP1 and SP2 areas must be set by the *bsp* command. The initial value is 0x0 to 0x3ff for SP1, and 0x0 to 0xff for SP2. The address of SP1 must be specified in units of 4 words.

Table 3.10.5.5 Command/menu item to set stack break

Function	Command	Menu	Button
Set stack break condition	bsp	[Break Stack Break...]	-

Forced break by the [Key Break] button



[Key Break] button

The [Key Break] button can be used to forcibly terminate the program under execution when the program has fallen into an endless loop or cannot exit a standby (HALT or SLEEP) state.

Map break and illegal instruction break

The program also breaks when one of the following errors is encountered during program execution:

(1) Access to undefined program area

A break occurs when an undefined area of the program memory map is accessed.

(2) Access to undefined data area

A break occurs when an undefined area of the data memory map is accessed.

(3) Write to data ROM area

A break occurs when a write to the data ROM area is attempted.

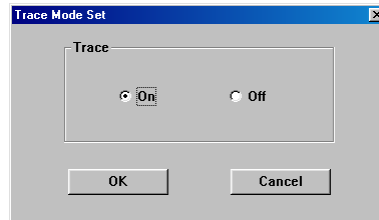
3.10.6 Trace Functions

The simulator has a function to trace program execution.

The trace function is initially disabled and can be enabled using the tm command.

Table 3.10.6.1 Trace mode setup command

Function	Command	Menu	Button
Set trace mode on and off	tm	[Trace Trace Mode Set...]	–



Trace data buffer and trace information

The simulator has a trace data buffer. When the simulator executes the program, the trace information on each executed instruction is taken into this buffer. The trace data buffer has the capacity to store information for 8,192 instructions. When the trace information exceeds this capacity, the data is overwritten, the oldest data first. Consequently, the trace information stored in the trace data buffer is always within 8,192 instructions. The trace data buffer is cleared when a program is executed, starting to trace the new execution data.

trace cycle	fetch addr	fetch code	fetch disasm	register	flag	data	trace
				A B X Y	E I C Z	addr data	SP in
00015	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00014	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21	r0
00013	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00012	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00011	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21	r0
00010	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00009	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00008	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21	r0
00007	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00006	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00005	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21	r0
00004	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--
00003	0126	0821	ldb %ext,0x21	0 0 3000 807A	1101	----	--
00002	0127	1AA1	bit [%y],0x01	0 0 3000 807A	0101	FF21	r0
00001	0128	06FD	jrz 0xfd	0 0 3000 807A	0101	----	--

The following lists the trace information that is taken into the trace data buffer in every instruction execution cycle. This list is corresponded to display in the [Trace] window.

- trace cycle: Trace cycle (decimal). The last information taken into the trace memory becomes 00001.
- fetch addr: Fetch address (hexadecimal).
- fetch code disasm: Fetch code (hexadecimal) and disassembled content.
- register: Values of A, B, X, and Y registers after cycle execution (hexadecimal).
- flag: States of E, I, C, and Z flags after cycle execution (binary).
- data: Accessed data memory address (hexadecimal), read/write (denoted by r or w at the beginning of data), and data (1-digit hexadecimal for 4-bit access; 4-digit hexadecimal for 16-bit access).
- SP: Stack access (1 for SP1 access; 2 for SP2 access).
- trace in: Unused

Displaying and searching trace information

The sampled trace information is displayed in the [Trace] window after a program execution has finished. In the [Trace] window, the entire trace data buffer can be seen by scrolling the window. The trace information can be displayed beginning from a specified cycle using a command. The display contents are as described above.

If the [Trace] window is closed, the information can be displayed in the [Command] window using a command.

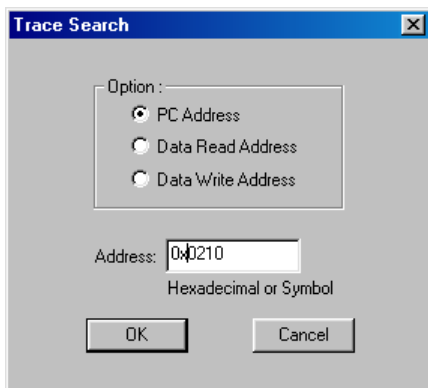
Table 3.10.6.2 Command/menu item to display trace information

Function	Command	Menu	Button
Display trace information	td	[View Trace]	-

It is possible to specify a search condition and display the trace information that matches a specified condition.

Table 3.10.6.3 Command/menu item to search trace information

Function	Command	Menu	Button
Search trace information	ts	[Trace Trace Search...]	-



The search condition can be selected from the following three:

1. Program's execution address
2. Address from which data is read
3. Address to which data is written

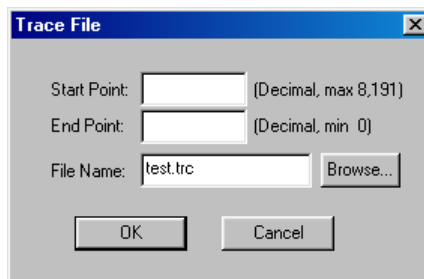
When the above condition and one address are specified, the system starts searching. When the trace information that matches the specified condition is found, the system displays the found data in the [Trace] window (or in the [Command] window if the [Trace] window is closed).

Saving trace information

The trace information within the specified range can be saved to a file.

Table 3.10.6.4 Command/menu item to save trace information

Function	Command	Menu	Button
Save trace information	tf	[Trace Trace File...]	-



3.11 Command List

Table 3.11.1 Command list

Classification	Command	Function
Program memory operation	a (as) [<addr> <mnemonic> [<file name>]]	In-line assemble
	pe [<addr> <code1> [..<code8>]]	Enter program code
	pf [<addr1> <addr2> <code>]	Fill program memory area
	pm [<addr1> <addr2> <addr3>]	Copy program memory area
Data memory operation	dd [<addr1> [<addr2>] [{-B -W -L -F -D}]]	Dump data memory data
	de [<addr. <data1> [..<data16>]]	Enter memory data
	df [<addr1> <addr2> <data>]	Fill data memory area
	dm [<addr1> <addr2> <addr3>]	Copy data memory area
	dw [<addr1> [..<addr4>]]	Set watch data address
Register operation	rd	Display register values
	rs [<reg> <value> [..<reg> <value>]]	Modify register value reg={pclalblxlylflsp1lsp2lxtlq}
Program execution	g [<addr1> [<addr2>]]	Execute program successively from current PC
	gr [<addr1> [<addr2>]]	Execute program successively after resetting CPU
	s [<step>]	Single stepping from current PC
	n [<step>]	Single stepping with skip subroutines
Reset CPU	rst	Reset CPU
Break	bp [<addr1> [..<addr16>]]	Set breakpoints
	bc [<addr1> [..<addr16>]]	Clear breakpoints
	bpc [<addr1> [..<addr16>]]	Clear breakpoints
	bd [<data> {rlwl*} <addr1> <addr2>]	Set data break condition
	bdc	Clear data break condition
	br [<reg> <value> [..<reg> <value>]]	Set register break condition reg={pclalblxlylflsp1lsp2lxtlq}
	brc	Clear register break condition
	bs [<pass> <addr1> [<addr2> [<addr3>]]]	Set sequential break condition
	bsc	Clear sequential break condition
	bsp [<addr1> <addr2> <addr3> <addr4>]	Set stack break condition
	bl	Display all break conditions
	bac	Clear all break conditions
Program display	u [<addr>]	Unassemble display
	sc [<addr>]	Source display
	m [<addr>]	Mix display
Symbol information	sy [{<\$<keyword> <#<keyword> }]/a]	Display symbol list
Load files	lf [<file name>]	Load IEEE-695 format absolute object file
	lo [<file name>]	Load Motorola S2 format file
	par [<file name>]	Load parameter file
Trace	tm [{on/off}]	Set trace mode
	td [<cycle>]	Display trace information
	ts [{pclrldw} <addr>]	Search trace information
	tf [[<cycle1> [<cycle2>]] <file name>]	Save trace information
Others	com [<file name> [<interval>]]	Load and execute command file
	cmw [<file name>]	Load and execute command file with execution interval
	rec [<file name>]	Record executed commands to file
	log [<file name>]	Logging
	ma	Display map information
	md [<option> <num> [..<option> <num>]]	Set modes option={-fl-ul-il-sl-cl-ill-cm}
	q	Quit debugger
	?	Display command usage

3.12 Component Mapping File (.cmp)

The component mapping file (peripheral setting file) is the text file to record the information required to simulate the CPU, key entry and LCD display by the simulator. The files that contain the internal peripheral circuit data for available models are provided. Use these files after adding the necessary information using a text editor. Since the simulator saves the LCD panel information set in the simulator, such as LCD panel color, to this file, do not set the file attribute to Read Only.

Example:

```
[ Internal ]
CPU=CPU.bmc
LCD=LcdDrv63.bmc
K/P/R port=KPRport.bmc
SVD=SVD63.bmc
Sound=Sound63.bmc
Serial=Serial63666.bmc (note 1)
Adc=Adc63.bmc (note 2)

[ Settings ]
CpuType=63666
OSC1=32.768KHz
OSC3=1.0MHz

[ External ]
EXT0=CMulDiv63.bmc, FF80h, FF86h
```

[Internal] Internal peripheral circuit parameters

The file does not allow the user to modify these parameters. However, in a development environment in which the S1C63/S1C88 simulator has already been installed, components that support the serial interface must be added here (note 1). The same applies when using the A/D converter (note 2). When using some type of MCU that does not incorporate the serial interface or A/D converter, set Serial = NullDev.bmc for (note 1), or Adc = NullDev.bmc for (note 2).

[Settings] CPU parameters

All of these parameters must be specified as follows:

```
[ Settings ]           ⇐ Start of Settings section
CpuType=63666         ⇐ Family name
OSC1=32.768KHz       ⇐ OSC1 clock frequency
OSC3=1.0MHz          ⇐ OSC3 clock frequency
```

[External] External device parameters

When parameters have been described in the file, do not modify or delete them.

```
[ External ]           ⇐ Start of External section
EXT0=CMulDiv63.bmc, FF80h, FF86h
```

Definition: Ext<N>=<device>.bmc, <Start_addr>, <End_addr>, <Option>

Ext<N>: N is a sequential device number beginning from 0.

<device>.bmc: Peripheral device definition file name

<Start_addr>: I/O map start address

<End_addr>: I/O map end address

<Option>: Peripheral specific option

Setting the BkLight

When using a backlights (BkLight), describe the information as follows:

```
Ext1=BkLight.bmc, FF31h, FF31h, 1L
```

Specify the I/O address of the port connected to the backlight. The start and end addresses have the same value. The option allows definition of the bit number and active level (H or L) to turn the backlight on. The example shown above defines so that the backlight goes on when bit 1 is set to low.

3.13 IOT File (.iot)

The IOT file is a text file in which settings and input data is written to simulate the general-purpose port and serial interface input/output operations and A/D conversions. These settings and data is loaded using the [I/O Terminal] window menu before performing input/output simulation. For more information on the [I/O Terminal] window and on using the window for input/output simulation, refer to Section 3.6.4, "[I/O Terminal] Window". To perform input/output simulation, first create an IOT file using a text editor as described below. Save the created data to a file using the file extension ".iot".

File contents

Write the input data, one for each section, e.g., [General I/O] and [A/D Converter]. Although the sequence of the sections does not matter, the data in each section is processed in the order in which they are written. No data need to be written for sections whose functions are not used.

Example: test.iot

<pre>[General I/O] watch P00, P01, P02, P10, P11, P20 5.0 K00=1 5.5 K00=0 8.0 K00=H 8.5 K00=L [A/D Converter] AVdd=3.0 AVss=0.2 AVref=2.9 [A/D CH2] 2.40,2.39,2.38,2.37,2.36 2.00, 2.10, 2.20, 2.30 0.5 0.65 0.80 [Serial CH1] "aBcDeFg" 0x25,0x20,0x41</pre>	<p>↓ Order in which the data is processed</p> <p>↓ Order in which the data is processed</p> <p>↓ Order in which the data is processed</p>	<p>Section (general-purpose input/output ports)</p> <p>Section (A/D converter voltage settings)</p> <p>Section (A/D channel 2 input settings)</p> <p>Section (Serial interface channel 1)</p>
---	---	---

General-purpose input/output ports

Write settings for general-purpose input/output ports in the [General I/O] section.

Specifying monitor ports

Specify the general-purpose input/output ports whose input/outputs are to be monitored with a watch statement.

Example: watch P00, P01, P02, P10, P11, P20

In the watch statement, write the port names whose input/outputs are to be checked (Kxx, Rxx, Pxx). When input/outputs on the ports written here change state from high to low or vice versa, a log is displayed on the [I/O Terminal] window. When not monitoring the port state, there is no need to write port names.

Specifying ports for input

When using general-purpose input/output ports for input, write the time (in seconds) and the port name for which to input signal and the input level applied to the port on a line by line basis. Input simulations on the ports are performed in the order they are written, beginning with the one written at the top.

Example: 5.0 K00=1
8.5 K00=0
+0.7 P02=H, P10=L

The beginning of each line indicates the time (in seconds) at which the signal is input to the port. The time can be set with a precision of up to 1 ms, using the decimal point. The values set here basically represent an elapsed time after reset. Or the values can be specified to indicate an elapsed time after the previous input by prefixing the value with the positive sign "+".

3 S1C63 FAMILY SIMULATOR

Enter this specification in the form of "port name = input level". Write H or 1 for the high-level input, and L or 0 for the low-level input.

In the above example, a high signal is input to the K00 port 5.0 seconds after executing a reset instruction. Likewise, a low signal is input to the K00 port 8.5 seconds later, and a high and a low signal respectively are input to the P02 port and P10 port at the same time 9.2 seconds later.

If input port (K port) interrupts have been enabled, an interrupt is generated synchronously with the input timing.

A/D converter

Write settings for the A/D converter in the [A/D Converter] and [A/D CHn] sections.

Setting the A/D converter voltages

In the [A/D Converter] section, write the converter's set voltages.

Example: [A/D Converter]

```
AVdd=3.0
AVss=0.2
AVref=2.9
```

The AVdd, AVss, and AVref lines represent the A/D converter's positive power supply voltage, negative power supply voltage, and reference voltage respectively. Write the voltage values in units of volts. The voltages can be set with a precision of up to 1 mV, using the decimal point. When using the A/D converter, always write the [A/D Converter] section that includes these three lines.

Specifying analog input voltages

In the [A/D CHn] section, write the input voltages to the A/D converter. Specify the input voltages for each channel. Input voltages do not need to be specified for unused channels.

Example: [A/D CH4]

```
2.40,2.39,2.38,2.37,2.36
2.00, 2.10, 2.20, 2.30
[A/D CH5]
0.5
0.65
0.80
```

Write the input voltages in units of volts. The voltages can be set with a precision of up to 1 mV, using the decimal point. Each time the target program executes A/D conversion, the written data is loaded sequentially from the top. When writing input voltages in one line, separate each entry with a comma (.). In this case, the data is loaded sequentially from the left.

If interrupts are enabled, an interrupt is generated a certain time (sampling time + A/D conversion time) after the input began.

If A/D conversion is performed on any channel which does not have data specified, channel processing assumes a level-0 voltage (equivalent to AVss) input. If A/D conversion is performed for more than a specified data count, the last data specified is used repeatedly.

Furthermore, if a value smaller than AVss is specified, it is read out as level-0 (equivalent to AVss); if a value larger than AVref is specified, it is read out as the maximum level (equivalent to AVref).

Serial interface

Write settings for the serial interface in the [Serial CHn] section. Make this setting for each channel.

For some types of MCU that contain only a single channel, specify "CH1" (channel 1). No data needs to be written for unused channels.

Example: [Serial CH1]

```
"aBcDeFg"
0x25,0x20,0x41,FER
```

Write the input data in either hexadecimal notation or string notation (enclosed in " "). Data in different notations cannot coexist in one line. Each time the program receives a signal via the serial interface, the written data is loaded sequentially from the top. When writing multiple hexadecimal data in a single line, delimit each entry with a comma (.). In this case, the data is loaded sequentially from the left.

If interrupts are enabled, an interrupt is generated synchronously with the timing at which the program finishes sampling the input. No interrupts are generated when signal is received on a channel that has no specified data.

The time required for sampling

Asynchronous system:	Cycle time of the clock source selected with the I/O register < number of bits (+ parity bit)
Clock-synchronous system (master):	Cycle time of the clock source selected with the I/O register < number of bits
Clock-synchronous system (slave):	User-defined transfer rate (bps)

For the clock-synchronous system (slave), specify the transfer rate (bps) in the manner shown below. Enter this specification before writing data. This setting is effective for only the synchronous system (slave).

```
Example: [Serial CH1]
         bps=2400
         "aBcDeFg"
         0x25,0x20,0x41,FER
```

Instead of hexadecimal data, the following can be specified:

```
FER: Framing error
PER: Parity error
OER: Overrun error
```

These data are valid only when the serial interface supports error interrupts. Do not specify this data for types of MCU that do not support error interrupts.

If interrupts have been enabled when this data is loaded, an error interrupt is generated.

For serial interface output, no particular specification needs to be written. When an output to any channel occurs, a log is displayed in the [I/O Terminal] window.

3.14 Simulator Project File (.spj)

The simulator project file is the text file that contains the names of the parameter file, LCD panel definition file, component mapping file and port setting file to be used. Use a text editor to create it. This file allows simultaneous selection of the files described above.

When the simulator starts up or the par command ([File | Load Parameter File]) is executed, a dialog box appears to select a parameter file or a simulator project file. When selecting a parameter file, a simulator project file is not necessary.

The following is an example of the file contents.

Example:

```
[Setting]
PAR=63666.par      ⇐ Parameter file
LCD=63666.lcd     ⇐ LCD panel definition file
CMP=63666.cmp     ⇐ Component mapping file
PRT=63666.prt     ⇐ Port setting file
```

3.15 Restrictions

- For the MPU models supported by the simulator, please refer to the "Supported MCU models and peripherals" described in the Readme_E.txt.
- The supported external devices are monochrome LCD panels, backlight, keys and key matrix.
- This simulator performs instruction-level simulation. Therefore, execution cycles shorter than the instruction cycle cannot be simulated.
- Since timers are simulated based on the instruction cycle, the timings are different from those of the actual hardware.
- The following functions are not supported:
 1. Buzzer output
 2. TOUT/FOUT output
 3. D/A converter and analog comparator
 4. Event counter mode of programmable timer
- To simulate the sound generator, a sound card that supports playback of PCM sound source must be installed in the PC.
- Two or more simulators cannot be executed simultaneously for multiple simulations.
- When using the multiple-key entry reset function of the K ports, the simulator cannot reset the CPU even if the four assigned keys are pressed simultaneously. To reset using four keys, press three keys immediately after pressing one other key or press two keys immediately after pressing two other keys. The CPU will be reset immediately without authorization time.
- The debugging functions of this simulator are compatible with the S1C63 Family debugger db63 including the menu and dialog configuration. However the functions shown below are disabled because the ICE is not used.
 1. On-the-fly function
 2. Execution time measurement function (cycle counter is available)
 3. Flash memory operation function (lfl, sfl, efl)
 4. Coverage function (cv, cvc)
 5. Option data dump function (od)
 6. Single delay trigger trace and PC area trace functions
 7. Break and trace trigger using external signals

For the restrictions, support functions and bug information of the latest version, refer to rel_sim63_E.txt of S5U1C88000Q.

For the models supported by the latest version, refer to Readme_E.txt of S5U1C88000Q.

3.16 Simulator Messages

Status message list

Status message	Content of message
Break by PC break	Break caused by PC breakpoint
Break by data break	Break caused by data break condition
Break by register break	Break caused by register break condition
Break by sequential break	Break caused by sequential break condition
Key Break	Break caused by [Key Break] button
Break by accessing no map program area	Break caused by accessing undefined program-memory area
Break by accessing no map data area	Break caused by accessing undefined data-memory area
Break by accessing ROM area	Break caused by writing to data ROM area
Out of SP1 area	Break caused by accessing outside SP1 stack area
Out of SP2 area	Break caused by accessing outside SP2 stack area

Error message list

Error message	Content of message (Commands involved)
Address out of range, use 0–0xXXXX	The specified program memory address is out of range. (a/as, pe, pf, pm, sc, m, u, g, gr, bp, bc, bs, tm, ts, cv)
Address out of range, use 0–0xFFFF	The specified data memory address is out of range. (dd, de, df, dm, dw, bd, ts)
Cannot load program/ROM data, check ABS file	Failed to load program/ROM data; some file other than IEEE-695 executable format was specified. (lf)
Cannot open file	The file cannot be opened. (lf, lo, com, cmw, log, rec)
Data out of range, use 0–0xF	The specified number is out of the data range. (de, df)
Different chip type, cannot load this file	A different ICE parameter is used in the file. (lf)
end address < start address	The start address is larger than the end address. (pf, pm, df, dm, bd, cv)
error file type (extension should be CMD)	The specified file extension is invalid. (com, cmw)
illegal code	The input code is not available. (pe, pf)
illegal mnemonic	The input mnemonic is invalid for S1C63000. (a/as)
Incorrect number of parameters	The parameter number is incorrect. (All commands)
Incorrect option, use -f/-u/-i/-s/-c/-il/-cm	An invalid mode setting option was specified. (md)
Incorrect r/w option, use r/w/*	An illegal R/W option was specified. (bd)
Incorrect register name, use A/B/X/Y/F	An invalid register name was specified. (br)
Incorrect register name, use PC/A/B/X/Y/F/SP1/SP2/EXT/Q	The specified register name is invalid. (rs)
Input address does not exist	Attempt is made to clear a break address that has not been set. (bp)
invalid command	This is an invalid command. (All commands)
invalid data pattern	The input data pattern is invalid. (bd, br)
invalid file name	The file name (extension) is invalid. (lo)
invalid value	The input data, address or symbol is invalid. (All commands)
Maximum nesting level(5) is exceeded, cannot open file	Nesting of the com/cmw command exceeds the limit. (com, cmw)
no such symbol	There is no such symbol. (All symbol support commands)
no symbol information	No symbol information is available since the ".abs" file has not been loaded. (sy)
Number of passes out of range, use 0–4095	The specified pass count for sequential break is out of range. (bs)
Number of steps out of range, use 0–65535	The specified step count is out of range. (s, n)
SP1 address out of range, use 0–0x3FF	The specified SP1 address is out of range. (bsp)
SP2 address out of range, use 0–0xFF	The specified SP2 address is out of range. (bsp)
symbol type error	The specified symbol type (program/data) is incorrect. (All symbol support commands)

Warning message list

Warning message	Content of message (Commands involved)
Break address already exists	Attempt is made to set an already-set break address. (bp)
Identical break address input	Input command contains identical address.
round down to multiple of 4	Watch data address is invalid. (dw)

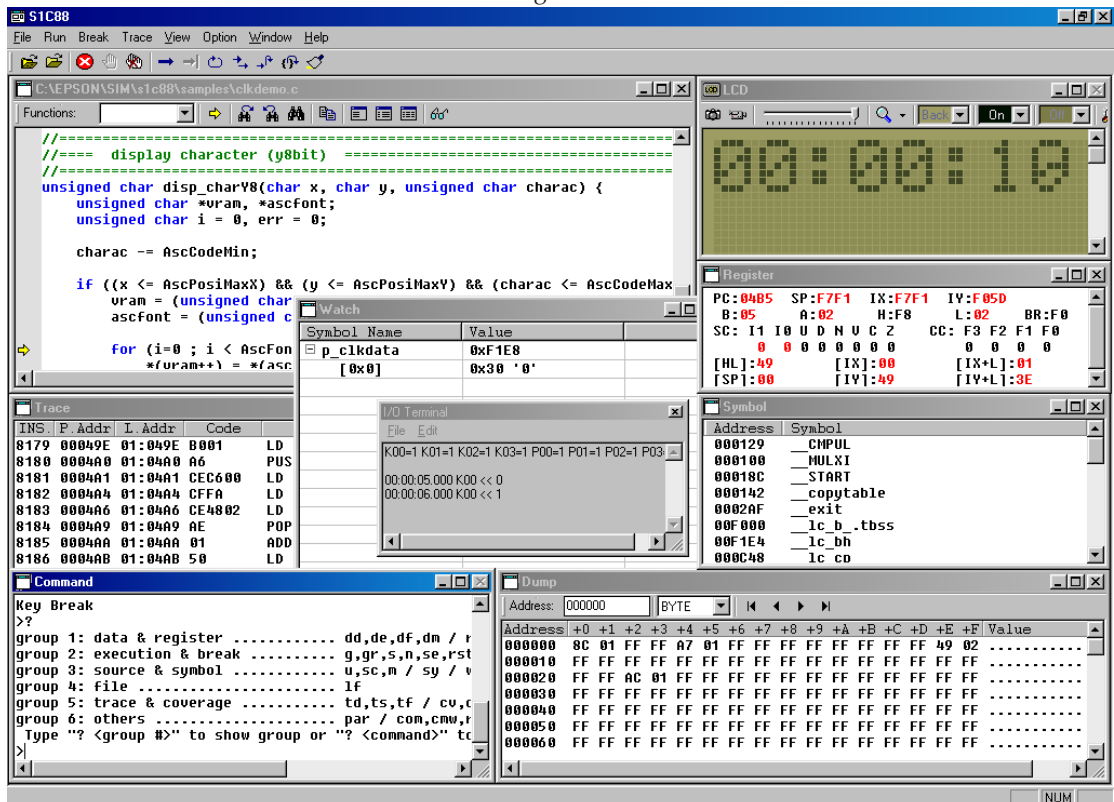
4 S1C88 FAMILY SIMULATOR

4.1 Overview

The Sim88 simulator is a development tool for the S1C88 Family of 8-bit single-chip microcomputers. The simulator included in this package allows you to debug software created with the S1C88 integrated tool (C compiler, assembler) using just a PC, without an in-circuit emulator (ICE) or other dedicated hardware. In addition to providing general simulator functions, the simulator simulates push-buttons or a key matrix that use I/O ports, inputs/outputs of serial/general ports, A/D conversion and LCD displays. The package includes utilities for creating bitmap and LCD panel data.

The simulator has the following features and functions:

- Operations including LCD panel display can be simulated with a PC alone without any debugging hardware.
- Various data can be referenced at the same time using multiple windows.
- Frequently used commands can be executed from tool bars and menus using a mouse.
- Also available are C source, disassembled code and symbol display functions.
- Consecutive program execution and three types of single-stepping are possible.
- Three break functions are supported.
- Trace and coverage functions.
- An automatic command execution function using a command file.



Note: For information about the operating system (OS) environment, refer to *Readme_E.txt* of *S5U1C88000Q*.

4.2 Software Development Flow

Figure 4.2.1 shows the typical software development flow for the S1C88 Family. The items in bold indicate tools and related files provided in this package.

Note: In addition to this package, software development for the S1C88 Family requires the S1C88 Family Integrated Tool Package.

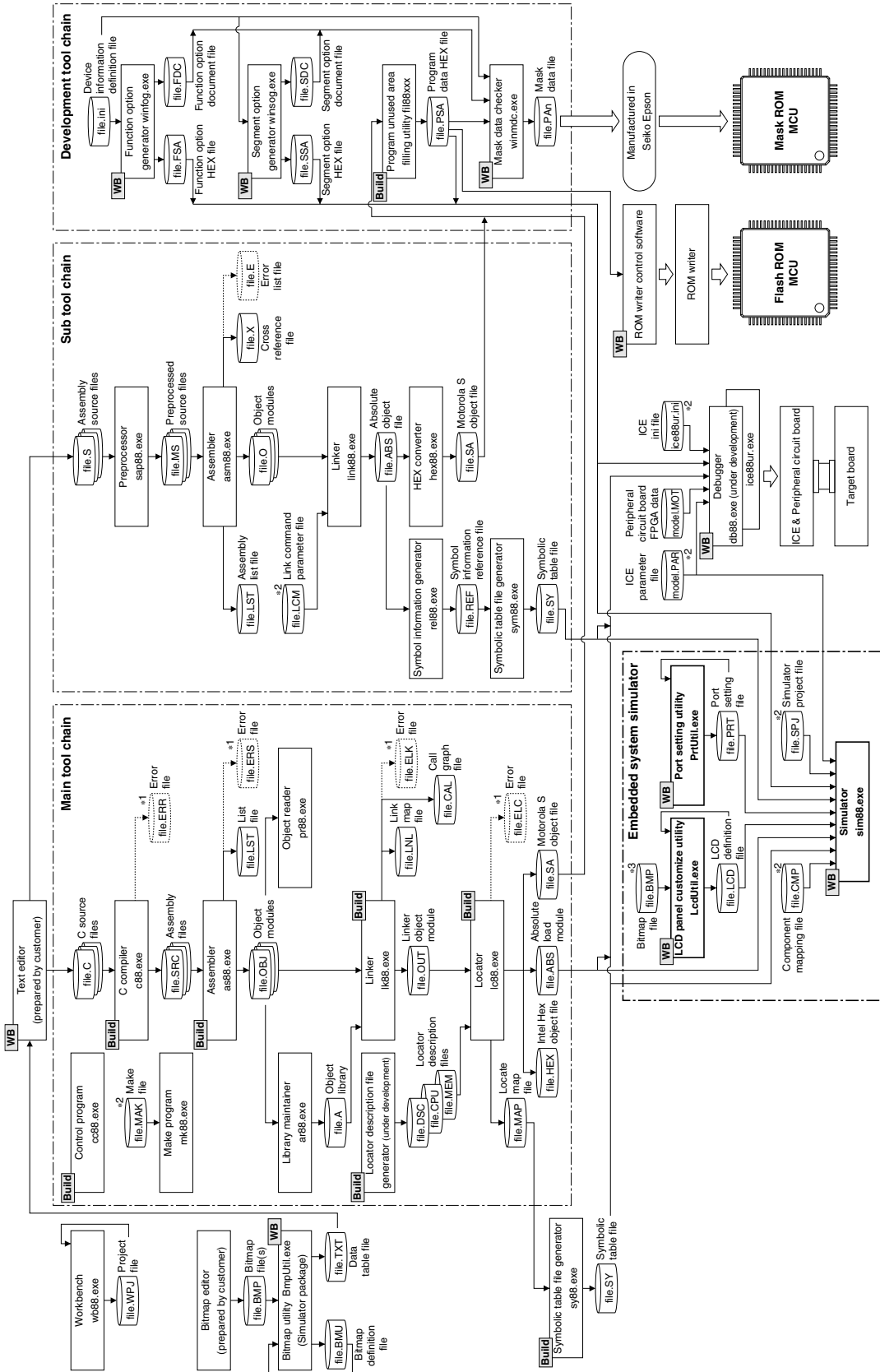


Fig. 4-2.1 Software development flow

4.3 Overview of Simulation Functions and Operations

This section explains how the target application is simulated on a PC with the simulator sim88.

Supported MPUs

For the MPU models supported by the simulator, please refer to the "Supported MCU models and peripherals" described in the Readme_E.txt.

Target applications

This simulator is best suited to simulating applications that require key input and LCD display functions, such as watches, pocket calculators, electronic pocketbooks, and portable games. OSC1 clock operation can be executed in real time. For OSC3 operation, real-time execution is possible in the range of 1 MHz to 2 MHz. (With PCs running on a 400-MHz Pentium or equivalent or faster and 64 MB RAM, real-time execution for approximately 1 MHz is possible.) However, for reasons involving instruction-level simulation accuracy, it is not possible to simulate high-accuracy timing tasks such as those for control systems.

Support for external devices is limited to ROM, RAM, the LCD drivers listed below, monochrome LCD panels, backlight, keys, and a key matrix.

Serial inputs/outputs, A/D conversions and general port inputs/outputs can be simulated using the text I/O terminal plug-in module.

Supported LCD drivers: S1D15210, S1D15600, S1D15601, S1D15602, S1D15605, S1D15606, S1D15607, S1D15608

Note: Simulation on a PC is subject to some other limitations. See Section 4.15, "Restrictions", and the Readme_E.txt file.

Entering schematic information into the simulator

To set the information required to simulate the operation of external devices, load data from files.

ROM and RAM mapping

Use a parameter file (file.par) to set the addresses to which devices are mapped. For more information on creating this file, see the S5U1C88000C Manual.

Mapping of external devices and the internally-generated clock frequency

Create a component mapping file (file.cmp) by writing addresses to which the LCD driver (S1D15210), backlight, and a backlight control bit are mapped, then load the file into the simulator. This allows simulated control of these external devices. Use the component mapping file to set simulation conditions such as OSC1/OSC3 oscillation clock frequencies. For more information on creating this file, see Section 4.12, "Component Mapping File (.cmp)".

Entering key and key matrix specifications

Create a port setting file (.prt) containing a description of the relationship between key input ports and the I/O ports comprising the key matrix and target and PC keyboards, then load the file into the simulator. This allows simulated key input with the PC keyboard. For more information on creating this file, see Chapter 7, "Port Setting Utility".

Entering LCD design

Create an LCD panel definition file (file.lcd) containing a record of LCD panel layout and SEG/COM port assignments, then load the file into the simulator. This allows simulated display on LCD panels. For more information on creating the LCD panel definition file, see Chapter 5, "LCD Panel Customize Utility".

These files normally are loaded when the simulator starts.

Loading and executing the target program

Use the simulator if the lf command to load the target program and the g (or gr, s, n, se) command to run it. To simulate the LCD display or key input during program execution, use the simulator's [LCD] and [I/O Terminal] windows.

LCD display: Displayed in the [LCD] window as on an actual LCD.

Key input: Activate the [LCD] window and type the appropriate key on the PC keyboard. You can also set and verify key input status in a dedicated window.

SVD evaluation: Manipulate the SVD slide bar on the [LCD] window to simulate SVD function.

Inputs/outputs of the serial interface and general ports, and A/D conversion:

Activate the [I/O Terminal] window and load an IOT file in which an input sequence is described to simulate the input/output functions.

4.4 Input/Output Files

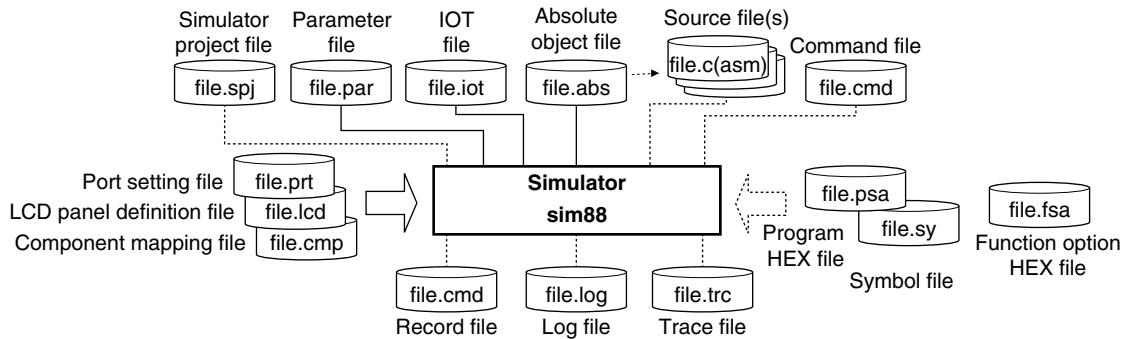


Fig. 4.4.1 Input/output files

Parameter file (file_name.par)

This text file contains memory information on each microcomputer model and is used to set the memory mapping information to the simulator. For the contents of this file, refer to the S5U1C88000C Manual.

Absolute object file (file_name.abs)

This is an IEEE-695 object file generated by the locator. By reading a file in this format that contains debug information, C source display and symbolic debugging can be performed.

Source file (file_name.c, file_name.asm)

This is the source file of the above object file. It is read when the simulator performs source display.

Internal ROM data HEX file (file_name.psa)

This is the program file generated by the fil88xxx unused ROM area FF filling utility in Motorola S2 format file. The unused area of the built-in ROM has been filled with FFH and the system code is set to the system reserved area.

Function option HEX file (file_name.fsa)

This is the mask option setup file in Motorola S2 format that is generated by the function option generator.

Symbol information file (file_name.sy)

This is the symbol information file generated by the symbol table file generator. By preparing the file with the same name as the program file in the same directory as the program file, it will be automatically loaded at the same time the program is loaded. This file allows the simulator to display the symbols defined in the source.

Simulator project file (file_name.spj)

This file is used to specify a parameter file, LCD panel definition file, component mapping file and port setting file at the simulator start up. Enter file names using an editor to create this file. The simulator can be started up if this file does not exist by selecting the files from a dialog box.

IOT file (file_name.iot)

This is the text file in which the input data used to simulate serial/general port inputs/outputs and A/D conversions are described.

LCD panel definition file (file_name.lcd)

This file includes an LCD panel layout bitmap and SEG/COM port allocation information. Create this file using the LCD panel customizing utility (LcdUtil).

Component mapping file (file_name.cmp)

This is the text file that sets the addresses where the external LCD driver and backlight are mapped.

Port setting file (file_name.prt)

This is the text file in which push keys, key-matrix configuration and the corresponding between the keys and ports are described.

Command file (file_name.cmd)

This text file contains a description of debug commands to be executed successively. By writing a series of frequently used commands in this file, the time and labor required for entering commands from the keyboard can be saved. The command described in the file are read and executed using the com command.

Log file (file_name.log)

This text file contains the executed commands and execution results. Output of this file can be controlled by the log command.

Record file (file_name.cmd)

This text file contains the executed commands. Output of this file can be controlled by the rec command. This command can be used as a command file.

Trace file (file_name.trc)

This text file contains the specified range of trace information. Output of this file can be controlled by the tf command.

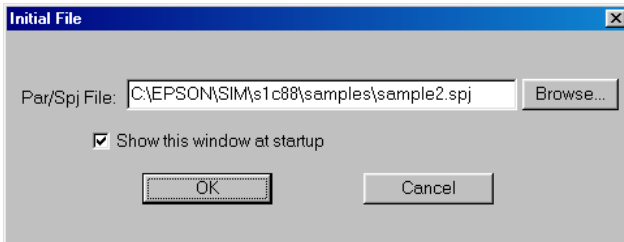
4.5 Starting and Terminating the Simulator



sim88.exe

Double-clicking this icon to start the simulator.

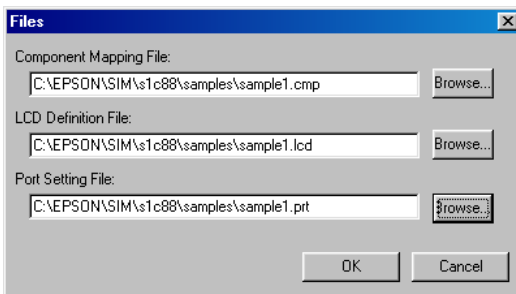
The dialog box shown below appears at the first time the simulator starts up. Enter the parameter file name or simulator project file name to the text box, or choose it using the [Browse...] button (see Section 4.14 for the simulator file).



If the [Show this window at startup] check box is deselected, this dialog box will not appear from the next startup of the simulator and the same file will be selected. After this, use the par command ([File | Load Parameter File...]) to redisplay this dialog box for changing the file.

When the check box is left on, the currently selected file name will appear in the text box at the next startup of the simulator allowing choose of the file by clicking the [OK] button only.

When a parameter file is selected, the dialog box shown below appears.



Enter the component mapping file name, LCD panel definition file name and port setting file name to the respective text boxes or choose them using the [Browse...] button. This dialog box will appear at the next start-up of the simulator even if the [Show this window at startup] is deselected. However, the selected file names will appear in the text boxes at the next startup of the simulator allowing choice of the files by clicking the [OK] button only.

This dialog box will not appear when a simulator project file is selected.

When invoking the simulator from the work bench wb88, these files are specified in a dialog box of the work bench. So the dialog boxes shown above do not appear when the simulator starts up.

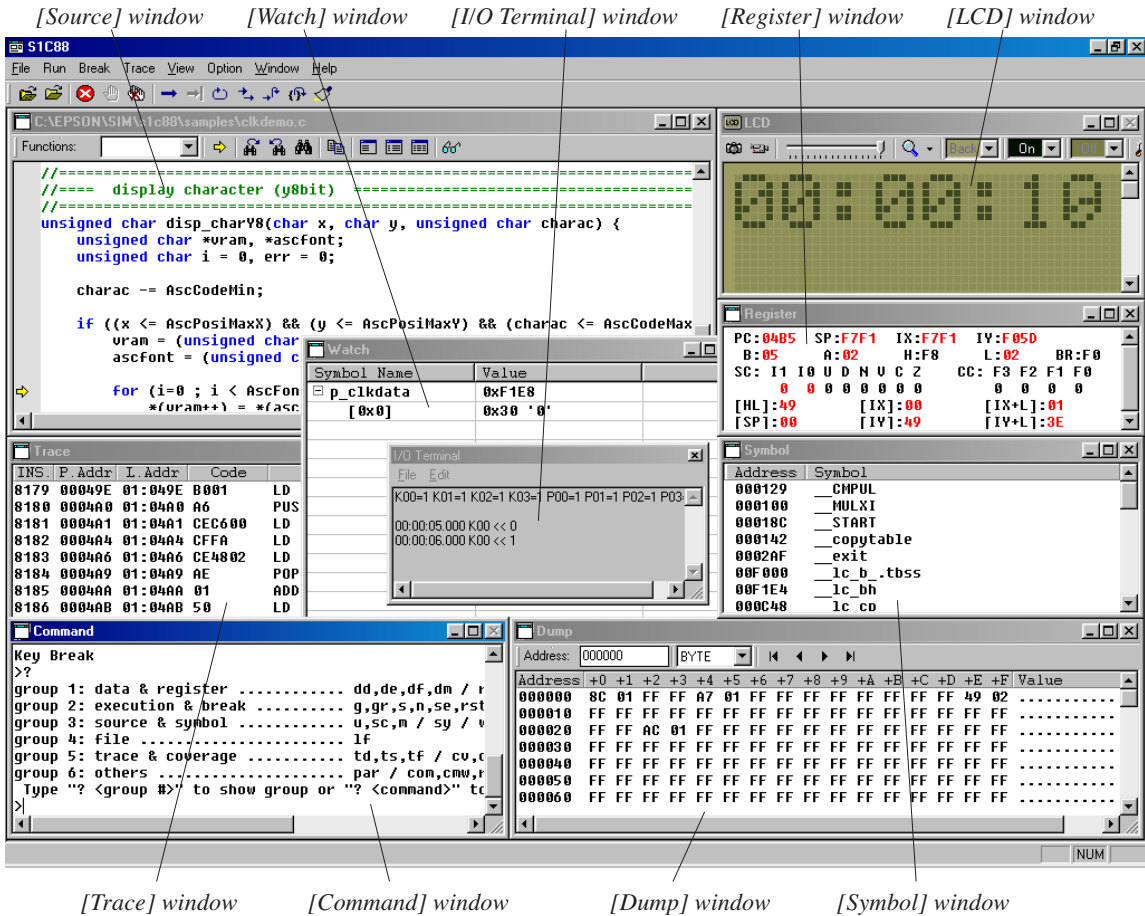
Select [Exit] from the [File] menu to terminate the simulator.

4.6 Windows

This section describes the types of windows used by the simulator.

4.6.1 Basic Structure of Window

The diagram below shows the window structure of the simulator.



Features common to all windows

(1) Open/close and activating a window

All windows except [Command] and [LCD] can be closed or opened.

To open a window, select the window name from the [View] menu. When a command is executed, the corresponding window opens if the command uses the window for displaying the executed results.

To close a window, click the [Close] box on the window.

The opened windows are listed in the [Window] menu. Selecting one from the list activates the selected window. It can also be done by simply clicking on an inactive window. Furthermore, pressing [Ctrl]+[Tab] switches the active window to the next open window.

(2) Resizing and moving a window

Each window can be resized as needed by dragging the boundary of the window with the mouse. The [Minimize] and [Maximize] buttons work in the same way as in general Windows applications. Each window can be moved to the desired display position by dragging the window's title bar with the mouse. However, windows can only be resized and moved within the range of the application window.

(3) Other

The opened windows can be cascaded or tiled using the [Window] menu.

4.6.2 [Command] Window

```

Command
>g
Key Break
PC:0105 SP:F7EE IX:0000 IY:0100
B:01 A:00 H:10 L:00 BR:F0
SC:11 I0 U D N U C Z CC:F3 F2 F1 F0
  0 0 0 0 1 0 1 0      0 0 0 0
>dd
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
000000 8C 01 FF FF A7 01 FF FF FF FF FF FF FF FF 49 02 .....I.
000010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000020 FF FF AC 01 FF FF FF FF FF FF FF FF FF FF FF FF .....
000030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
>
  
```

The [Command] window is used to do the following:

(1) Entering debug commands

When the prompt ">" appears in the [Command] window, the system will accept a command entered from the keyboard.

(2) Displaying debug commands selected from menus or tool bar

When a command is executed by selecting the menu item or tool bar button, the executed command line is displayed in the [Command] window.

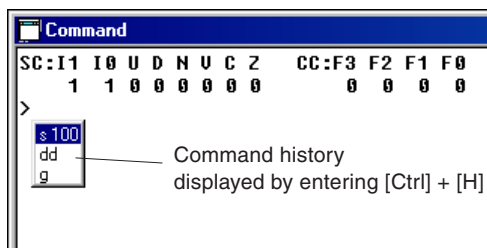
(3) Displaying command execution results

The [Command] window displays command execution results. However, some command execution results are displayed in the [Source], [Dump], [Register], [Symbol] or [Trace] windows. The contents of these execution results are displayed when their corresponding windows are open. If the corresponding window is closed, the execution result is displayed in the [Command] window.

When writing to a log file, the content of the write data is displayed in the window. (Refer to the description for log command.)

(4) Displaying the command history

Sim88 stores up to the 32 most recent commands executed since startup in memory. (If any command has been executed twice or more, it is registered only once.) The commands stored in memory can be recalled by entering the [Ctrl] + [H] keys when the [Command] window is active.



4 S1C88 FAMILY SIMULATOR

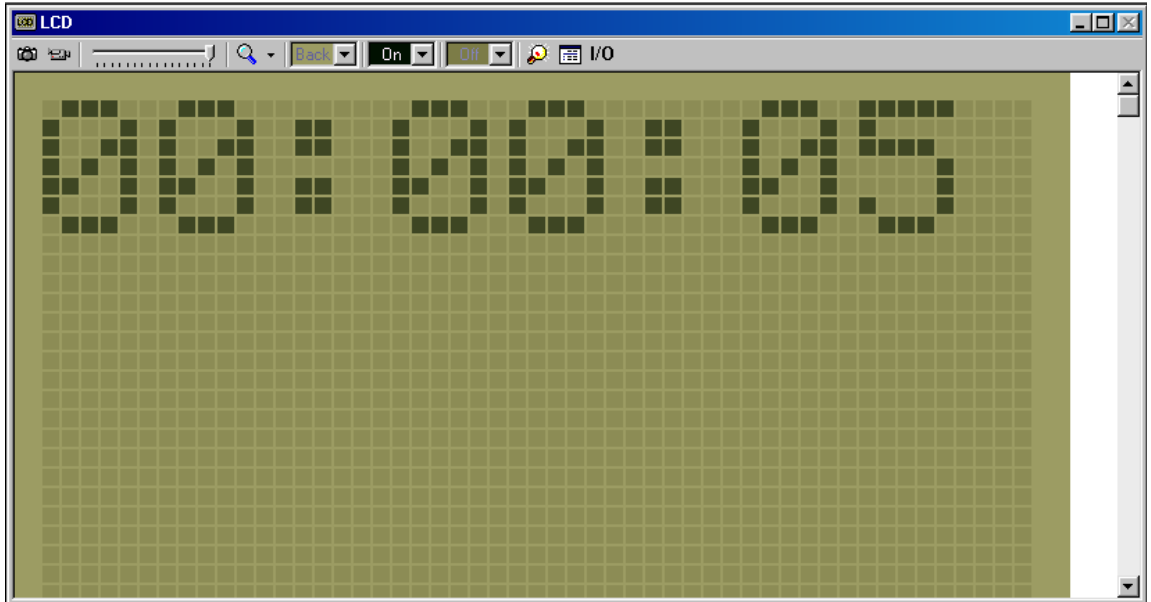
- Simply enter [Ctrl] + [H] to display a command history in popup list form. Double-click a command to repeat, or select a command with the up or down arrow keys and press [Enter]. The command is pasted into the prompt position. It can then be executed by pressing the [Enter] key. If the command history has only one previous command registered, the command is pasted directly into the prompt position without being displayed in a popup list.
- Enter [Ctrl] + [H] after entering any character to display a command history in one of the following ways:
 - If the command history has several commands registered that begin with that character (string), those commands are listed. Then, when another character (string) is entered, one of the recently executed commands among those listed is selected (highlighted) that includes the character (string).
 - If the command history contains only a single command registered that begins with the character (string), the command is pasted directly into the prompt position.
 - If the command history does not contain any commands registered that begin with the character (string), no operation is performed.

For example, if the command history contains the three commands dd, sy, and s:

- Enter [Ctrl] + [H] after entering the character 's'. The commands s and sy are listed. Here, the recently executed command s is displayed above the other commands and highlighted.
- If you follow by entering a 'y', command sy is highlighted.
- Enter [Ctrl] + [H] after entering the character 'd' to paste the command dd into the prompt position.

Note: The [Command] window cannot be closed.

4.6.3 [LCD] Window

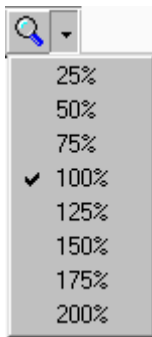


The [LCD] window has the following functions:

(1) LCD display simulation

Displays the LCD panel defined in the LCD panel definition file. The icons and dot matrix change their display status according to the program being executed.

Furthermore, the panel size and the color can be set using the following controls.



Scaling button & drop-down list

The panel size is magnified in 25% steps every time the button is clicked. If the size has reached 200%, the next click reduces the size to 25%. The drop-down list allows direct selection of the magnification rate.

[Back], [On], [Off] drop-down lists

Set the LCD panel color.

[Back] Background color

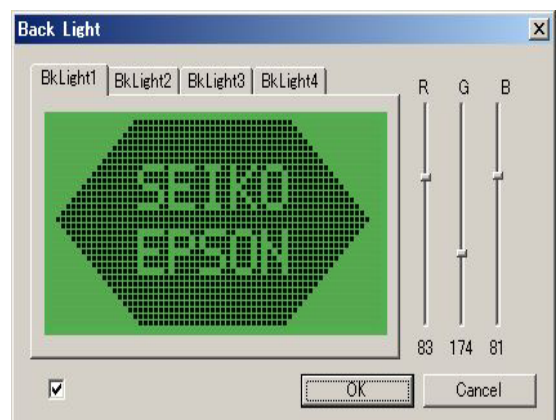
[On] Dot color when it is on

[Off] Dot color when it is off



[Backlight] button

When this button is clicked, the dialog box shown below appears allowing registration of up to 4 backlight colors. The color can be adjusted using the RGB sliders and selecting the lower left check box turns the backlight on.



(2) Capturing the panel image



Camera button

Clicking this button captures the LCD panel image at that point and then transfers it to the clipboard as a bitmap image. The captured image can be pasted to paint software to print and/or saved to a file.

Since this button is invalid during program execution, program execution must be broken at the desired image before it can be captured.



Video button

Clicking this button enables the panel image recording function (can be saved as an AVI file).

When starting the program execution in this status, two dialog boxes to specify a file name and a video compression format appear sequentially. Enter a file name and choose an available video compression format. The recording continues until the program execution is broken.

The generated file can be played back with the Windows standard medium player.

This button cannot be cancelled after it has been clicked once. To cancel the recording, click on the [Cancel] button in the file name input dialog box. The program will be executed, but the recording is cancelled.

Note: The recording operation reduces the program execution speed.

(3) Key entry simulation

If the program being executed is waiting a key entry, key entry operation can be simulated using the PC keyboard after activating the [LCD] window.

The correspondence between PC keys and ports should be defined in the port setting file.

This definition list can be displayed using the [Key List] button.



[Key List] button

Port	Target Key	PC Key
<input type="checkbox"/> K00	Key A	ESC
<input type="checkbox"/> K01	Key B	Shift
<input checked="" type="checkbox"/> K02	Key C	Ctrl
<input type="checkbox"/> K03	Key D	Key-1
<input type="checkbox"/> K04	Key E	Key-2
<input type="checkbox"/> K05	Key F	A
<input type="checkbox"/> K06	Key G	B

The displayed contents are port name, target key name and PC key name, respectively from the left. The correspondence between PC keys and target keys can be verified here. The symbol on the left indicates the port status; H indicates that the port is high level and L indicates low level. The input level is different by the port specification of the model.

The key entry status can also be set in break status. The key entry status is maintained during the next program execution until the user operates the key. This allows setting of the key entry status during single-step operation.

The input port is fixed at active level if the pressed key is directly connected to the input port. In the case of a key matrix, the input port goes active level only when the corresponding output port goes active level.

(4) SVD simulation



This slider changes the detection level of the SVD to 16 steps.

(5) Simulating serial/general-purpose port input/output and A/D conversion

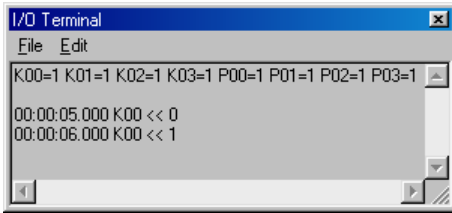
Use the [I/O] button to display the [I/O Terminal] window and to load an IOT file. This allows simulation of serial interface input/output, general-purpose port input/output, and A/D conversion. For more information, refer to the next section.



[I/O] button

Note: The [LCD] window cannot be closed.

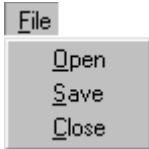
4.6.4 [I/O Terminal] Window



The [I/O Terminal] window provides an I/O text terminal function to display the input/output status of the general-purpose ports, serial interface, and A/D converter by simulating their inputs/outputs.

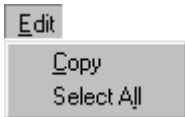
(1) Menu

[File] menu



- Open Loads an IOT file.
- Save Saves the logs displayed in the window to a text file.
- Close Closes the IOT file (halts I/O simulation).

[Edit] menu



- Copy Copies a data range selected within the window to the clipboard.
- Select All All selects all logs in the window.

* The Copy/Select All menu can also be displayed by right-clicking anywhere in the window.

(2) IOT file

The IOT file is a text file that contains a description of the following specifications. It is created in a general-purpose editor.

- a. Specify the general-purpose ports (Rxx, Pxx) to monitor the input/output status.
- b. Specify the input timing to and the input level of general-purpose ports (Kxx, Pxx).
- c. Specify the power supply voltage and the reference voltage of the A/D converter.
- d. Specify the A/D input voltage.
- e. Specify the serial interface input data.

Example:

```
[General I/O]
watch P00, P01, P02, P10, P11, P20  ← a
5.0 K00=1                               ← b
5.5 K00=0                               The first value indicates the time in seconds from a reset.
8.0 K00=H                                1 / H = High, 0 / L = Low
8.5 K00=L

[A/D Converter]
AVdd=3.0                                 ← c
AVss=0.2                                 The values indicate voltages (V).
AVref=2.9

[A/D CH2]
2.40, 2.39, 2.38, 2.37, 2.36             ← d
2.00, 2.10, 2.20, 2.30                   The values indicate voltages (V).
0.5
0.65
0.80

[Serial CH1]
"aBcDeFg"                               ← e
0x25, 0x20, 0x41                         Specify the input data using a character string
                                           or hexadecimal value.
```

Load this file to simulate the status of each input pin. For detailed information on this file, refer to Section 4.13, "IOT File (.iot)".

(3) Starting and finishing input/output simulation

Follow the procedure given below to start input/output simulation:

1. Display the [I/O Terminal] window.
2. Load the IOT file by selecting [Open] from the [File] menu.
3. Run the target program from a reset state.

Although the IOT file can be loaded while running the target program, the simulator must be reset before input/output status can be simulated.

To terminate input/output simulation, select [Close] from the [File] menu.

(4) Simulation and log display

When the specified input/output ports change state (from 1 to 0 or vice versa), or the serial interface receives input data or sends output data, or the A/D converter receives analog signals as input, the pertinent information is displayed in the window.

Example:

```

P00=1 R01=1
AVdd=3.000 AVss=0.200 AVref=2.900
00:00:01.029 S2 << 31h(1)
00:00:02.033 S2 << 32h(2)
00:00:03.036 S1 >> 33h(3)
00:00:04.040 S1 >> 34h(4)
00:00:05.000 P30 << 0 P31 << 0
00:00:06.016 AD0 << 2.400
00:00:07.020 AD0 << 2.390
00:00:08.023 AD0 << 2.380
00:00:09.027 AD0 << 2.370
00:00:05.000 R01 >> 1
    
```

Initial data

Log data

Immediately after a reset, the simulator displays the initial data. The contents displayed here are the initial values of the general-purpose input/output ports specified with a "Watch" statement in the IOT file's [General I/O] section and the "AVdd", "AVss", and "AVref" values specified in the [A/D Converter] section. These values are not displayed unless they are specified.

Displayed following the above is data input or output during the simulation. The numeric value appearing at the beginning of each line indicates the elapsed time (in hours, minutes, and seconds) since the simulator was reset. The symbols "<<" and ">>" denote input and output, respectively. The data for each port and the contents of processing are described below.

General-purpose input/output ports:

When input or output on the ports specified with a Watch statement changes state, the simulator displays the port names (Kxx, Pxx, Rxx) and the value after the change (0 or 1). Changes in input occur synchronously with the timing written in the IOT file. If interrupts are enabled at this time, an input interrupt is generated according to the interrupt conditions set.

A/D converter:

Each time the target program performs an A/D conversion, the data written in the IOT file is loaded in sequence from the top (left) to simulate the A/D conversions performed. At this time, the simulator displays the channel numbers (ADx) and the input voltages (V) loaded from the file. If interrupts are enabled, an A/D conversion interrupt is generated after a certain period of time has elapsed (sampling time + A/D conversion time) after the start of input. If a converted channel has no specified data, the input on that channel is read out as level 0 voltage (equivalent to AVss). If A/D conversion is performed more than a specified data count, the last data specified is read out repeatedly. If a value smaller than AVss was specified, it is read out as level 0 (equivalent to AVss); if a value larger than AVref was specified, it is read out as the maximum level (equivalent to AVref).

Serial interface:

As the target program performs input/output via the serial interface, the simulator displays the channel numbers (S1, S2) and the input/output values (hexadecimal values and ASCII characters) for each byte transferred. For input, the input data written in the IOT file is read out sequentially from the top (left). If interrupts are enabled, a serial interface interrupt is generated synchronously with the timing at which the program finishes sampling the input.

(5) Saving logs

The logs displayed in the [I/O Terminal] window can be saved to text files using [Save] from the [File] menu.

Furthermore, a range of logs selected by dragging the mouse in the [I/O Terminal] window can be copied to the clipboard using [Copy] from the [Edit] menu. The copied log can be pasted from the clipboard into any document by using an editor. To copy all logs, use [Select All] from the [Edit] menu to select all logs before copying to the clipboard.

Precaution

- In development environments where the S1C63/88 simulator is already installed, the components corresponding to the serial interface must be added to the component map file (.cmp). When using a model that does not support the serial interface, add NullDev.bmc to the component map file. When using the A/D converter, make any necessary corrections to the component map file.

Example: 88348.cmp [Internal] CPU=CPU.bmc LCD=LcdDrv88.bmc K/P/R port=KPRport.bmc SVD=SVD88.bmc Sound=Sound88.bmc Serial=Serial88.bmc ← Add Adc=NULLDev.bmc ← Add	Example: 88349.cmp [Internal] CPU=CPU.bmc LCD=LcdDrv88.bmc K/P/R port=KPRport.bmc SVD=SVD88.bmc Sound=Sound88.bmc Serial=Serial88.bmc ← Add Adc=Adc88.bmc ← Add
---	---

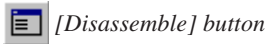
- The time at which the simulator simulates input/output operations is calculated from its instruction execution cycles. This time differs slightly from the actual MCU (actual time).

4.6.5 [Source] Window

The [Source] window displays the program code. The following three display modes are supported:

1. Disassemble display mode

After disassembling the loaded object, the simulator displays the addresses, codes, and mnemonics in it. To open the [Source] window in this mode, select [Source | Disassemble] from the [View] menu. To go to disassemble display mode while in another mode, select [Source | Disassemble] as described above, or click the [Disassemble] button on the [Source] window, or run the u command. When the [Source] window is in this display mode, the word "Disassemble" is displayed on the title bar. This display mode can be selected regardless of the type of object file loaded.



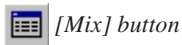
2. Source display mode

In this mode, the simulator displays the corresponding source for an object that includes the current program counter address. However, this mode can be selected only when an absolute object file (.abs) in IEEE-695 format containing debug information for source display purpose is loaded. To open the [Source] window in this mode, select [Source | Source] from the [View] menu. To go to source display mode while in another mode, select [Source | Source] as described above, or click the [Source] button on the [Source] window, or run the sc command. When an absolute object file (.abs) that contains C source debug information is loaded while the [Source] window is open, the [Source] window automatically enters this mode. In this display mode, the source file name is displayed on the title bar.



3. Mix display mode

In this mode, the simulator displays the source and its disassembled contents (address, code, and mnemonic) separately in the upper and lower rows. However, this mode can be selected only when an absolute object file (.abs) in IEEE-695 format containing debug information for source display purpose is loaded. To open the [Source] window in this mode, select [Source | Mix] from the [View] menu. To go to mix display mode while in another mode, select [Source | Mix] as described above, or click the [Mix] button on the [Source] window, or run the m command. When the [Source] window is in this display mode, the word "Mix" is displayed on the title bar.

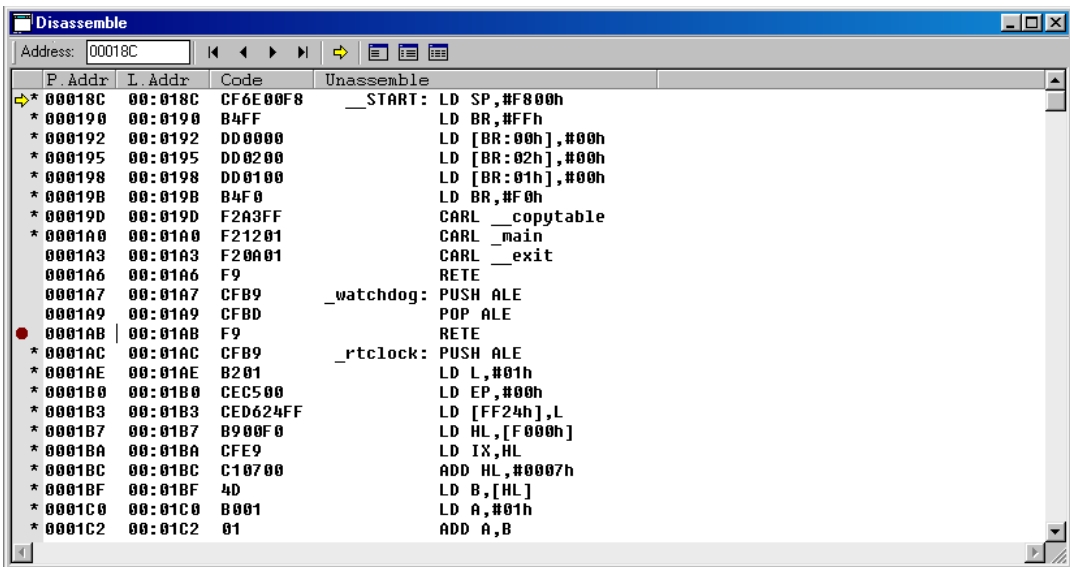


* Source display

The source of any object can be displayed only when an absolute object file in IEEE-695 format that contains debug information for source display purpose is loaded.

Furthermore, because the source file is loaded after locating it from the object file's debug information (relative path information for the source file), if the source file is removed or relocated (i.e., its relative position from the object file has changed), the source is not displayed. In this case, the window in source display mode is left blank, and in mix display mode, the window shows only the disassembled contents.

Disassemble display mode



Described below are the functions of the [Source] window in disassemble display mode:


(1) Displaying program code

The window displays the physical/logical addresses, codes, and disassembled contents.

Program display location can be changed by the following method as well as scrolling.

- Enter an address in the [Address] text box. Or specify an address using the u command.

The program is displayed from the selected address.

- 
 - Displays the beginning or end area of the memory.
 - Displays one page before or after in the current window size.
 - Displays the program from the current PC address.

Note: The S1C88 Family processors use variable length mnemonics, so that when the window is scrolled upward, the disassembled contents shown on the window may differ from the actual code.

* Updating of display

When a program is loaded and executed (g, gr, s, n, se, or rst command), or the memory contents are changed (de, df, or dm command), the display contents are updated. In this case the [Disassemble] window updates its display contents so that the current PC address can always be displayed.

(2) Displaying the current PC

The current PC (program counter) address is indicated by a yellow arrow at the beginning of the line.


(3) Displaying PC breakpoints


The address line where a breakpoint is set is indicated by a red ● mark at the beginning of the line.

(4) Coverage information

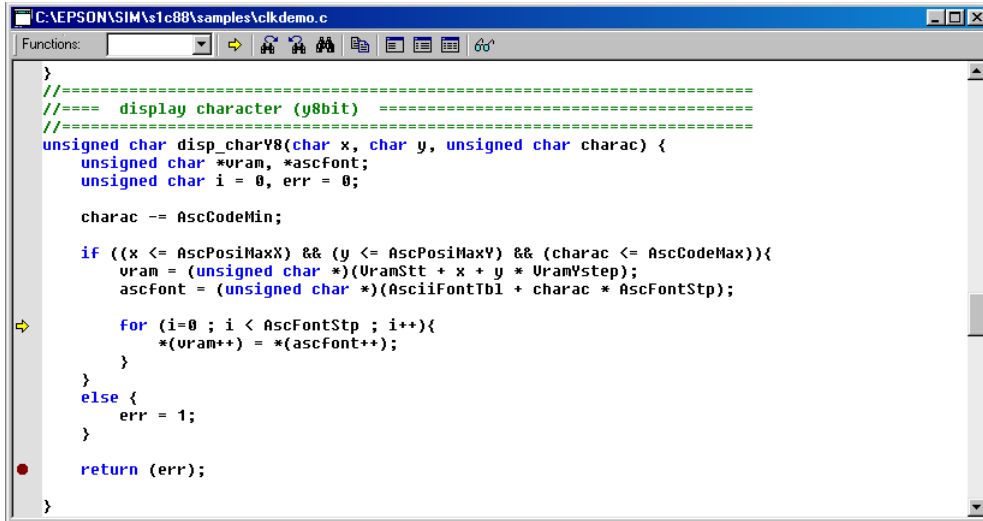
When the coverage function is turned on using the md command, the following program execution places an * at the beginning of the executed address line.

(5) Setting a break at the cursor position

 Place the cursor at an address line where a breakpoint is to be set. Then click on the [Break] button. A PC breakpoint will be set at that address. If the same is done at the address line where a PC breakpoint has been set, the breakpoint will be cleared. This function allows setting of two or more breakpoints.

 If the [Go to Cursor] button is clicked, the program will execute beginning with the current PC position, and program execution breaks at the line where the cursor is located.

Source display mode



Described below are the functions of the [Source] window in source display mode:

(1) Displaying program code

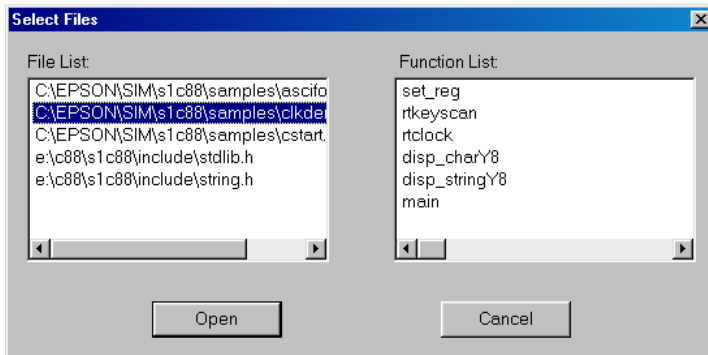
The window displays the source of the loaded object. The source automatically displayed here includes the address indicated by the current PC (program counter). The comment lines, reserved words, and any text other than these two types are displayed in green, blue, and black, respectively. The tab width is set to a length of four characters. The program display position can be changed in the following manner, as well as by scrolling:



- Select a function name from the [Functions] pull-down list. The source is displayed from the beginning of that function.

- Click the [Current PC] button. The source is displayed from the current PC address.

- To display another source file, click the [Source Files] button to bring up the dialog box shown below and to select the desired source file from the list of sources.



* Updating of display

When a program is loaded and executed (g, gr, s, n, se, or rst command) and program execution is halted midway, the display contents are updated. In this case, the source that includes the current PC address is displayed in the window. If the corresponding source cannot be found, the [Select Files] dialog box shown above appears, prompting for selection of the source to be displayed.

(2) Displaying the current PC

The source lines that include the address indicated by the current PC (program counter) are marked with a yellow arrow at the beginning of the line.

(3) Displaying PC breakpoints

The source lines that include any address that has been set as a breakpoint are marked with a red ● mark at the beginning of the line.

(4) Setting a break at the cursor position



Place the cursor at the source line at which a breakpoint is to be set. Then click the [Break] button. This sets the source line (the start address of the effective object code corresponding to the source) as a breakpoint. (A breakpoint can also be set by double-clicking anywhere in the line.) If the same action is performed at the source line in which a PC breakpoint has been set, the breakpoint is cleared. Multiple breakpoints can be set, one breakpoint per source line. However, no breakpoints can be set in source lines that do not have actual code. Note that due to optimization by the C compiler, no code can be generated for some C statements that would otherwise have code generated. For source lines at which breakpoints cannot be set, change to mix display mode and check.



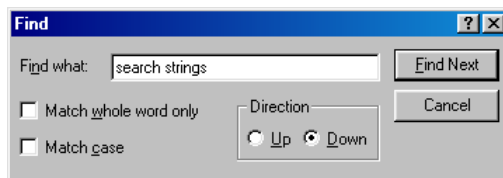
Click the [Go to Cursor] button. The program starts running from the current PC and breaks at the line at which the cursor is positioned. In this case, the cursor must also be located at the source lines that have the actual code. Clicking the [Go to Cursor] button has no effect unless the source has the actual code.

(5) Searching for a character string

In source display mode, the [Source] window displays the following find buttons, permitting a search for a character string.



Click the [Find] button to display the dialog box shown below, allowing you to specify a search string.



Enter a search string in the [Find what] edit box and click the [Find Next] button. The string search proceeds in the downward direction of the [Source] window (toward the end of the program) from the current cursor position. If an instance of the specified string is found in the [Source] window, it is placed in a selected state.

When the [Find Next] button is clicked again, the next instance of the specified string is sought from that position forward. To search up (toward the beginning of the program), select the [Up] button for [Direction]. To search for instances that completely match the specified string, check the [Match whole word only] check box. Or to discriminate between uppercase and lowercase letters when searching, check the [Match case] check box, before clicking the [Find Next] button.



Select a string by dragging the mouse in the [Source] window and clicking the [Find Next] button on the [Source] window. The string search proceeds in the downward direction of the [Source] window (toward the end of the program) from that selected position. If an instance of the string is found, the newly found string is placed in a selected state. When the [Find Next] button is clicked again, the next instance of the string is sought from that position forward. This search is case-insensitive, and instances that do not completely match the string will also be found.



The [Find Previous] button functions in the same way as the [Find Next] button described above, except that string searches proceed up (toward the beginning of the program).

(6) Registering symbols in the [Watch] window



Select a symbol name in the window by dragging with the mouse (displayed in reverse video when selected) and click the [Watch] button. The symbol is registered to the symbol list of the [Watch] window. Once registered this way, the value of that symbol can be verified in the [Watch] window.

(7) Displaying variable values

```

C:\EPSON\SIM\1c88\samples\clkdemo.c
Functions:
set_reg();
//<display clock data>
err = disp_stringV8(0,0,
} while(err = 0x00
// free((char*)p_clkdata);
}

```

Place the mouse cursor at a variable name in the displayed source (need not to click), and the value of that variable (or address for a pointer variable) is displayed. The variable types (signed/unsigned) int, long, and short are displayed in decimal notation, while addresses, structures, and unions are displayed in hexadecimal notation. To display the values of structure members, the member's variable name needs to be selected with the mouse. For array elements, variable names must be selected with the mouse. Out-of-scope variables are not displayed.

Mix display mode

```

Mix
Address: 0003B5
//==== display string (y8bit) =====
//-----
unsigned char disp_stringV8(char x, char y, unsigned char *string) {
* 0003B5 00:03B5 CF6A0400 _disp_stringV8: SUB SP,#0004h
* 0003B9 00:03B9 48 LD B,a
* 0003BA 00:03BA CEC600 LD XP,#00h
* 0003BD 00:03BD CFFA LD IX,SP
* 0003BF 00:03BF CE5402 LD [IX+02h],L
    unsigned char err = 0;
* 0003C2 00:03C2 B200 LD L,#00h
    while ((*string != NULL) || err !=0) {
* 0003C4 00:03C4 F133 JRS 33h
    err = disp_charV8(x,y,*string);
* 0003C6 00:03C6 CF7700 LD [SP+00h],IY
* 0003C9 00:03C9 CEC700 LD YP,#00h
* 0003CC 00:03CC 5F LD H,[IY]
* 0003CD 00:03CD CEC600 LD XP,#00h
* 0003D0 00:03D0 CFFA LD IX,SP
* 0003D2 00:03D2 CE4C03 LD [IX+03h],B
* 0003D5 00:03D5 CEC700 LD YP,#00h
* 0003D8 00:03D8 CFFE LD IY,SP
* 0003DA 00:03DA CE5102 LD L,[IY+02h]

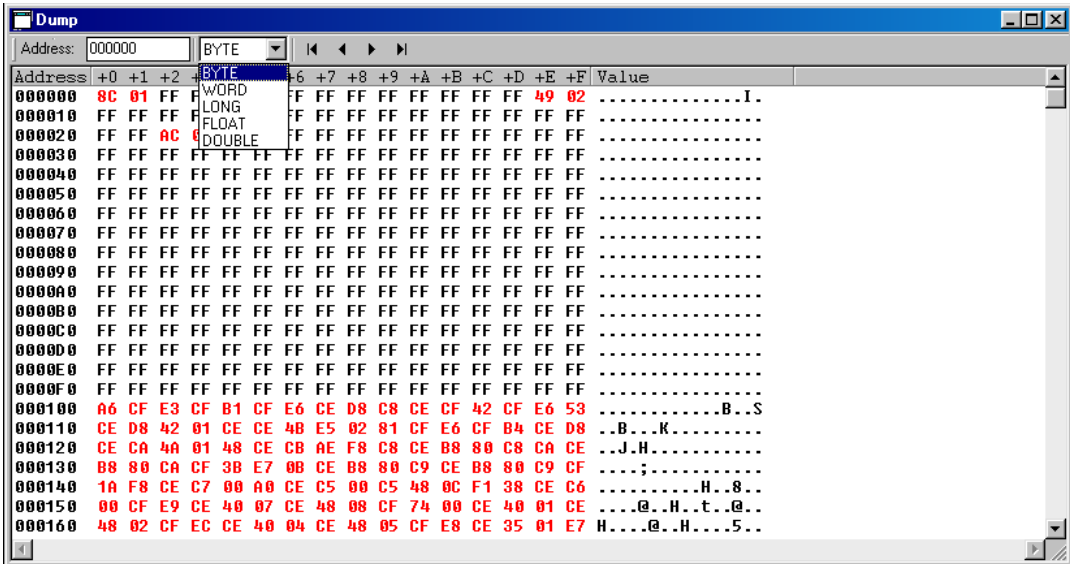
```

The mix display mode is functionally the same as disassemble display mode. The difference is that each source line and the disassembled contents of the corresponding object code (physical/logical address, object code, and mnemonic) are displayed one for one in the upper and lower rows. However, mix display mode can only be selected when an absolute object file (.abs) in IEEE-695 format containing debug information is loaded.

The displayed source lines cannot be operated on - for example, by setting a break. Various display manipulating and break setting operations can only be performed on the disassembled display contents. For [Source] window functions that can be used in mix display mode, refer to the description of disassemble display mode.

The source lines and the disassembled contents are displayed in black and gray, respectively.

4.6.6 [Dump] Window



(1) Displaying data memory contents


The [Dump] window displays the memory dump results in hexadecimal numbers.

Data is displayed in byte units by default. It can be changed to another size using the pull-down box.

Memory display location can be changed by the following method as well as scrolling.

- Enter an address in the [Address] text box. Or specify an address using the dd command.

Data is displayed from the selected address.

- 
 - Displays the beginning or end area of the memory.
 - Displays one page before or after in the current window size.

* Updating of display

The display contents of the [Dump] window are updated automatically when memory contents are modified with a command (de, df, or dm command), or by direct modification. After executing the program (g, gr, s, n, se, or rst command), the display contents are also updated. To refresh the [Dump] window manually, execute the dd command or click the vertical scroll bar.

After program execution is completed, the value changed during execution is displayed in red.

(2) Direct modification of data memory contents

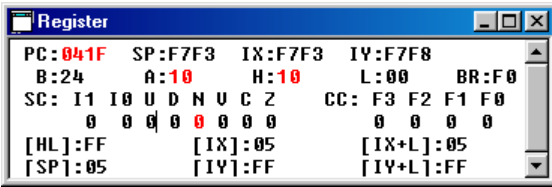
The [Dump] window allows direct modification of data memory contents. To modify data on the [Dump] window, place the cursor at the front of the data to be modified or double click the data, and then type a hexadecimal character (0–9, a–f). Data in the address will be modified with the entered number and the cursor will move to the next address. This allows successive modification of a series of addresses.

(3) Displaying decimal data

Hover the mouse cursor over data (need not to click) during [BYTE], [WORD], or [LONG] display, and the data is displayed in decimal notation (signed int or unsigned int). For [BYTE], the data is also displayed in bits.

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8
000100	CE	BD	00	E6	08	CE	80	CE	91
000110	CF	B1	CF	E6	CE	D8	C8	CE	CF
000120	01	CE	CE	4B	[BIT1001110]		CF	E6	
000130	01	48	CE	CB	[int -50]		C7	00	
000140	F1	38	CE	C8	[uint 206]		CE	40	

4.6.7 [Register] Window



(1) Displaying register contents

The [Register] window displays the contents of the S1C88 CPU registers, condition flags and the memory pointed by the [HL], [SP], [IX], [IY], [IX+L] and [IY+L] registers.

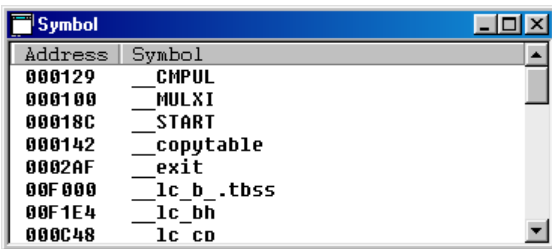
* Updating the display

The display is updated when registers are dumped (rd command), when register data is modified (rs command), when the CPU is reset (rst command), or after program execution (g, gr, s, se, or n command) is completed. After program execution is completed, the value changed during execution is displayed in red.

(2) Direct modification of register contents

The [Register] window allows direct modification of register contents. To modify data on the [Register] window, select (highlight) the data to be modified and type a hexadecimal number (0-9, a-f), then press [Enter]. The register data will be modified with the entered number.

4.6.8 [Symbol] Window

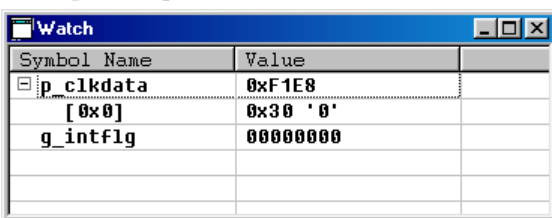


The [Symbol] window can display the symbol list, if a symbol file (.sy) is loaded.

Symbols are listed in alphabetical order by default. It can be changed to address order using the "sy /a" command.

- * The symbol file is automatically read when a target program (Motorola S2 format) is loaded. However, it must be the same name (extension is .sy) and be located in the same directory as the target program file. Note that a symbol file is not read when an IEEE-695 program file is loaded.

4.6.9 [Watch] Window



The window shows the name and the current value of the symbol registered using the w command or the [Source] window [Watch] button. The value is displayed in the format specified by the w command. If the symbol is an array, structure, or union, a icon is displayed. Clicking this icon displays the array, structure, or union members hierarchically.

The registered symbols can also be removed or have their display formats changed (e.g., from hexadecimal to decimal) from a menu displayed by right-clicking the symbol. However, display formats can be changed only for types such as int, char, long, and short, and cannot be changed for addresses. The addresses are always displayed in hexadecimal notation.

Note that symbol display on this window is possible only when an absolute object file (.abs) in IEEE-695 format containing information on the specified symbol is loaded.

Note: If the -O1 option is specified when compiling, unnecessary symbols may be removed for code optimization, and no symbol information may be generated. Such symbols cannot be registered in the [Watch] window.

* Updating the display

The display is updated after program execution (g, gr, s, se, or n command) is completed.

4.6.10 [Trace] Window

Trace																
INS	P Addr	L Addr	Code	Mnemonic	BA	HL	IX	IV	SP	BR	EP	XP	YP	SC	CC	Memory
0217	000499	01:0499	93	INC IV	3E84	F828	F828	F060	F7F3	F0	00	00	00	00	--N-C-	0000
0218	00049A	01:049A	92	INC IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	--N-C-	0000
0219	00049B	01:049B	CF7601	LD [SP+01h],IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	--N-C-	0000 MW:[00F7F4]=29 HW:[00F7
0220	00049E	01:049E	B001	LD A,#01h	3E01	F828	F829	F060	F7F3	F0	00	00	00	00	--N-C-	0000
0221	0004A0	01:04A0	A6	PUSH IP	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	--N-C-	0000 MW:[00F7F2]=00 HW:[00F7
0222	0004A1	01:04A1	CEC600	LD XP,#00h	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	--N-C-	0000
0223	0004A4	01:04A4	CFFA	LD IX,SP	3E01	F828	F7F1	F060	F7F1	F0	00	00	00	00	--N-C-	0000
0224	0004A6	01:04A6	CE4802	LD B,[IX+02h]	0401	F828	F7F1	F060	F7F1	F0	00	00	00	00	--N-C-	0000 MR:[00F7F3]=04
0225	0004A9	01:04A9	AE	POP IP	0401	F828	F7F1	F060	F7F3	F0	00	00	00	00	--N-C-	0000 MR:[00F7F1]=00 HR:[00F7
0226	0004AA	01:04AA	01	ADD A,B	0405	F828	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0227	0004AB	01:04AB	50	LD L,A	0405	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0228	0004AC	01:04AC	B105	LD B,#05h	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0229	0004AE	01:04AE	42	LD A,L	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0230	0004AF	01:04AF	A6	PUSH IP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F2]=00 HW:[00F7
0231	0004B0	01:04B0	CEC600	LD XP,#00h	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0232	0004B3	01:04B3	CFFA	LD IX,SP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0233	0004B5	01:04B5	CE4402	LD [IX+02h],A	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F3]=05
0234	0004B8	01:04B8	AE	POP IP	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000 MR:[00F7F1]=00 HR:[00F7
0235	0004B9	01:04B9	3A80	XOR A,#80h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	--N----	0000
0236	0004BB	01:04BB	CEB880	XOR B,#80h	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	--N----	0000
0237	0004BE	01:04BE	31	CP A,B	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0238	0004BF	01:04BF	CEE0CB	JRS LT,CBh	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0239	0004C2	01:04C2	F105	JRS 05h	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000

After the trace function is turned on by the md command, the simulator samples trace information while the target program is running. The trace data buffer has a capacity for 8192 instructions (overwritten from the beginning if the capacity is exceeded), and its data can be displayed in the [Trace] window.

The following lists the trace contents:

- Instruction number
- Fetched code and disassembled contents
- Register and condition flag contents
- Memory access status (R/W, address, data)

This window also displays the trace data search results by the ts command.

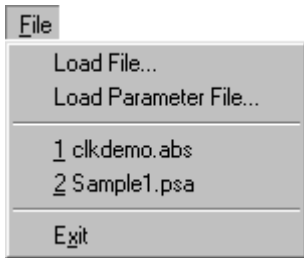
* Updating of display:

The contents of the [Trace] window are cleared when the target program is executed. After the execution has finished, the [Trace] window displays the contents of the trace buffer.

4.7 Menu

This section outlines the menu bar available with the simulator. The menu bar has eight menus, each including frequently-used commands.

[File] Menu



The file names listed in this menu are recently used files. Selecting one opens the file.

[Load File...]

This button reads an object file in the IEEE-695 format or an internal ROM HEX file in Motorola S2 format into the simulator. It performs the same function when the lf command is executed.

[Load Parameter File...]

This button reads a parameter file into the simulator. It performs the same function when the par command is executed.

[Exit]

This menu item quits the simulator. It performs the same function when the q command is executed.

[Run] Menu



[Go]

This menu item executes the target program from the address indicated by the current PC. It performs the same function when the g command is executed.

[Go to Cursor]

This menu item executes the target program from the address indicated by the current PC to the cursor position in the [Source] window (the address of that line). It performs the same function when the g <address> command is executed. Before this menu item can be selected, the [Source] window must be open and the address line where the program is to break must be clicked.

[Go after Reset]

This button resets the CPU and then executes the target program after fetching the reset vector. It performs the same function when the gr command is executed.

[Step]

This menu item executes one instruction step at the address indicated by the current PC. It performs the same function when the s command is executed.

[Next]

This button executes one instruction step at the address indicated by the current PC. If the instruction to be executed is cars, carl, call, or int, it is assumed that a program section until control returns to the next address constitutes one step and all steps of their subroutines are executed. This button performs the same function when the n command is executed.

[Step Exit]

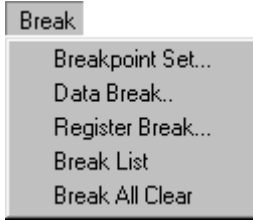
This button executes the target program from the address indicated by the current PC. If the program starts from inside a subroutine, the program execution will stop when the sequence returns to the parent routine. This button performs the same function when the se command is executed.

[Reset CPU]

This menu item resets the CPU. It performs the same function when the rst command is executed.

[Command File...]

This menu item reads a command file and executes the debug commands written in that file. It performs the same function when the com or cmw command is executed.

[Break] Menu**[Breakpoint Set...]**

This menu item displays, sets or clears PC breakpoints using a dialog box. It performs the same function as executing the bp command.

[Data Break...]

This menu item displays, sets or clears data break conditions using a dialog box. It performs the same function as executing the bd command.

[Register Break...]

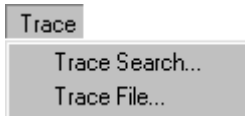
This menu item displays, sets or clears register break conditions using a dialog box. It performs the same function as executing the br command.

[Break List]

This menu item displays the all break conditions that have been set. It performs the same function as executing the bl command.

[Break All Clear]

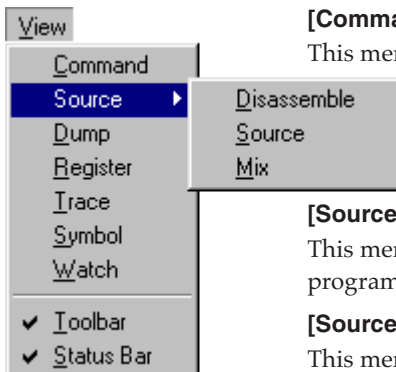
This menu item clears all break conditions. It performs the same function as executing the bac command.

[Trace] Menu**[Trace Search...]**

This menu item searches trace information from the trace data buffer under the condition specified using a dialog box. It performs the same function as executing the ts command.

[Trace File...]

This menu item saves the specified range of the trace information displayed in the [Trace] window to a file. It performs the same function as executing the tf command.

[View] Menu**[Command]**

This menu item activates the [Command] window.

[Source - Disassemble]

This menu item opens or activates the [Source] window and displays the program in the disassemble display mode.

[Source - Source]

This menu item opens or activates the [Source] window and displays the program in the source display mode.

[Source - Mix]

This menu item opens or activates the [Source] window and displays the program in the mix display mode.

[Dump]

This menu item opens or activates the [Dump] window and displays the memory contents from the memory start address.

[Register]

This menu item opens or activates the [Register] window and displays the current values of the registers.

[Trace]

This menu item opens or activates the [Trace] window and displays the trace data sampled in the trace data buffer.



[Symbol]

This menu item opens or activates the [Symbol] window and displays the symbol list if a symbol file has been loaded.

[Watch]

This menu item opens or activates the [Watch] window and displays the contents of the symbol registered.

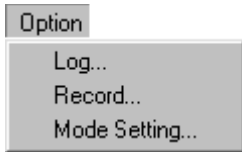
[Toolbar]

This menu item shows or hides the toolbar.

[Status Bar]

This menu item shows or hides the status bar.

[Option] Menu



[Log...]

This menu item starts or stops logging using a dialog box. It performs the same function as executing the log command.

[Record...]

This menu item starts or stops recording of a command execution using a dialog box. It performs the same function as executing the rec command.

[Mode Setting...]

This menu item sets the trace function, coverage function, step-display mode, and execution interval time for the cmw command. It performs the same functions as executing the md command.

[Window] Menu



[Cascade]

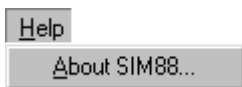
This menu item cascades the opened windows.

[Tile]

This menu item tiles the opened windows.

This menu shows the currently opened window names. Selecting one activates the window.

[Help] Menu



[About SIM88...]

This menu item displays an About dialog box for the simulator.

4.8 Tool Bar

This section outlines the tool bar available with the simulator.

The tool bar has 12 buttons, each one assigned to a frequently used command.



The specified function is executed when you click on the corresponding button.



[Load File] button

This button reads an absolute object file in IEEE-695 format, a program file in Motorola S2 format, or a function option file into the simulator. It performs the same function when the If command is executed.



[Load Parameter] button

This button reads a parameter file into the simulator. It performs the same function when the par command is executed.



[Key Break] button

This button forcibly breaks execution of the target program. This function can be used to cause the program to break when the program has fallen into an endless loop.



[Break] button

Use this button to set and clear a breakpoint at the address where the cursor is located in the [Source] window. This function is valid only when the [Source] window is open.



[Break All Clear] button

This button clears all break conditions. It performs the same function as executing the bac command.



[Go] button

This button executes the target program from the address indicated by the current PC. It performs the same function when the g command is executed.



[Go to Cursor] button

This button executes the target program from the address indicated by the current PC to the cursor position in the [Source] window (the address of that line). It performs the same function when the g <address> command is executed.

Before this button can be selected, the [Source] window must be open and the address line where the program is to break must be clicked.



[Go after Reset] button

This button resets the CPU and then executes the target program after fetching the reset vector. It performs the same function when the gr command is executed.



[Step] button

This button executes one instruction step at the address indicated by the current PC. It performs the same function when the s command is executed.



[Next] button

This button executes one instruction step at the address indicated by the current PC. If the instruction to be executed is cars, carl, call, or int, it is assumed that a program section until control returns to the next address constitutes one step and all steps of their subroutines are executed. This button performs the same function when the n command is executed.



[Step Exit] button

This button executes the target program from the address indicated by the current PC. If the program starts from inside a subroutine, the program execution will stop when the sequence returns to the parent routine. This button performs the same function when the se command is executed.



[Reset CPU] button

This button resets the CPU. It performs the same function when the rst command is executed.

4.9 Method for Executing Commands

All debug functions can be performed by executing debug commands. This section describes how to execute these commands.

4.9.1 Entering Commands from Keyboard

Select the [Command] window (by clicking somewhere on the [Command] window). When the prompt ">" appears on the last line in this window and a cursor is blinking behind it, the system is ready to accept a command from the keyboard.

Input a debug command at the prompt position. The commands are not case-sensitive; they can be input in either uppercase or lowercase.

General command input format

```
>command [ parameter [ parameter ... parameter ] ] ⌘
```

- A space is required between a command and parameter.
- Space is required between parameters.

Use the arrow keys, [Back Space] key, or [Delete] key to correct erroneous input.

When you press the [Enter] key after entering a command, the system executes that command. (If the command entered is accompanied by guidance, the command is executed when the necessary data is input according to the displayed guidance.)

Input example:

```
>g⌘           (Only a command is input.)
>com test.cmd⌘ (A command and parameter are input.)
```

Command input accompanied by guidance

For commands that cannot be executed unless a parameter or the commands that modify the existing data are specified, a guidance mode is entered when only a command is input. In this mode, the system brings up a guidance field, so input a parameter there.

Input example:

```
>lf⌘
File name   ? :test.abs⌘ ← Input data according to the guidance (underlined part).
>
```

• Commands requiring parameter input as a precondition

The If command shown in the above example reads a program file into the simulator. Commands like this that require an entered parameter as a precondition are not executed until the parameter is input and the [Enter] key pressed. If a command has multiple parameters to be input, the system brings up the next guidance, so be sure to input all necessary parameters sequentially. If the [Enter] key is pressed without entering a parameter in some guidance session of a command, the system assumes the command is canceled and does not execute it.

• Commands that replace existing data after confirmation

The commands that rewrite memory or register contents one by one provide the option of skipping guidance (do not modify the contents), returning to the immediately preceding guidance, or terminating during the input session.

```
[Enter] key    Skips input.
[^] key       Returns to the immediately preceding guidance.
[q] key       Terminates the input session.
```

Input example:

```
>de $\mathcal{N}$             $\Leftarrow$  Command to modify data memory.
Data enter address ? :001000 $\mathcal{N}$   $\Leftarrow$  Inputs the start address.
001000 A:1 $\mathcal{N}$         $\Leftarrow$  Modifies address 001000H to 1.
001001 A:^ $\mathcal{N}$        $\Leftarrow$  Returns to the immediately preceding address.
001000 1:0 $\mathcal{N}$        $\Leftarrow$  Inputs address 001000H back again.
001001 A: $\mathcal{N}$ 
001002 A: $\mathcal{N}$ 
001001 A:q $\mathcal{N}$        $\Leftarrow$  Terminates the input session.
>
```

Numeric data format of parameter

For numeric values to be accepted as a parameter, they must be input in hexadecimal numbers for almost all commands. However, some parameters accept decimal or binary numbers.

The following characters are valid for specifying numeric data:

Hexadecimal: 0-9, a-f, A-F, *

Decimal: 0-9

Binary: 0, 1, *

("*" is used to mask bits when specifying a data pattern.)

Specification with a symbol

For address specifications, the symbols can also be used when an IEEE-695 absolute object file (.abs) or a symbol file (.sy) is loaded.

Input example:

```
>u Main $\mathcal{N}$           $\Leftarrow$  Displays the program from the label Main
```

- * The symbol file (.sy) is automatically loaded simultaneously with the target program in the Motorola S2 format. However, it must be the same name (extension is .sy) and be located in the same directory as the target program file. When an IEEE-695 program file is specified, the simulator does not load a symbol file.

Notes:

- If the specified symbol is not found, sim88 handles the specified string as a hexadecimal (e.g., ABC). However, if the string includes other than the specified hexadecimal characters, an error is assumed.

- If the -O1 option is specified when compiling the C source, some symbols written in the source may not actually be used for reasons of code optimization. In such cases, debug information for that symbol is not output to the .abs file, whether or not the -g option is specified.

```
Example: int x,y,xy;
         x = GLOBAL_X * 100;
         y = GLOBAL_Y * 100;
         xy = x * y;
```

In this example, because variable xy become nonexistent due to optimization, the contents of xy cannot be referenced when debugging.

If after evaluating the executable file created by specifying the -O0 option (optimization OFF), it is recreated by specifying the -O1 option (optimization ON), program behavior cannot be guaranteed. Be sure to reverify the executable file whenever it is recreated in this way.

Successive execution using the [Enter] key

The commands listed below can be executed successively by using only the [Enter] key after executing once. Successive execution here means repeating the previous operation or continuous display of the previous contents.

Execution commands: g, s, n, se, com
















Display commands: u, dd, td

The successive execution function is terminated when some other command is executed.

4.9.2 Executing from Menu or Tool Bar

The menu and tool bar are assigned frequently-used commands as described in Sections 4.7 and 4.8. A command can be executed simply by selecting desired menu command or clicking on the tool bar button. Table 4.9.2.1 lists the commands assigned to the menu and tool bar.

Table 4.9.2.1 Commands that can be specified from menu or tool bar

Command	Function	Menu	Button
lf	Load program file	[File Load File...]	
par	Load parameter file	[File Load Parameter File...]	
g	Execute program successively	[Run Go]	
g <address>	Execute program to <address> successively	[Run Go to Cursor]	
gr	Reset CPU and execute program successively	[Run Go after Reset]	
s	Single step execution	[Run Step]	
n	Step execution with skip subroutine	[Run Next]	
se	Exit from subroutine	[Run Step Exit]	
com	Load and execute command file	[Run Command File...]	–
cmw	Load and execute command file with wait	[Run Command File...]	–
rst	Reset CPU	[Run Reset CPU]	
bp, bc (bpc)	Set/clear PC breakpoint	[Break Breakpoint Set...]	
bd, bdc	Set/clear data break	[Break Data Break...]	–
br, brc	Set/clear register break	[Break Register Break...]	–
bl	Break list	[Break Break List]	–
bac	Clear all break conditions	[Break Break All Clear]	
ts	Search trace information	[Trace Trace Search...]	–
tf	Save trace information to file	[Trace Trace File...]	–
u	Disassemble display	[View Source Disassemble]	 *
sc	Source display	[View Source Source]	 *
m	Mix display	[View Source Mix]	 *
dd	Dump memory	[View Dump]	–
rd	Display register values	[View Register]	–
td	Display trace information	[View Trace]	–
sy	Display symbol list	[View Symbol]	–
w	Display symbol information	[View Watch]	–
	Register symbols	–	 *
log	Turn log output on or off	[Option Log...]	–
rec	Record commands to a command file	[Option Record...]	–
md	Set modes	[Option Mode Setting...]	–

* Located in the [Source] window

4.9.3 Executing from a Command File

Another method for executing commands is to use a command file that contains descriptions of a series of debug commands. By reading a command file into the simulator the commands written in it can be executed.

Creating a command file

Create a command file as a text file using an editor.

Although there are no specific restrictions on the extension of a file name, Seiko Epson recommends using ".cmd".

Command files can also be created using the rec command. The rec command creates a command file and saves the executed commands to the file.

Example of a command file

The example below shows a command group that loads a program file, sets a breakpoint and then executes the program.

Example: File name = start.cmd

```
lf test.psa
bp 0004d7
g
```

A command file to write the commands that come with a guidance mode can be executed. In this case, be sure to break the line for each guidance input item as a command is written.

Reading in and executing a command file

The simulator has the com and cmw commands available that can be used to execute a command file. The com command reads in a specified file and executes the commands in that file sequentially in the order they are written.

The cmw command performs the same function as the com command except that each command is executed at intervals specified by the md command (1 to 256 seconds).

Example: com start.cmd

```
cmw test.cmd
```

The commands written in the command file are displayed in the [Command] window.

Restrictions

Another command file can be read from within a command file. However, nesting of these command files is limited to a maximum of five levels. An error is assumed and the subsequent execution is halted when the com or cmw command at the sixth level is encountered.

4.9.4 Log File

The executed commands and the execution results can be saved to a file in text format that is called a "log file". This file allows verification of the debug procedures and contents. The contents displayed in the [Command] window are saved to this file.

Command example

```
>log tst.log
```

After the simulator is set to the log mode by the log command (after it starts outputting to a log file), the log command toggles (output turned on in log mode \cap output turned off in normal mode). Therefore, you can output only the portions needed can be output to the log file.

Display of [Command] window in log mode

The contents displayed in the [Command] window during log mode differ from those appearing in normal mode.

- (1) When executing a command when each window is open
(When the window that displays the command execution result is opened)
Normal mode: The contents of the relevant display window are updated. The execution results are not displayed in the [Command] window.
Log mode: The same contents as those displayed in the relevant window are also displayed in the [Command] window. However, changes made to the relevant window by scrolling or opening it are not reflected in the [Command] window.
- (2) When executing a command while each window is closed
When the relevant display window is closed, the execution results are always displayed in the [Command] window regardless of whether operation is in log mode or normal mode.






4.10 Debug Functions

This section outlines the debug features of the simulator, classified by function.

4.10.1 Loading Files

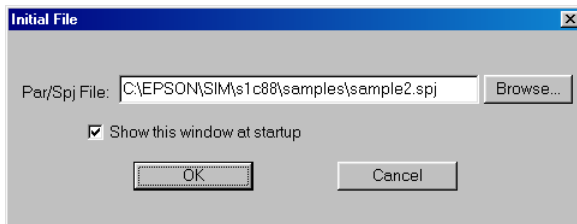
Table 4.10.1.1 lists the files read by the simulator and the load commands.

Table 4.10.1.1 Files and load commands

File	Type	Generation tool	Command	Menu	Button
1. Parameter file	.par	–	par	[File Load Parameter File...]	
2. LCD panel definition file	.lcd	LcdUtil	–	–	–
3. Component mapping file	.cmp	–	–	–	–
4. Port setting file	.prt	PrtUtil	–	–	–
5. Simulator project file	.spj	–	par	[File Load Parameter File...]	
6. IEEE-695 absolute object file	.abs	lc88	lf	[File Load File...]	
7. Motorola S2 program file	.psa	fil88xxx	lf	[File Load File...]	
8. Function option file	.fsa	fog88xxx or winfog	lf	[File Load File...]	
9. Symbol file	.sy	sy88, sym88	–	–	–
10. Command file	.cmd	–	com/cmw	[Run Command File...]	–

Files 1 to 4 are required for starting the simulator.

Either a parameter file (1) or a simulator project file (5) must be selected in the dialog box shown below at the first time the simulator starts up.

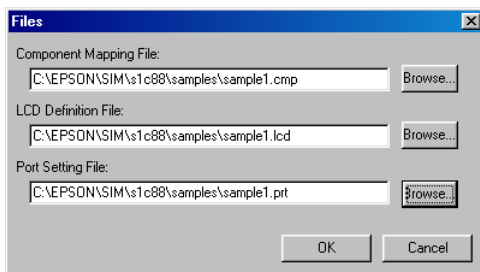


Enter the parameter file name or simulator project file name to the text box, or choose it using the [Browse...] button.

If the [Show this window at startup] check box is deselected, this dialog box will not appear from the next startup of the simulator and the same file will be selected.

Loading a parameter file resets the simulator. The memory mapping information set by the parameter file can be displayed using the ma command. Refer to the "S5U1C88000C Manual II" for more information on the parameter file.

When a parameter file is selected, the dialog box shown below appears to select files from 2 to 4.



When a simulator project file is selected, this dialog box does not appear because the file names (2 to 4) are specified in the file.

4 S1C88 FAMILY SIMULATOR

Enter a component mapping file name, LCD panel definition file name and a port setting file name to the respective text boxes, or chose using the [Browse...] button.

These files are selected once, the file names will appear in the text boxes at the next startup of the simulator allowing choice of the files by clicking the [OK] button only.

The lf command loads an IEEE-695 absolute object file (.abs), a Motorola S2 program file (.psa) or a function option HEX file (.fsa). The simulator distinguishes these files with the specified extension.

The symbol file is required to specify addresses using the symbols defined in the source when debugging a Motorola S2 program file. Debugging can be done even if this file is not loaded. The symbol file is loaded simultaneously with the program file by the lf command. However, it must be the same name (extension is .sy) and be located in the same directory as the program file. When the symbol file is loaded, the contents of the file can be displayed in the [Symbol] window or the [Command] window using the sy command.

When an IEEE-695 absolute object file that contains symbol information is loaded, the simulator does not read the symbol file as the object file allows symbolic debugging.

Refer to Section 4.9.3, for the command file.




4.10.2 Source Display and Symbolic Debugging Function

The simulator allows program debugging while displaying the C source statements. Address specification using a symbol name is also possible.

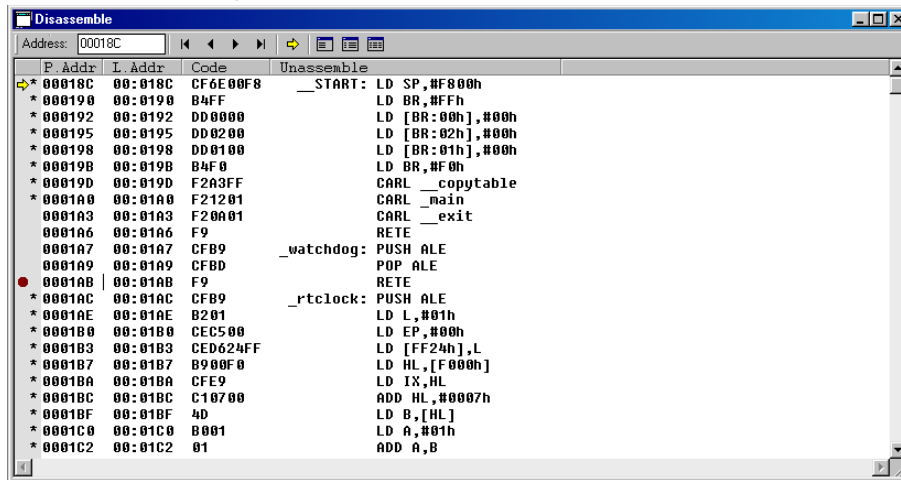
Displaying program code

The [Source] window displays the program in the specified display mode. The display mode can be selected from among the three modes: Disassemble display mode, Source display mode and Mix display mode.

Table 4.10.2.1 Commands/menu items/tool bar buttons to switch display mode

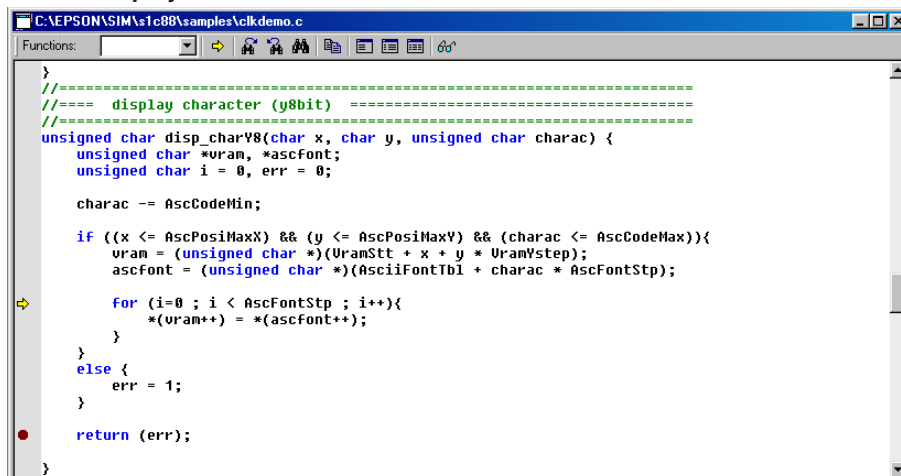
Function	Command	Menu	Button
Disassemble display mode	u	[View Source Disassemble]	
Source display mode	sc	[View Source Source]	
Mix display mode	m	[View Source Mix]	

(1) Disassemble display mode



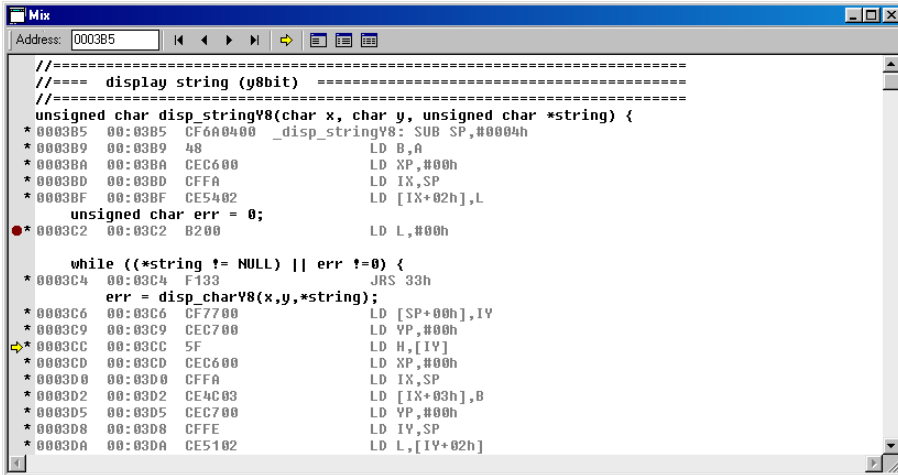
In this mode, the simulator displays the program codes after disassembling into mnemonics.

(2) Source display mode



In this mode, the source that contains the code at the current PC address is displayed. This mode is available only when an IEEE-695 absolute object file that contains source debugging information has been loaded.

(3) Mix display mode



This mode displays both sources and the disassembled codes of the corresponding object codes. This mode is available only when an IEEE-695 absolute object file that contains source debugging information has been loaded.

Refer to Section 4.6.5, "[Source] Window" for display contents and operation on the window.

Symbol reference

When debugging a program after loading an object file (.abs) in the IEEE-695 format, the symbols defined in the source file can be used to specify an address. This feature can be used when entering a command having <address> in its parameter from the [Command] window or a dialog box. However, the object file loaded must contain symbol information.

To perform symbolic debugging after loading a program file (.psa) in the Motorola S2 format, it is necessary to prepare a symbol file with the same name as the program file in the same directory. The symbol file is loaded simultaneously with the program file.

The symbols used in the program and the defined addresses can be displayed in the [Command] window or the [Symbol] window.

Table 4.10.2.2 Symbol list display command/menu item

Function	Command	Menu	Button
Displaying symbol list	sy	[View Symbol]	-

4.10.3 Displaying/Modifying Memory and Register Data

The simulator has functions to operate on the memory and registers. Available memory area is set to the simulator according to the map information that is given in a parameter file.

Memory operation

The following operations can be performed on the memory areas (ROM, RAM, display memory, I/O memory):

Table 4.10.3.1 Memory operation commands/menu item

Function	Command	Menu	Button
Dumping memory data	dd	[View Dump]	–
Entering/modifying memory data	de	–	–
Rewriting specified area	df	–	–
Coping specified area	dm	–	–
Searching data	ds	–	–

(1) Dumping memory

The memory contents are displayed in a specified size (Byte, Word, Long, Float, Double) hexadecimal dump format. If the [Dump] window is opened, the contents of the [Dump] window are updated; if not, the contents of the data memory are displayed in the [Command] window.

(2) Entering/modifying data

Data at a specified address is rewritten by entering hexadecimal data. Data can be directly modified on the [Dump] window.

(3) Rewriting specified area

An entire specified area is rewritten with specified data.

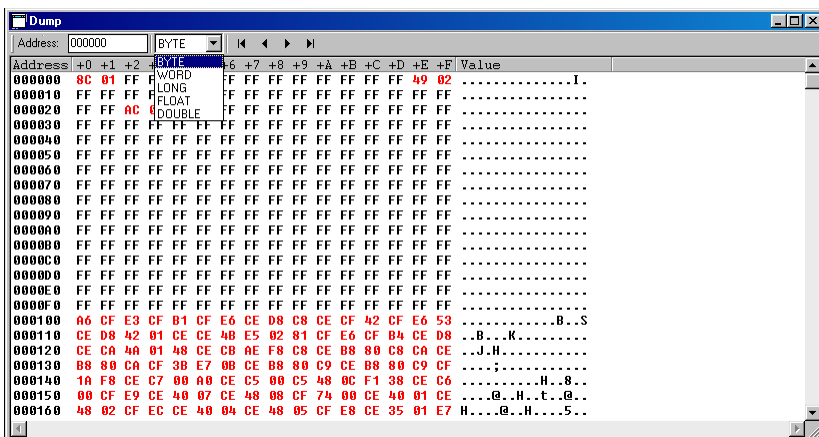
(4) Copying specified area

The content of a specified area is copied to another area.

(5) Searching data

An specified data can be searched within a specified area. The [Command] window displays the results up to 256 found data. The [Data] window shows found data within the current displayed area in green.

See Section 4.6.6, "[Dump] Window", for display contents and operation on the [Dump] window.



Operating registers

The following operations can be performed on registers:

Table 4.10.3.2 Register operation commands/menu item

Function	Command	Menu	Button
Displaying register values	rd	[View Register]	-
Modifying register value	rs	-	-

(1) Displaying registers

Register contents and the contents of the memory specified in register indirect addressing can be displayed in the [Register] or [Command] window.

Registers: PC, SP, IX, IY, A, B, H, L, BR, CB, NB, EP, XP, YP, SC (I1, I0, U, D, N, V, C, Z)
and CC (F3, F2, F1, F0)

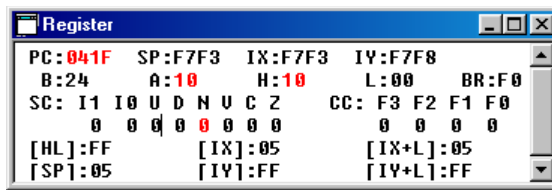
Memory: [HL], [SP], [IX], [IY], [IX+L], [IY+L]

(2) Modifying register values

The contents of the above registers can be set to any desired value.

The register values can be directly modified on the [Register] window.

See Section 4.6.7, "[Register] Window", for display contents and operation on the [Register] window.



4.10.4 Executing Program

The simulator can execute the target program successively or execute instructions one step at a time (single-stepping).



Successive execution

(1) Types of successive execution

There are two types of successive execution available:

- Successive execution from the current PC
- Successive execution after resetting the CPU

Table 4.10.4.1 Commands/menu items/tool bar buttons for successive execution

Function	Command	Menu	Button
Successive execution from current PC	g	[Run Go]	
Successive execution after resetting CPU	gr	[Run Go after Reset]	

(2) Stopping successive execution

Using the successive execution command (g <address1> <address2>), can specify up to two temporary break addresses that are only effective during program execution.

The temporary break address can also be specified from the [Source] window.

If the cursor is placed on an address line in the [Source] window and the [Go to Cursor] button clicked, the program starts executing from the current PC address and breaks after executing the instruction at the address the cursor is placed.

Note that when displaying C source in source display mode, the cursor must be located at one of the source lines expanded into effective source code. If the cursor is located at any source line, such as a comment line or declaration statement that is not compiled into object code, the program is not executed, even if you click the [Go to Cursor] button. (Refer to the description of the PC break function.)

Except being stopped by this temporary break, the program continues execution until it is stopped by one of the following causes:

- Break conditions set by a break set up command are met.
- The [Key Break] button is clicked.
- An undefined area is accessed.



[Key Break] button * When the program does not stop, use this button to forcibly stop it.

Note: If program execution is halted in C source display mode, the simulator displays the source for an object that includes the halted address. However, if no sources exist at the halted address, a [Source Files] dialog box is displayed, prompting for selection of a source file.

(3) Simulation for LCD panel display and key inputs

The [LCD] window shows the LCD panel images according to the program sequence during the program execution. It also allows simulation of key inputs using the keyboard of the computer. These functions are configured with the component mapping file, LCD panel definition file and port setting file loaded at the startup of the simulator. See Section 4.6.3, "[LCD] Window", for more information. Furthermore, serial/general port inputs/outputs and A/D conversion can be simulated by reading an IOT file from the [I/O Terminal] window. See Section 4.6.4, "[I/O Terminal] Window", for details.

Single-stepping

(1) Types of single-stepping

There are three types of single-stepping available:

- **Single-stepping C statements or instructions (STEP)**

In C source display mode, the program is single-stepped, one C source line at a time. In disassemble display or mix display mode, the program is single-stepped, one instruction at a time.

- **Single-stepping other than functions or subroutines (NEXT)**

In C source display mode, function calls in the program currently being executed are skipped by handling each function call from entry until the return simply as a single step. Other program parts are single-stepped in the same way as for STEP.

In disassemble display or mix display mode, the cars, carl, call, and int instructions till returned to the next step by a return instruction are executed as a single step. Other instructions are singled-stepped in the same way as for STEP.

- **Terminating at a function or subroutine (STEP EXIT)**

In C source display mode, the program is successively executed from the current function until it returns to the higher-level function, and is halted after returning. Do not run this single-stepping mode in the main function.

In disassemble display or mix display mode, the program is successively executed from the current subroutine until it is returned to the higher-level subroutine by a return instruction, and is halted after returning. At the highest level, the program is executed in the same way as when run by the g command. If a lower-level subroutine is called, and returned from it, the program execution is not halted.

In either case, the program starts executing from the current PC.

Table 4.10.4.2 Commands/menu items/tool bar buttons for single-stepping

Function	Command	Menu	Button
Stepping	s	[Run Step]	
Stepping except functions/subroutines	n	[Run Next]	
Exit from function/subroutine	se	[Run Step Exit]	

When executing s or n by command input, the number of steps to be executed can be specified, up to 65,535 steps. When using menu commands or tool bar buttons, the program is executed one step at a time.

In the following cases, single-stepping is terminated before a specified number of steps is executed:

- The [Key Break] button is clicked.
- An undefined area is accessed.

Single-stepping is not suspended by breaks set by the user such as a PC break or data break.



[Key Break] button * When the program does not stop, use this button to forcibly stop it.

(2) Display during single-stepping

In the initial simulator settings, the display is updated as follows:

When the [Source], [Register], or [Dump] window is open, the display contents are also updated after the last step has been executed. If the [Register] window is closed, its contents are displayed in the [Command] window.

The display mode can be changed so that display contents will be updated every step using the md command.

(3) Key entry simulation during single-stepping

Key entry status can be set on the [Key List] window displayed by clicking on the [Key List] button on the [LCD] window and is maintained during single-stepping.

(4) Precautions to be observed when single-stepping C sources



When single-stepping a program in C source display mode, the program is basically executed one source line at a time. However, source lines that do not have the corresponding object code, or lines without user sources (e.g., functions automatically generated by inline assembler or compiler) are skipped until the next line is reached that has effective object code. Accordingly, the number of steps executed varies depending on how C statements are written.

Example: `for(x=0; x<10; x++) a[x]=x;` ... Executed in one step.

```
for(x=0; x<10; x++)
    a[x]=x;           ... 20 steps need to be executed before exiting the for statement.
```

Resetting the CPU

Table 4.10.4.3 Commands/menu items/tool bar buttons for resetting CPU

Function	Command	Menu	Button
Reset CPU	rst	[Run Reset CPU]	
Successive execution after resetting CPU	gr	[Run Go from Reset]	

The CPU is reset when the gr command is executed, or by executing the rst command.

The following shows the initial settings when the CPU is reset.

(1) Internal registers of the CPU and memory

The CPU internal registers are initialized as follows during initial reset.

Table 4.10.4.4 Initial settings

Register name	Code	Bit length	Initial value
Data register A	A□	8□	Undefined
Data register B	B□	8□	Undefined
Index (data) register L	L□	8□	Undefined
Index (data) register H	H□	8□	Undefined
Index register IX	IX□	16□	Undefined
Index register IY	IY□	16□	Undefined
Program counter	PC□	16□	Undefined*
Stack pointer	SP□	16□	Undefined□
Base register	BR□	8□	Undefined
Zero flag	Z□	1□	0
Carry flag	C□	1□	0
Overflow flag	V□	1□	0
Negative flag	N□	1□	0
Decimal flag	D□	1□	0
Unpack flag	U□	1□	0
Interrupt flag 0	I0□	1□	1
Interrupt flag 1	I1□	1□	1
New code bank register	NB□	8□	01H
Code bank register	CB□	8□	Undefined*
Expand page register	EP□	8□	00H
Expand page register for IX	XP□	8□	00H
Expand page register for IY	YP	8	00H

* Reset exception processing loads the reset vector stored in bank 0, 000000H–000001H into the PC. At the same time, 01H of the NB initial value is loaded into CB.

The internal RAM and external RAM are not initialized at initial reset.

The respectively stipulated initializations are done for internal peripheral circuits.

(2) Redisplaying the [Source] and [Register] windows

Because the PC is reset, the [Source] window is redisplayed beginning with that address.

The [Register] window is redisplayed with the settings above.

4.10.5 Break Functions

The target program is made to stop executing by one of the following causes:

- Break command conditions are satisfied.
- The [Key Break] button is clicked.
- An undefined area is accessed.

Break by command

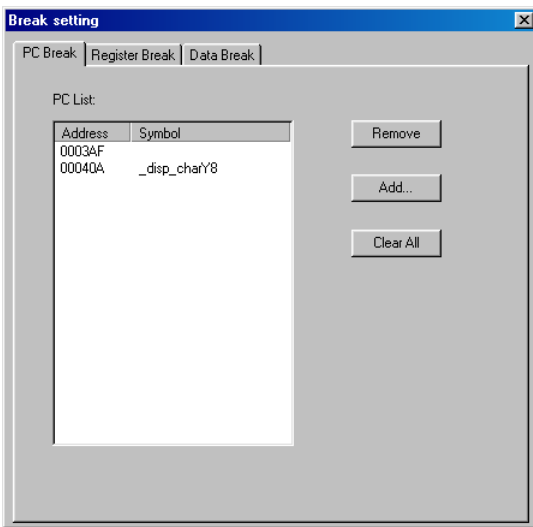
The simulator has three types of break functions that allow the break conditions to be set by a command. When the set conditions in one of these break functions are met, the program under execution is made to break.

(1) Break by PC

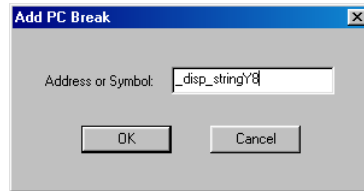
This function causes the program to break when the PC matches the set address. The program is made to break after executing the instruction at that address. The PC breakpoints can be set up to 64 addresses.

Table 4.10.5.1 Commands/menu items/tool bar button to set breakpoints

Function	Command	Menu	Button
Set breakpoints	bp	[Break Breakpoint Set...]	
Clear break points	bc (bpc)	[Break Breakpoint Set...]	



Selecting [Breakpoint Set...] from the [Break] menu displays the [Break setting] dialog box, listing the PC breakpoints set on its [PC Break] tab screen. To add a PC breakpoint, click the [Add...] button. This displays the [Add PC Break] dialog box. In this dialog box, enter the address (hexadecimal) or symbol at which a PC breakpoint is to be set and click [OK].



To remove a breakpoint, select its address from the list and click the [Remove] button. Or click the [Clear All] button to remove all of the PC breakpoints set.

- * The temporary break addresses that can be specified by the successive execution commands (g) do not affect the set addresses in the breakpoint list.

The addresses that are set as PC breakpoints are marked with a ● as they are displayed in the [Source] window.

Example in source display mode

```

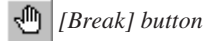
    }
    ● return (err);
    }
    
```

Example in disassemble display mode

```

0001A7 00:01A7 CFB9 _watchdog: PUSH ALE
0001A9 00:01A9 CFB0          POP ALE
● 0001AB 00:01AB F9          RETE
* 0001AC 00:01AC CFB9 _rtcLock: PUSH ALE
    
```


Using the [Break] button easily allows the setting and canceling of breakpoints.



Click on the line in the [Source] window at where the program break is desired (after moving the cursor to that position) and then click on the [Break] button. A ● mark will be placed at the beginning of the line indicating that a breakpoint has been set there, and the address is registered in the breakpoint list. Clicking on the line that begins with a ● and then the [Break] button cancels the breakpoint you have set, in which case the address is deleted from the breakpoint list.

Setting breakpoints during source display mode

In the [Source] window in source display mode, there are lines at which breakpoints can be set and those at which breakpoints cannot be set. No breakpoints can be set in source lines that do not have actual code generated.

```
Example: 1    void func(void)      // NG
          2    {                  // OK
          3        int a;         // NG
          4        int x=0;       // OK
          5        a = x;        // OK
          6    }
```

Line 1 is a function declaration that does not have actual code (same as a label declaration in the assembler). A breakpoint cannot be set here.

Line 3 is a variable declaration that does not have actual code. A breakpoint cannot be set here.

Line 4 is a variable declaration that has initialization code generated for it. A breakpoint can be set here.

Line 2 allows a breakpoint to be set. However, the breakpoint is set in line 4 (instruction at the beginning of that function).

Line 5 is an effective line that has actual code. A breakpoint can be set here.

Line 6 is a function termination (equivalent to mnemonic ret). A breakpoint can be set here.

However, if optimized during compiling, some lines become unusable in terms of setting a breakpoint. In the above example, since nothing is derived by executing each line (rewriting of only local variables involved, and that of global variables nonexistent), the actual code may be lost by optimization.

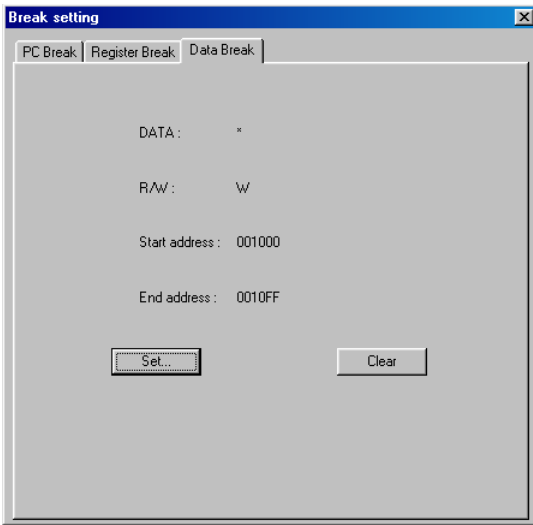
The same applies for lines whose execution can be halted by the [Go to Cursor] button.

(2) Data break

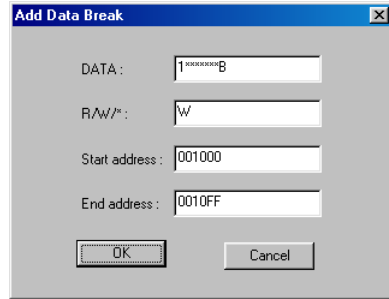
This break function allows a break to be executed when a location in the specified memory area is accessed. In addition to specifying a memory area in which to watch accesses, specification as to whether the break is to be caused by a read or write, as well as specification of the content of the data read or written. The read/write condition can be masked, so that a break will be generated for whichever operation, read or write, is attempted. Similarly, the data condition can also be masked in bit units. A break occurs after completing the cycle in which an operation to satisfy the above specified condition is performed.

Table 4.10.5.2 Commands/menu item to set data break

Function	Command	Menu	Button
Set data break condition	bd	[Break Data Break...]	–
Clear data break condition	bdc	[Break Data Break...]	–



Selecting [Data Break...] from the [Break] menu displays the [Break setting] dialog box, listing the data break conditions set on its [Data Break] tab screen. To set data break conditions, click the [Set...] button. This displays the [Add Data Break] dialog box. In this dialog box, enter the desired conditions and click [OK].



For example, if the program is executed after setting the data break condition as Address = 001000H–0010FFH, Data pattern = * (mask) and R/W = W, the program breaks after writing any data to the area from address 001000H to address 0010FFH.

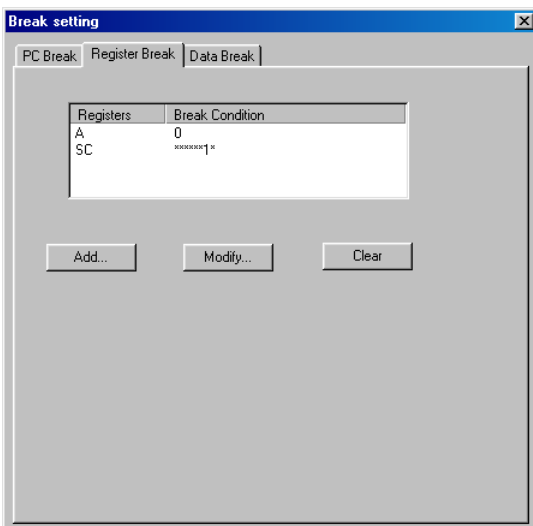
To clear the data break conditions, click the [Clear] button.

(3) Register break

This break function causes a break when the PC, SP, IX, IY, A, B, HL, BR, CB, NB, EP, XP and YP registers and flags (I1, I0, U, D, N, V, C, Z) reach a specified value. Each register can be masked (so they are not included in break conditions). The flag register can be masked in bit units. A break occurs when the above registers are modified to satisfy all set conditions.

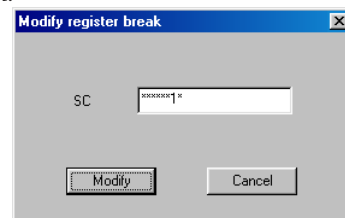
Table 4.10.5.3 Commands/menu item to set register break

Function	Command	Menu	Button
Set register break condition	br	[Break Register Break...]	–
Clear register break condition	brc	[Break Register Break...]	–

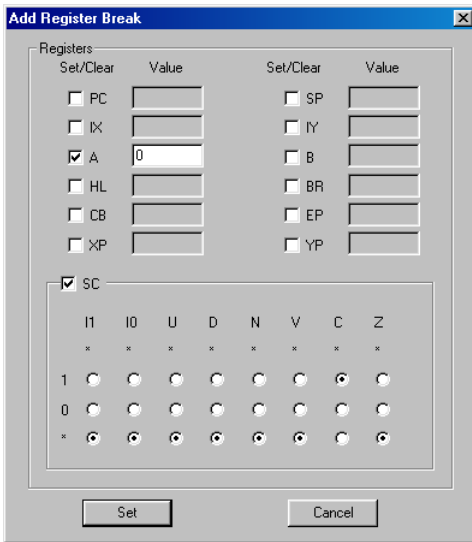


Selecting [Register Break...] from the [Break] menu displays the [Break setting] dialog box, listing the register break conditions set on its [Register Break] tab screen.

To correct the register break conditions set, select the register name from the list of the [Break setting] dialog box and click the [Modify...] button. The [Modify register break] dialog box shown below appears, displaying the conditions currently set. Correct the value shown in this dialog box and click the [Modify] button.



To clear the register break conditions, click the [Clear] button.



To set a new register break condition or add to the existing one, click the [Add...] button on the [Break setting] dialog box.

The [Add Register Break] dialog box shown to the left appears. Enter the desired condition and click [Set]. For example, if the program is executed after setting 0 for the data of register A and 1 for the data of flag C and masking all others, the program execution breaks when the A register is cleared to 0 and the C flag is set to 1.

(4) Other break commands

Commands are available to display all break conditions set in the [Command] window and to clear all break conditions.

Table 4.10.5.4 Other break commands

Function	Command	Menu	Button
Display all break conditions	bl	[Break Break List]	–
Clear all break conditions	bac	[Break Break All Clear]	

Forced break by the [Key Break] button

The [Key Break] button can be used to forcibly terminate the program under execution when the program has fallen into an endless loop or cannot exit a standby (HALT or SLEEP) state.



[Key Break] button

Break due to an illegal access

The program execution will be terminated when an error listed below occurs.

Accessing to an undefined program area

A break occurs when the target program jumps to a program area that has not been defined in the parameter file.

In this case the following message is displayed in the [Command] window.

Break by accessing no map program area

>

Accessing to outside the stack area

A break occurs when a stack operation is executed to an area outside the stack defined in the parameter file.

In this case the following message is displayed in the [Command] window.

Out of stack area

>

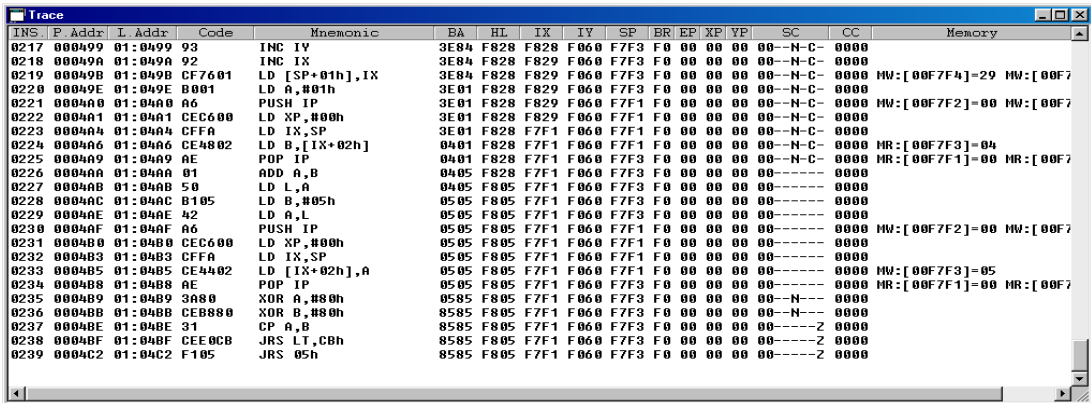
4.10.6 Trace Functions

The simulator has a function to trace program execution.

The trace function is initially disabled and can be enabled using the md command ([Option | Mode setting...]).

Trace data buffer and trace information

The simulator has a trace data buffer. When the simulator executes the program, the trace information on each executed instruction is taken into this buffer. The trace data buffer has the capacity to store information for 8,192 instructions. When the trace information exceeds this capacity, the data is overwritten, the oldest data first. Consequently, the trace information stored in the trace data buffer is always within 8,192 instructions. The trace data buffer is cleared when a program is executed, starting to trace the new execution data.



The following lists the trace information that is taken into the trace data buffer in every instruction execution cycle. This list is corresponded to display in the [Trace] window.

- INS: Executed instruction number (0 to 8191, decimal)
0000 means oldest trace data.
- P Addr: PC address (hexadecimal physical address)
- L Addr: PC address (hexadecimal logical address)
- Code: Instruction code (hexadecimal)
- Mnemonic: Disassembled instruction code
- BA to CC: Values of the CPU registers (hexadecimal)
- IDZC: Values of I, D, Z and C flags (binary) after cycle execution
- Memory: Memory read/write operation (MR: or MW:), accessed memory address (hexadecimal), and data (hexadecimal)

Displaying and searching trace information

The sampled trace information is displayed in the [Trace] window after a program execution has finished. In the [Trace] window, the entire trace data buffer can be seen by scrolling the window. The trace information can be displayed beginning from a specified cycle using a command. The display contents are as described above.

If the [Trace] window is closed, the information can be displayed in the [Command] window using a command.

Table 4.10.6.1 Command/menu item to display trace information

Function	Command	Menu	Button
Display trace information	td	[View Trace]	-

It is possible to specify a search condition and display the trace information that matches a specified condition.

The search condition can be selected from the following three:

1. Program's execution address
2. Address from which data is read
3. Address to which data is written

When the above condition and one address are specified, the system starts searching. When the trace information that matches the specified condition is found, the system displays the found data in the [Trace] window (or in the [Command] window if the [Trace] window is closed).

Table 4.10.6.2 Command/menu item to search trace information

Function	Command	Menu	Button
Search trace information	ts	[Trace Trace Search...]	–

Saving trace information

The trace information within the specified range can be saved to a file.

Table 4.10.6.3 Command/menu item to save trace information

Function	Command	Menu	Button
Save trace information	tf	[Trace Trace File...]	–

4.10.7 Coverage

The simulator can sample coverage information (i.e., information on addresses at which a program is executed) and the information can be displayed in the [Command] window.

This function is initially disabled and can be enabled using the md command ([Option | Mode setting...]). Because the executed address range is displayed as shown below, it is possible to know which areas have not been executed.

```
000100 - 000108
000200 - 00020f
:
```

Table 4.10.7.1 Coverage commands

Function	Command	Menu	Button
Display coverage information	cv	–	–
Clear coverage information	cvc	–	–

When the coverage function is enabled, the [Source] window shows the executed address with an * placed at the beginning of the line.

4.11 Command List

Table 4.11.1 Command list

Classification	Command	Function
Memory operation	dd [<addr1> [<addr2>] [{-BI-WI-LI-FI-D}]] [<addr1> [<@size>] [{-BI-WI-LI-FI-D}]]	Dump memory data
	de [<addr. <data1> [..<data16>]]	Enter memory data
	df [<addr1> <addr2> <data>] [<addr1> <@size> <data>]	Fill memory area
	dm [<addr1> <addr2> <addr3>] [<addr1> <@size> <addr2>]	Copy memory area
	ds [<addr1> <addr2> <data>] [<addr1> <@size> <data>]	Search memory data
Register operation	rd	Display register values
	rs [<reg> <value> [..<reg> <value>]]	Modify register value reg={PCISPIIXIYIAIBIHLIBRICBIEPIXIYPISCII1IIOUIDINIVIZIC}
Program execution	g [<addr1> [<addr2>]]	Execute program successively from current PC
	gr [<addr1> [<addr2>]]	Execute program successively after resetting CPU
	s [<step>]	Single stepping from current PC
	n [<step>]	Single stepping with skip subroutines
	se	Exit from subroutine
CPU reset	rst	Reset CPU
Break	bp [<addr1> [..<addr16>]]	Set breakpoints
	bc [<addr1> [..<addr16>]]	Clear breakpoints
	bpc [<addr1> [..<addr16>]]	
	bd [<data> {rlwl*} <addr1> <addr2>]	Set data break condition
	bdc	Clear data break condition
	br [<reg> <value> [..<reg> <value>]]	Set register break condition reg={PCISPIIXIYIAIBIHLIBRICBIEPIXIYPISC}
	brc	Clear register break condition
	bl	Display all break conditions
	bac	Clear all break conditions
Program display	u [<addr>]	Disassemble code display
	sc [<addr>]	Source display
	m [<addr>]	Mix display
Symbol display	sy [/a]	Display symbol list
	w <symbol> [;HIDIQIB] [/A]	Display symbol information
Load file	lf [<file name>]	Load program/option HEX file
	par [<file name>]	Load parameter file
Trace	td [<index>]	Display trace information
	ts [{pclrldw} <addr>]	Search trace information
	tf [{<index1> [<index2>]] <file name>]	Save trace information
Others	cv [<addr1> [<addr2>]]	Display coverage information
	cvc	Clear coverage information
	com [<file name> [<interval>]]	Load and execute command file
	cmw [<file name>]	Load and execute command file with execution interval
	rec [<file name>]	Record executed commands to file
	log [<file name>]	Logging
	ma	Display map information
	md [<option> <num> [..<option> <num>]]	Set modes option={-tml-cvl-sl-cm}
	q	Quit debugger
	?	Display command usage

4.12 Component Mapping File (.cmp)

The component mapping file (peripheral setting file) is the text file to record the information required to simulate the CPU, key entry and LCD display by the simulator. The files that contain the internal peripheral circuit data for available models are provided. Use these files after adding the necessary information using a text editor. Since the simulator saves the LCD panel information set in the simulator, such as LCD panel color, to this file, do not set the file attribute to Read Only.

Example:

```
[ Internal ]
CPU=CPU.bmc

LCD=LcdDrv88.bmc
K/P/R port=KPRport.bmc
SVD=SVD88.bmc
Serial=Serial88.bmc (note 1)
Adc=Adc88.bmc (note 2)

[ Settings ]
CpuType=88348
ChipMode=MCU
CpuModel=3
OSC1=32.768KHz
OSC3=4.9152MHz

[ External ]
Ext0=SED15210.bmc,080000,080002
```

[Internal] Internal peripheral circuit parameters

The file does not allow the user to modify these parameters. However, in a development environment in which the S1C63/S1C88 simulator has already been installed, components that support the serial interface must be added here (note 1). The same applies when using the A/D converter (note 2). When using some type of MCU that does not incorporate the serial interface or A/D converter, set Serial = NullDev.bmc for (note 1), or Adc = NullDev.bmc for (note 2).

Edit the parameter file (.par) to set the ROM/RAM size.

[Settings] CPU parameters

All of these parameters must be specified as follows:

[Settings]	⇐ Start of Settings section
CpuType= 88348	⇐ Family name
ChipMode= MCU	⇐ Chip type (MCU or MPU)
CpuModel= 3	⇐ CPU model (1 or 3)
OSC1= 32.768KHz	⇐ OSC1 clock frequency
OSC3= 4.9152MHz	⇐ OSC3 clock frequency

[External] External device parameters

Use the parameter file (.par) to set the ROM/RAM size.

When using external devices other than ROM and RAM, describe the information of the devices according to the address decoder settings. The currently supported devices are LCD drivers and backlights (BkLight).

```
[ External ]           ⇐ Start of External section
Ext0=SED15210.bmc,080000,080001
Ext1=BkLight.bmc,180004,180004,4L
```

4 S1C88 FAMILY SIMULATOR

Definition: Ext<N>=<device>.bmc, <Start_addr>, <End_addr>, <Option>

Ext<N>: N is a sequential device number beginning from 0.

<device>.bmc: Peripheral device definition file name

<Start_addr>: I/O map start address

<End_addr>: I/O map end address

<Option>: Peripheral specific option

Setting the external LCD driver

The S1D15210 shown in the example is mapped to a 2-byte space by connecting it to A0 the most significant bit of the address bus. The example shown above sets addresses 080000H (A0=0) and 080001H (A0=1) assuming that the connected CE signal becomes active from address 080000H. When data is written to this address, it is output to D0–D7 of the S1D15210. On the other hand, the D0–D7 data output from the S1D15210 can be read from this address.

Setting the BkLight

Specify the I/O address connected to the backlight. The start and end addresses have the same value. The option allows definition of the bit number and active level (H or L) to turn the backlight on. The example shown above defines so that the backlight goes on when bit 4 is set to low.

4.13 IOT File (.iot)

The IOT file is a text file in which settings and input data is written to simulate the general-purpose port and serial interface input/output operations and A/D conversions. These settings and data is loaded using the [I/O Terminal] window menu before performing input/output simulation. For more information on the [I/O Terminal] window and on using the window for input/output simulation, refer to Section 4.6.4, "[I/O Terminal] Window". To perform input/output simulation, first create an IOT file using a text editor as described below. Save the created data to a file using the file extension ".iot".

File contents

Write the input data, one for each section, e.g., [General I/O] and [A/D Converter]. Although the sequence of the sections does not matter, the data in each section is processed in the order in which they are written. No data need to be written for sections whose functions are not used.

Example: test.iot

<pre>[General I/O] watch P00, P01, P02, P10, P11, P20 5.0 K00=1 5.5 K00=0 8.0 K00=H 8.5 K00=L [A/D Converter] AVdd=3.0 AVss=0.2 AVref=2.9 [A/D CH2] 2.40,2.39,2.38,2.37,2.36 2.00, 2.10, 2.20, 2.30 0.5 0.65 0.80 [Serial CH1] "aBcDeFg" 0x25,0x20,0x41</pre>	<p>↓ Order in which tl data is processed</p> <p>↓ Order in which tl data is processed</p> <p>↓ Order in which tl data is processed</p>	<p>Section (general-purpose input/output ports)</p> <p>Section (A/D converter voltage settings)</p> <p>Section (A/D channel 2 input settings)</p> <p>Section (Serial interface channel 1)</p>
---	--	---

General-purpose input/output ports

Write settings for general-purpose input/output ports in the [General I/O] section.

Specifying monitor ports

Specify the general-purpose input/output ports whose input/outputs are to be monitored with a watch statement.

Example: watch P00, P01, P02, P10, P11, P20

In the watch statement, write the port names whose input/outputs are to be checked (Kxx, Rxx, Pxx). When input/outputs on the ports written here change state from high to low or vice versa, a log is displayed on the [I/O Terminal] window. When not monitoring the port state, there is no need to write port names.

Specifying ports for input

When using general-purpose input/output ports for input, write the time (in seconds) and the port name for which to input signal and the input level applied to the port on a line by line basis. Input simulations on the ports are performed in the order they are written, beginning with the one written at the top.

Example: 5.0 K00=1
8.5 K00=0
+0.7 P02=H, P10=L

The beginning of each line indicates the time (in seconds) at which the signal is input to the port. The time can be set with a precision of up to 1 ms, using the decimal point. The values set here basically represent an elapsed time after reset. Or the values can be specified to indicate an elapsed time after the previous input by prefixing the value with the positive sign "+".

4 S1C88 FAMILY SIMULATOR

Enter this specification in the form of "port name = input level". Write H or 1 for the high-level input, and L or 0 for the low-level input.

In the above example, a high signal is input to the K00 port 5.0 seconds after executing a reset instruction. Likewise, a low signal is input to the K00 port 8.5 seconds later, and a high and a low signal respectively are input to the P02 port and P10 port at the same time 9.2 seconds later.

If input port (K port) interrupts have been enabled, an interrupt is generated synchronously with the input timing.

A/D converter

Write settings for the A/D converter in the [A/D Converter] and [A/D CHn] sections.

Setting the A/D converter voltages

In the [A/D Converter] section, write the converter's set voltages.

Example: [A/D Converter]

```
AVdd=3.0
AVss=0.2
AVref=2.9
```

The AVdd, AVss, and AVref lines represent the A/D converter's positive power supply voltage, negative power supply voltage, and reference voltage respectively. Write the voltage values in units of volts. The voltages can be set with a precision of up to 1 mV, using the decimal point. When using the A/D converter, always write the [A/D Converter] section that includes these three lines.

Specifying analog input voltages

In the [A/D CHn] section, write the input voltages to the A/D converter. Specify the input voltages for each channel. Input voltages do not need to be specified for unused channels.

Example: [A/D CH4]

```
2.40,2.39,2.38,2.37,2.36
2.00, 2.10, 2.20, 2.30
[A/D CH5]
0.5
0.65
0.80
```

Write the input voltages in units of volts. The voltages can be set with a precision of up to 1 mV, using the decimal point. Each time the target program executes A/D conversion, the written data is loaded sequentially from the top. When writing input voltages in one line, separate each entry with a comma (.). In this case, the data is loaded sequentially from the left.

If interrupts are enabled, an interrupt is generated a certain time (sampling time + A/D conversion time) after the input began.

If A/D conversion is performed on any channel which does not have data specified, channel processing assumes a level-0 voltage (equivalent to AVss) input. If A/D conversion is performed for more than a specified data count, the last data specified is used repeatedly.

Furthermore, if a value smaller than AVss is specified, it is read out as level-0 (equivalent to AVss); if a value larger than AVref is specified, it is read out as the maximum level (equivalent to AVref).

Serial interface

Write settings for the serial interface in the [Serial CHn] section. Make this setting for each channel.

For some types of MCU that contain only a single channel, specify "CH1" (channel 1). No data needs to be written for unused channels.

Example: [Serial CH1]

```
"aBcDeFg"
0x25,0x20,0x41,FER
```

Write the input data in either hexadecimal notation or string notation (enclosed in "). Data in different notations cannot coexist in one line. Each time the program receives a signal via the serial interface, the written data is loaded sequentially from the top. When writing multiple hexadecimal data in a single line, delimit each entry with a comma (.). In this case, the data is loaded sequentially from the left.

If interrupts are enabled, an interrupt is generated synchronously with the timing at which the program finishes sampling the input. No interrupts are generated when signal is received on a channel that has no specified data.

The time required for sampling

Asynchronous system:	Cycle time of the clock source selected with the I/O register < number of bits (+ parity bit)
Clock-synchronous system (master):	Cycle time of the clock source selected with the I/O register < number of bits
Clock-synchronous system (slave):	User-defined transfer rate (bps)

For the clock-synchronous system (slave), specify the transfer rate (bps) in the manner shown below. Enter this specification before writing data. This setting is effective for only the synchronous system (slave).

```
Example: [Serial CH1]
         bps=2400
         "aBcDeFg"
         0x25,0x20,0x41,FER
```

Instead of hexadecimal data, the following can be specified:

```
FER: Framing error
PER: Parity error
OER: Overrun error
```

These data are valid only when the serial interface supports error interrupts. Do not specify this data for types of MCU that do not support error interrupts.

If interrupts have been enabled when this data is loaded, an error interrupt is generated.

For serial interface output, no particular specification needs to be written. When an output to any channel occurs, a log is displayed in the [I/O Terminal] window.

4.14 Simulator Project File (.spj)

The simulator project file is the text file that contains the names of the parameter file, LCD panel definition file, component mapping file and port setting file to be used. Use a text editor to create it. This file allows simultaneous selection of the files described above.

When the simulator starts up or the par command ([File | Load Parameter File]) is executed, a dialog box appears to select a parameter file or a simulator project file. When selecting a parameter file, a simulator project file is not necessary.

The following is an example of the file contents.

Example:

```
[Setting]
PAR=88348.par      ⇐ Parameter file
LCD=88348dmt.lcd  ⇐ LCD panel definition file
CMP=88348dmt.cmp  ⇐ Component mapping file
PRT=88348dmt.prt  ⇐ Port setting file
```

4.15 Restrictions

- For the MPU models supported by the simulator, please refer to the "Supported MCU models and peripherals" described in the Readme_E.txt.
- The supported external devices are ROMs, RAMs, the LCD drivers listed below, monochrome LCD panels, backlight, keys and key matrix.
S1D15210, S1D15600, S1D15601, S1D15602, S1D15605, S1D15606, S1D15607, S1D15608
- The simulator supports external ROM, RAM and LCD drivers that are interfaced through the external bus. Serial connection of the external LCD driver and accessing to external ROM/RAM via general-purpose ports are not supported.
- This simulator performs instruction-level simulation. Therefore, execution cycles shorter than the instruction cycle cannot be simulated.
- Since timers are simulated based on the instruction cycle, the timings are different from those of the actual hardware.
- Wait-cycles for accessing the external peripheral devices cannot be simulated properly.
- The following functions are not supported:
 1. Buzzer output
 2. TOUT/FOUT output
 3. D/A converter and analog comparator
 4. Event counter mode of programmable timer
- To simulate the sound generator, a sound card that supports playback of PCM sound source must be installed in the PC.
- Two or more simulators cannot be executed simultaneously for multiple simulations.
- When using the multiple-key entry reset function of the K ports, the simulator cannot reset the CPU even if the four assigned keys are pressed simultaneously. To reset using four keys, press three keys immediately after pressing one other key or press two keys immediately after pressing two other keys. Furthermore, the CPU will be reset immediately without the 2 seconds of authorization time needed for the ICE.

For the restrictions, support functions and bug information of the latest version, refer to rel_sim88_E.txt of S5U1C88000Q.

For the models supported by the latest version, refer to Readme_E.txt of S5U1C88000Q.

4.16 Simulator Messages

Status message list

Message	Description	Commands involved
Break by accessing no map program area	Break caused by accessing undefined program-memory area.	
Break by data break	Break caused by data break condition.	
Break by PC break	Break caused by PC breakpoint.	
Break by register break	Break caused by register break condition.	
Key Break	Break caused by [Key Break] button.	
No coverage data	No coverage information.	cv
No matched trace data	No trace information matches the specified condition.	ts
No trace data	No trace information.	td, tf, ts
Out of stack area	Break caused by accessing outside stack area.	
Set to log off mode	Set log function off.	log
Set to log on mode	Set log function on.	log
Set to record off mode	Set record function off.	rec
Set to record on mode	Set record function on.	rec
Simulator initialization error!	Failed to initialize the simulator.	
Symbol file does not exist	The corresponding symbol file does not exist in the same directory as the program file.	lf
Symbol file is loaded	The corresponding symbol file is loaded with the program file.	lf

Error message list

Message	Description	Commands involved
Address out of range, use 0 - 0x7FFFFFFF	The address is outside program memory area.	g, bp, bc(bpc), u, ts, cv
Address out of range, use 0 - 0xFFFFFFF	The address is outside data memory area.	dd, de, df, dm, bd, ts
Cannot open file	The file cannot be opened.	lf, par, com, cmw
Data out of range, use 0 - 0xFF	Data value is invalid.	de, df
End address < start address	The end address is less than the start address.	dd, df, dm, cv
End index < start index	The end index is less than the start index.	tf
Error file type (extension should be CMD)	The file extension is invalid as command file.	com, cmw
Error file type (extension should be PAR)	The file extension is invalid as parameter file.	par
Incorrect number of parameters	Number of parameters is invalid.	
Incorrect r/w option, use r/w*	The read/write option is invalid.	bd
Incorrect register name, use PC/SP/SP/IX/IY/A/B/HL/BR/CB/EP/XP/YP/SC	The register name is invalid for setting register break condition.	br
Incorrect register name, use PC/SP/SP/IX/IY/A/B/HL/BR/CB/EP/XP/YP/SC/11/I0/U/D/N/V/Z/C	The register name is invalid.	rs
Index out of range, use 0 - 8191	The index is over the valid range.	td
Input address does not exist	Attempt is made to clear a breakpoint that has not been set.	bp, bc (bpc)
Invalid command	The command is invalid.	
Invalid data pattern	The data pattern is invalid.	bd, br
Invalid display unit, use -B/-W/-L/-F/-D	The option is invalid for assigning display unit.	dd
Invalid file name	The file extension is invalid as program or function option file.	lf
Invalid option, use -tm/-cv/-s/-cm	The option is invalid for setting mode.	md
Invalid value	The input value is invalid.	
Maximum nesting level(5) is exceeded, cannot open file	The nesting level of command file is over 5.	com, cmw
No symbol information	No symbol information (symbol file is not loaded).	sy
Number of steps out of range, use 0 - 65535	Number of steps in single-stepping is invalid.	s, n
This command is not supported in current mode	Trace/coverage commands are not supported in trace/coverage off mode.	td, ts, tf, cv, cvc

Warning message list

Message	Description	Commands involved
64 break addresses are already set	Attempt is made to set more than 64 breakpoints.	bp
Break address already exists	The address is already set as a breakpoint.	bp
Identical break address input	The same address is input more than once in the command line.	bp

5 LCD PANEL CUSTOMIZE UTILITY

5.1 Overview

The LCD Panel Customize Utility (LcdUtil.exe, hereafter LcdUtil) creates the panel layout and COM/SEG port assignment data needed to simulate monochrome LCD panel displays with the simulator (sim63.exe, sim88.exe). Since they are loaded from bitmap files (.bmp), you can create icons or other display objects using general-purpose paint software to simulate the end-product screen. This utility is a device-independent tool; LCD driver settings that differ between microcomputer types are made using a device information definition file.

5.2 Input/Output Files

Figure 5.2.1 shows the input/output files for LcdUtil.

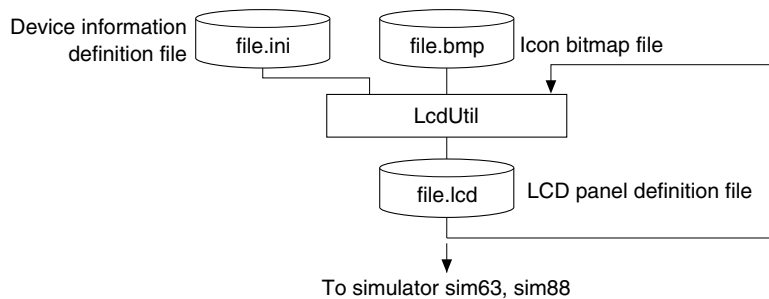


Fig. 5.2.1 LcdUtil input/output files

Device information definition file (file_name.ini)

This file contains information on LCD pins for an S1C63xxx/S1C88xxx. Use only files supplied by Seiko Epson. This file is compatible only with the model indicated by the file name. Do not attempt to modify the contents of the file or use the file for any other model.

Bitmap file (file_name.bmp)

Graphic objects representing icons are loaded from files in this format (monochrome bitmap). After loading multiple parts individually, edit the layout in LcdUtil.

LCD panel definition file (file_name.lcd)

This file contains information on the panel layout bitmap and COM/SEG pin assignments. Load this file into the simulator when simulating LCD display.

5.3 Starting and Exiting

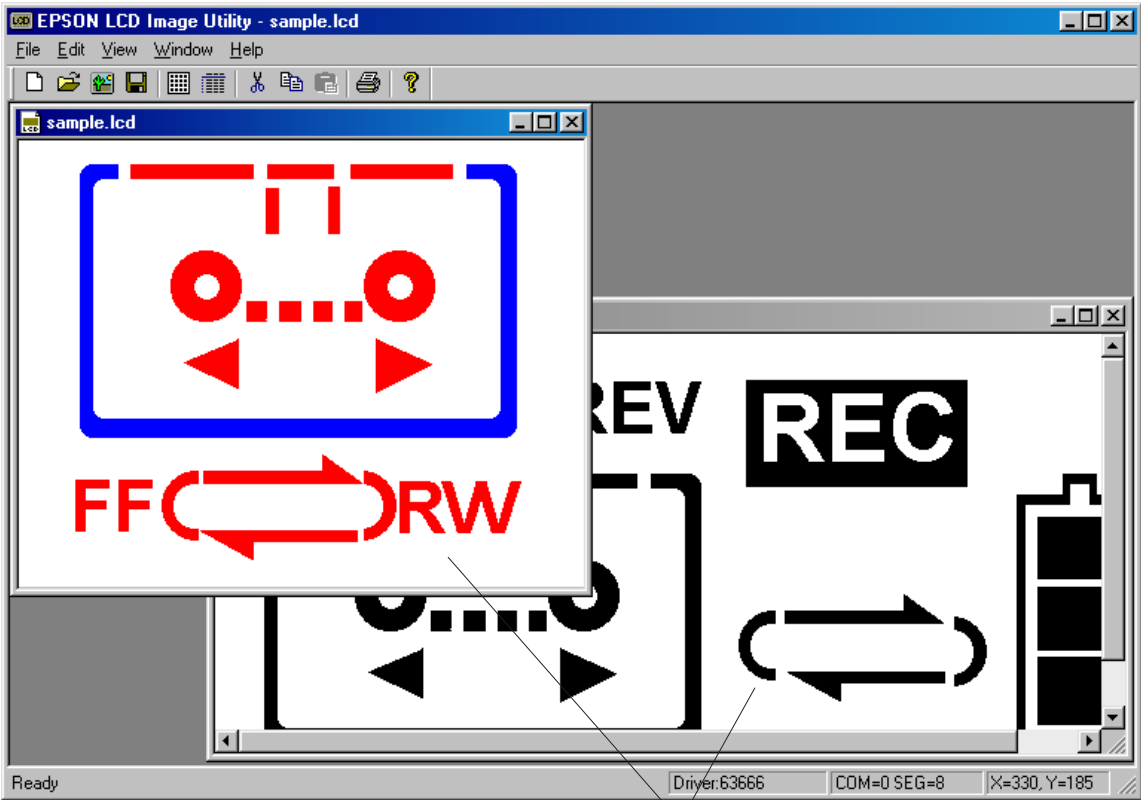


LcdUtil.exe

Double-click this icon to start LcdUtil.

To exit LcdUtil, select [Exit] from the [File] menu.

5.4 Window



Panel edit window

Panel edit window

When you open a bitmap file (.bmp) or LCD panel definition file (.lcd), the file is displayed in this window. In this window, you can lay out the panel or assign COM/SEG ports. You can open multiple instances of this window and move data between the two windows through Drag & Drop.

Basic window operations

Closing and activating the window

To close the window, click the [Close] button on the window. The windows being opened are listed on the [Window] menu. Select a window name from this menu to activate the window, or click anywhere in the desired window. You can also use the [Ctrl] + [Tab] keys to switch the active window from one window to the next.

Resizing and moving the window

Drag the window boundary to resize any window. The [Minimize] and [Maximize] buttons work in the same way as for general Windows applications. You can reposition any window within the display by dragging the title bar, but the panel edit window can only be resized and moved within the application window. If the image displayed in a window is extends beyond the window in any direction, a scroll bar appears.

Other

To align open windows, select [Cascade] or [Tile] from the [Window] menu.

5.5 Menus and Toolbar

5.5.1 Menus

[File] menu

File	
<u>N</u> ew	Ctrl+N
<u>O</u> pen...	Ctrl+O
Open Bitmap File...	
<u>C</u> lose	
<u>S</u> ave	Ctrl+S
Save <u>A</u> s...	
<u>P</u> rint...	Ctrl+P
Print <u>P</u> review	
Print <u>S</u> etup...	
1 icon.bmp	
2 sample.lcd	
<u>E</u> xit	

[New] ([Ctrl] + [N])

Opens a new panel edit window.

[Open...] ([Ctrl] + [O])

Opens an LCD panel definition file (.lcd).

[Open Bitmap File...]

Opens a bitmap file (.bmp).

[Save] ([Ctrl] + [S])

Saves the contents of the active panel edit window to the LCD panel definition file (.lcd), overwriting the previous file.

[Save As...]

Saves the contents of the active panel edit window to the LCD panel definition file (.lcd) under another name.

[Print...] ([Ctrl] + [P])

Sends the bitmap of the active panel edit window to a printer.

[Print Preview]

Displays an on-screen print image of the active panel edit window.

[Print Setup...]

Brings up a dialog box to select paper size or printer.

File list

Listed here are recently opened files. You may open any of these files by selecting its filename.

[Exit]

Exits LcdUtil.

[Edit] menu

Edit	
<u>C</u> ut	Ctrl+X
<u>C</u> opy	Ctrl+C
<u>P</u> aste	Ctrl+V
I <u>n</u> sert dot <u>m</u> atrix	Ctrl+M
<u>I</u> con List	Ctrl+I
Resize LCD	
Group Icon	
Release Group	

[Cut] ([Ctrl] + [X])

Cuts and copies a selected portion of the panel edit window to the clipboard.

[Copy] ([Ctrl] + [C])

Copies a selected portion of the panel edit window to the clipboard.

[Paste] ([Ctrl] + [V])

Pastes the portion copied to the clipboard into the upper left corner of the panel edit window.

[Insert dot matrix] ([Ctrl] + [M])

Inserts a dot matrix image into the panel edit window. Specify sizes and other parameters using the dialog box.

[Icon List] ([Ctrl] + [I])

Displays a list of icons currently in the active panel edit window. You can also perform COM/SEG assignments from the dialog box open here.

[Resize LCD]

Sets the size of the LCD panel. When you open a new panel edit window, the default size is 640 < 480.

[Group Icon]

Groups multiple icons into one icon.

[Release Group]

Ungroups the grouped icon into individual icons.

[View] menu



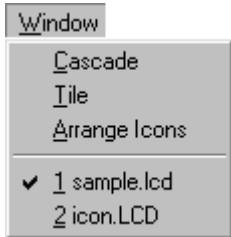
[Toolbar]

Alternately shows or hides the toolbar as you click this menu command.

[Status Bar]

Alternately shows or hides the status bar as you click this menu command.

[Window] menu



[Cascade]

Aligns the open panel edit windows, overlapping them diagonally.

[Tile]

Aligns the open panel edit windows by placing them side by side in rows.

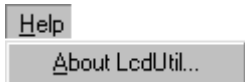
[Arrange Icons]

Lines up the minimized panel edit window icons at the bottom of the application window area.

Window list

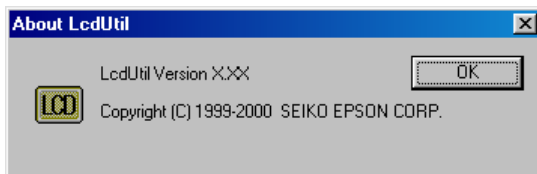
Currently open panel edit window names are listed here. Select a window name here to make the corresponding panel edit window active.

[Help] menu



[About LcdUtil...]

Displays version information for LcdUtil.



5.5.2 Toolbar Buttons



[New] button

Opens a new panel edit window.



[Open] button

Opens an LCD panel definition file (.lcd).



[Bitmap] button

Opens a bitmap file (.bmp).



[Save] button

Saves the contents of the active panel edit window to the LCD panel definition file (.lcd), overwriting the previous file.



[Dot Matrix] button

Inserts a dot matrix image into the panel edit window. Specify sizes and other parameters using the dialog box.



[Icon List] button

Displays a list of icons present in the active panel edit window. You can also perform COM/SEG assignments from the dialog box opened here.



[Cut] button

Cuts and copies a selected portion from the panel edit window to the clipboard.



[Copy] button

Copies a selected portion from the panel edit window to the clipboard.



[Paste] button

Pastes the portion copied to the clipboard into the upper left corner of the panel edit window.



[Print] button

Sends the bitmap of the active panel edit window to a printer.



[About] button

Displays version information for LcdUtil.

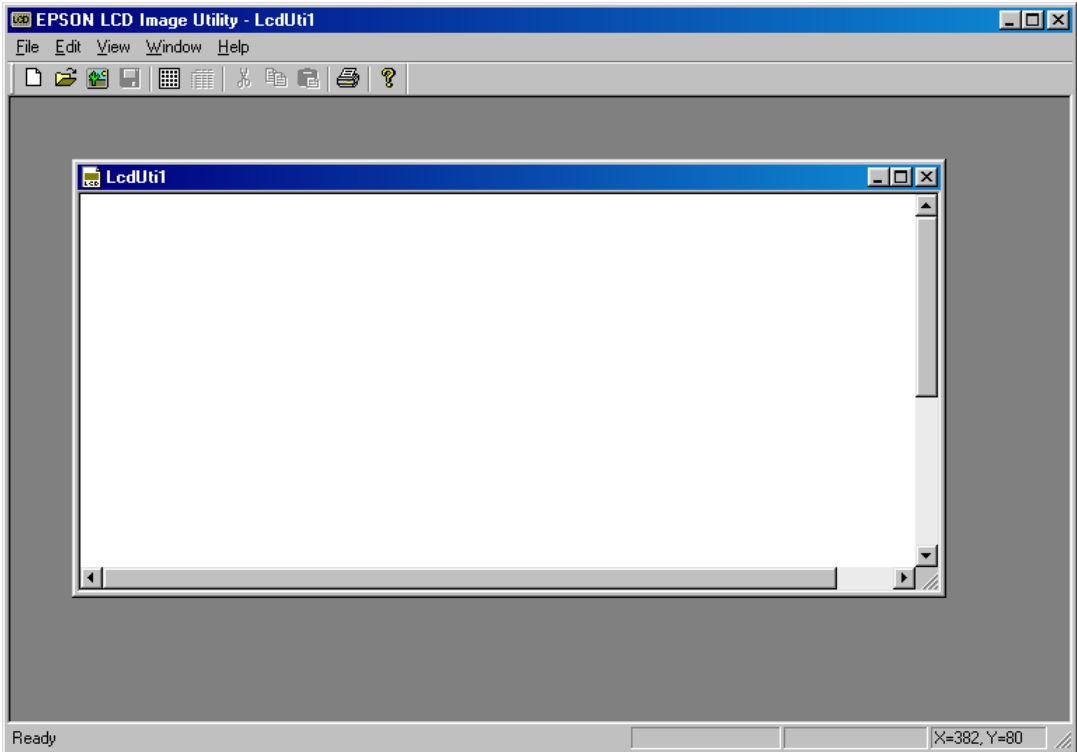
5.6 Creating LCD Panel Data

The panel edit window allows you to lay out icons and dot (pixel) matrix blocks just as they appear in the actual panel. COM/SEG ports can also be assigned from the panel edit window, as explained below.

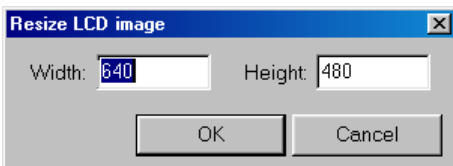
5.6.1 Creating a New Panel and Setting the Panel Size

To create an LCD panel definition file by placing a dot (pixel) matrix and icons on a blank panel edit window, you must first set up a new panel according to the following procedure.

- 1) Select [New] from the [File] menu (or click the [New] button). A blank panel edit window opens.



- 2) Select [Resize LCD] from the [Edit] menu. A [Resize LCD image] dialog box appears.



Enter a desired panel size and click [OK]. The value specified here represents the size of the LCD panel simulated with the simulator and must be specified by the number of pixels on the computer screen. The default size when creating a new panel is 640 × 480 pixels. You also need to set sizes for icon bitmaps and dot matrix to be placed. The maximum panel size is 1280 × 1024 dots.

You can change the panel size after finishing placement of icons or dot matrix.

A space for the panel has now been created. Here, you can place a dot matrix or the bitmaps of icons you've created separately. When creating an LCD panel definition file based on icons loaded from a bitmap file, the procedure given above for creating a new panel is unnecessary.

5.6.2 Creating Icons

This section explains how to assign icons to ports.

Creating bitmap data

Prepare bitmaps of icons using a scanner, or by creating with general paint software. Consider the following when preparing bitmaps.

Number of colors and file format

Create icons in black on white background and save the created icons in monochrome bitmap format (.bmp). Although LcdUtil can load bitmaps in up to 65,535 colors, this data is converted into black & white two-level data when loaded.

Size

LcdUtil allows you to specify the size of the LCD panel to be simulated in number of on-screen pixels. ([Resize LCD] on the [Edit] menu.) Determine the sizes of individual icons to suit the simulated panel size as you create them. Icon sizes cannot be adjusted in LcdUtil. The maximum bitmap data size is 1280 x 1024 dots.

Pixel size can be specified in relation to pixels on the computer screen. When simulating a panel configured with a mixture of icons and dot matrix, consider the size of the matrix as you determine panel and icon size. The panel size can be enlarged or reduced in some steps between 25% and 200% in the simulator.

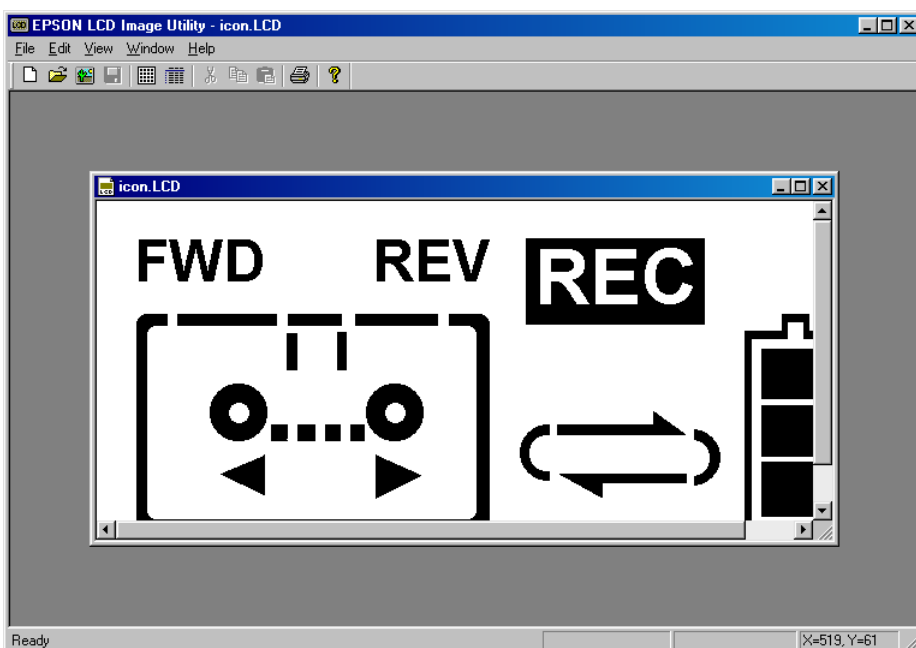
If you only need to verify the operation of your program, you are not required to create precise, faithful copies of icon bitmaps. This capability is merely provided to enable highly-accurate display simulation for design evaluation of the LCD panel.

Number of files

All of the icons may be saved into a single file, or in multiple separate files. Due to the limited editing capabilities of LcdUtil, we recommend saving icons to a single file when creating bitmaps. We also recommend determining their layout before you load the file.

Loading bitmap data

Start LcdUtil and select [Open Bitmap File...] from the [File] menu (or click the [Bitmap] button). A standard [Open] dialog box appears. Select and load the created bitmap file. A panel edit window opens, showing the loaded bitmap.



Editing icon layouts

From the loaded bitmap, areas comprised of consecutive black pixels are cut out as parts and recognized as icons. You can perform the following operations on these parts.

Selection

Click and select a part to change its highlight color to blue. The operations described below and COM/SEG port assignments are applied to the part in this selected state.

Move

Drag the part. You can move it into another panel edit window.

Copy

Copy and paste a part using the menu or toolbar buttons.

Delete

Use the [delete] key to delete a part.

Group

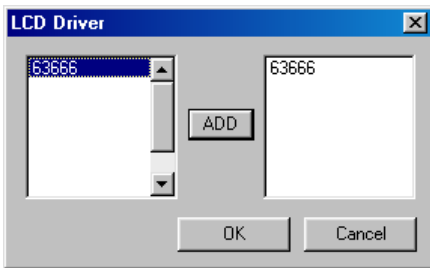
Hold down the [Ctrl] key while clicking one part and another. This allows selection of multiple parts. Then select [Group Icon] from the [Edit] menu. The parts are now grouped and may be treated as a single icon. To ungroup the grouped icon, select [Release Group] from the [Edit] menu.

Note: LcdUtil is provided with simple editing function. We recommend saving icons to a single file when creating bitmaps, as well as determining their layout before loading the file.

Port assignments

You can perform COM/SEG port assignments in the panel edit window. The procedure is given below.

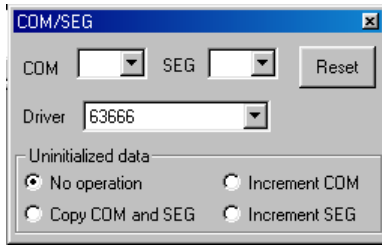
- 1) Double-click the icon to which you want to assign a port.
- 2) An [LCD Driver] dialog box appears.



From the list, select the model for which your application is being developed and click the [ADD] button. To use one of the external LCD drivers on the list in the S1C88 Family, select and add it in the same way. Then click the [OK] button. To cancel, click [Cancel].

* This dialog box appears when you assign ports for the first time. It is not displayed once you select a driver. Since the driver cannot be changed after ports are assigned, choose a driver carefully.

3) A [COM/SEG] dialog box appears.



Select a COM port and SEG port number from the pull-down list.

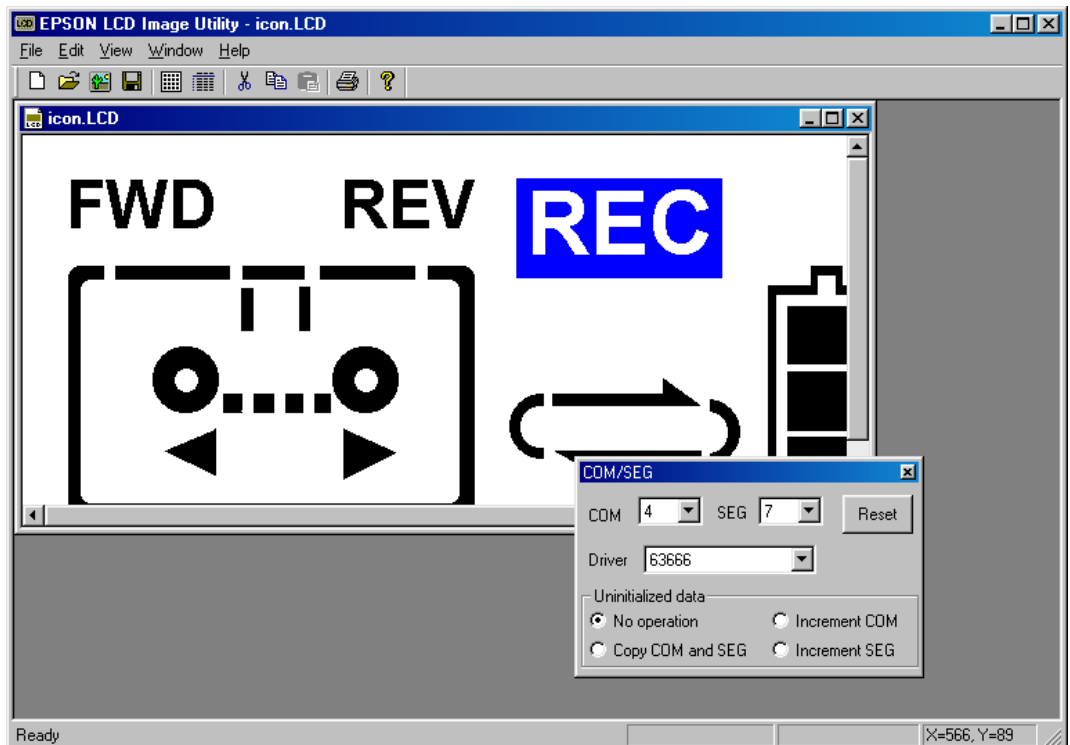
Click the [Reset] button to clear the selected COM/SEG ports. The selected [Driver] is also restored to the default driver.

After selecting COM and SEG ports, click anywhere on the panel edit window or close this dialog box before assigning the next icon.

The [Uninitialized data] radio buttons are used to select the following functions:

- | | |
|------------------|--|
| No operation | The COM and SEG boxes are cleared every time an undefined icon is selected. |
| Copy COM and SEG | The COM and SEG boxes maintain the previous set numbers even if an undefined icon is selected. |
| Increment COM | The SEG box maintains the previous set numbers even if an undefined icon is selected. The COM port number is incremented (+1). |
| Increment SEG | The COM box maintains the previous set numbers even if an undefined icon is selected. The SEG port number is incremented (+1). |

The icons with assigned ports are displayed in red. Moving the mouse pointer over any of those icons displays its assigned port and driver information in the status bar. The port information is also shown at the mouse cursor position.



For icons comprised of multiple parts, assign the same COM/SEG port to each part, or group them into a single icon before assigning COM/SEG ports. This allows selection of all parts with a single click.

This dialog box is also displayed when you double-click an icon with assigned ports. This allows you to correct port assignments at a later time.

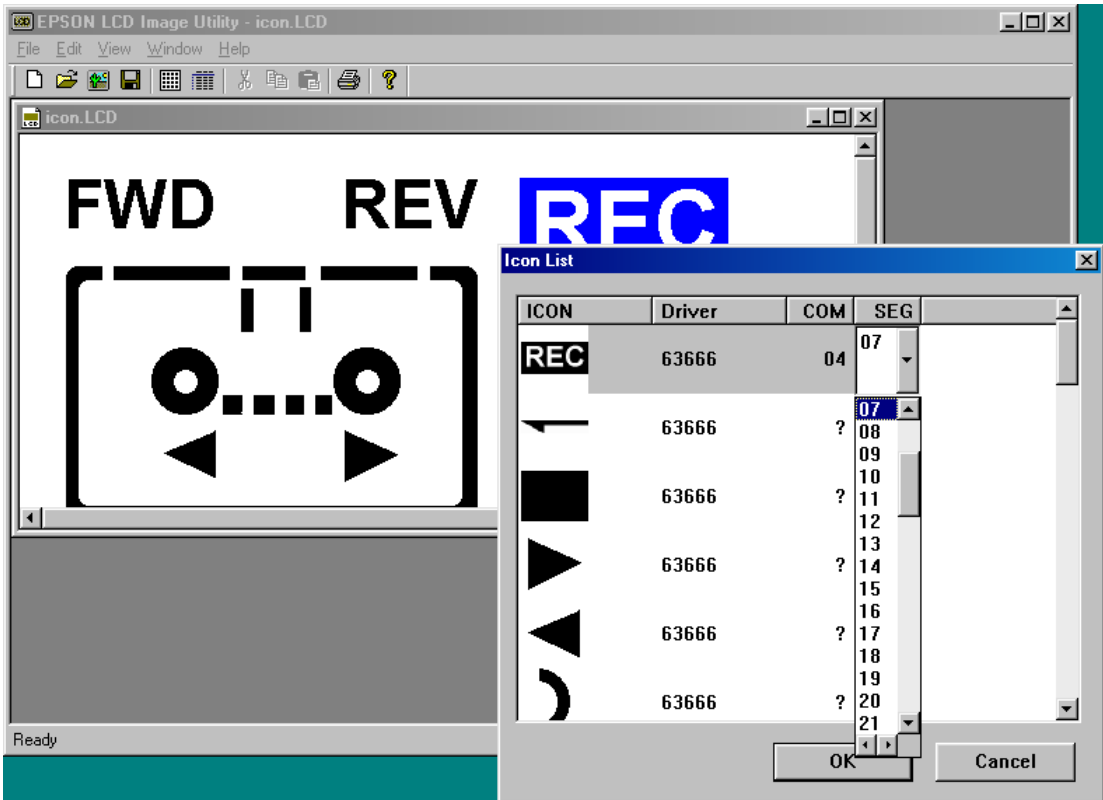
5 LCD PANEL CUSTOMIZE UTILITY

Note: When you copy and paste an icon with ports already assigned, or move such an icon into another panel edit window, its port assignment information is cleared. When editing layout of a panel on another panel edit window, move the icons for it to that window before assigning ports. If you group parts with different assigned ports into a single icon, the port assignment for the last part selected for grouping is applied to all parts of the grouped icon. The previous settings will not be restored when you ungroup the icon.

- 4) Select [Save] or [Save As...] from the [File] menu to save the contents set as an LCD panel definition file.

Displaying an icon list

Select [Icon List] from the [Edit] menu (or click the [Icon List] button) to display a list of icons within the currently active panel edit window.

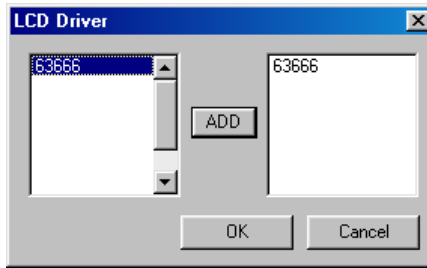


Click and select an icon on the list. The icon in the panel edit window turns blue to indicate the icon's position in the panel edit window of the selected icon. When you click a driver or COM/SEG number on this list, a pull-down list is displayed to enable you to change the set contents.

5.6.3 Creating a Dot (pixel) Matrix

This section explains how to create a dot matrix and how to assign ports to a dot matrix.

- 1) Create a new panel according to the procedure described in Section 5.6.1, or open a bitmap file containing icons or an LCD panel definition file. For the panel size, first calculate the size of the dot matrix, then determine a panel size greater than this value.
- 2) Select [Insert dot matrix] from the [Edit] menu (or click the [Dot Matrix] button).
- 3) The [LCD Driver] dialog box appears.



Select the model for which you are developing your application from the list and click the [ADD] button. To use one of external LCD drivers on the list, select and add it in the same way and click the [OK] button. To cancel, click [Cancel].

* This dialog box appears when no LCD drivers have been selected. It is not displayed when icons already have assigned ports.

- 4) A [Dot (pixel) Matrix] dialog box appears.

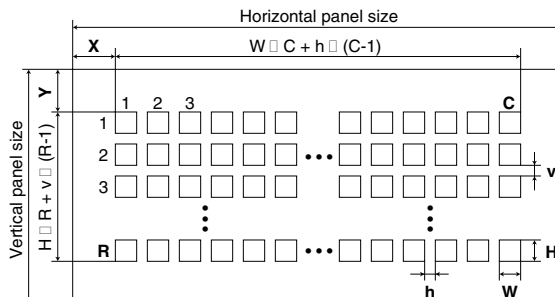
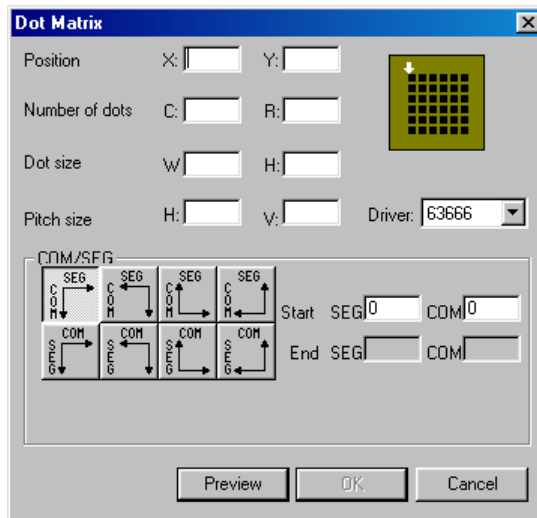


Fig. 5.6.3.1 Setting up a dot matrix

5 LCD PANEL CUSTOMIZE UTILITY

Set the following in this dialog box.

Position

Specify the distance from the upper left edge of the panel to the upper left edge of the dot matrix by the number of pixels on the computer screen. (X and Y in the diagram)

Number of dots

Specify the number of dots along the horizontal and the vertical directions of the dot matrix. (C and R in the diagram)

Dot size

Specify dot size for the dot matrix by number of pixels on the computer. (W and H in the diagram)

Pitch size

Specify the intervals at which individual dots are displayed by number of pixels on the computer. (h and v in the diagram)

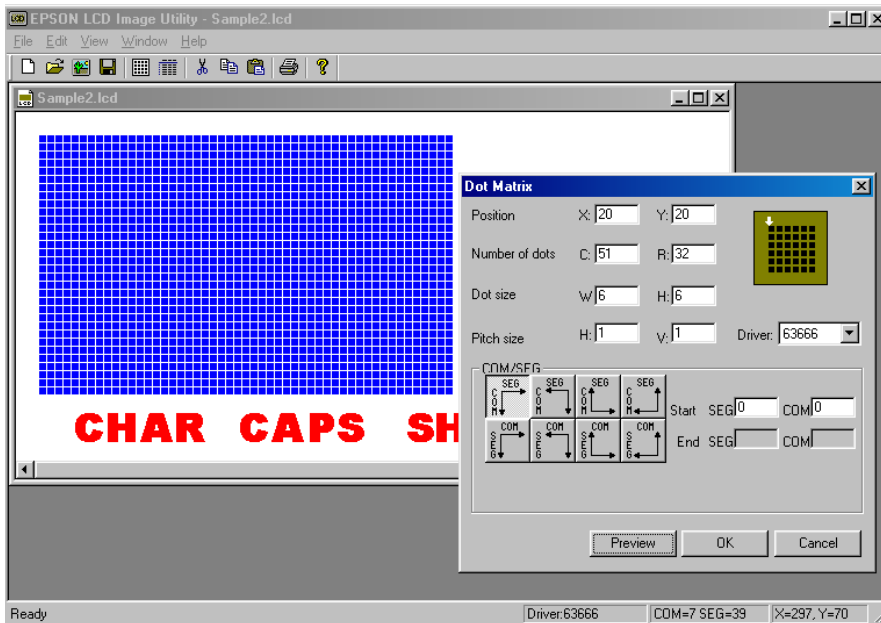
Driver

Select an LCD driver. If you have multiple LCD drivers driving a single dot matrix, create a dot matrix for each driver.

COM/SEG

Use one of the eight buttons to select a starting point for assignment. In the [Start] text box, set the COM/SEG port numbers assigned to that dot. Other dots are assigned automatically, according to the button you select.

Click the [Preview] button after setting each of the above [items]. A dot matrix of the specified size is displayed at the specified position in the panel edit window. This preview is shown to allow you to verify the contents you've set. The matrix is not actually created until you click the [OK] button.



The created dot matrix is shown in red. Move the mouse pointer over any of the dots to display information on assigned ports and driver in the status bar. The port information is also displayed at the mouse cursor position.

When you double-click the dot matrix, a [Dot Matrix] dialog box appears to allow you to change the settings at a later time. You can also change the position of the dot matrix by dragging it to a desired position.

Note: Copying and pasting a dot matrix deletes its port assignment information, although information on position and size is retained.

5.7 *Precautions*

For the restrictions, support functions and bug information of the latest version, refer to rel_utility_E.txt of S5U1C88000Q.

6 BITMAP UTILITY

6.1 Overview

The Bitmap Utility (BmpUtil.exe, hereafter BmpUtil) creates bitmap image data (e.g., character data) for the dot matrix LCD display. The output data is created in the assembly source format with specified labels assigned to it to allow you to assemble data without modifications and link it to the program. This utility is compatible with bit layouts for the LCD controller and display memory of the model listed in Sections 3.3 and 4.3. (see the Technical Manual for the LCD controller and display memory.)

6.2 Input/Output Files

Figure 6.2.1 shows input/output files for BmpUtil.

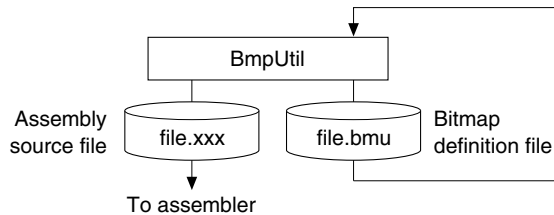


Fig. 6.2.1 BmpUtil input/output files

Bitmap definition file (file_name.bmu)

This file contains the defined bitmap data and the COM/SEG output pattern. This file always has ".bmu" as its extension.

Assembly source file (file_name.xxx)

This text file can be read by the S1C63/S1C88 Family assembler. Bitmap data is written with a data setting pseudo-instruction (.word in the S1C63 Family, DB in the S1C88 Family). You can assign an extension to this file.

6.3 Starting and Exiting



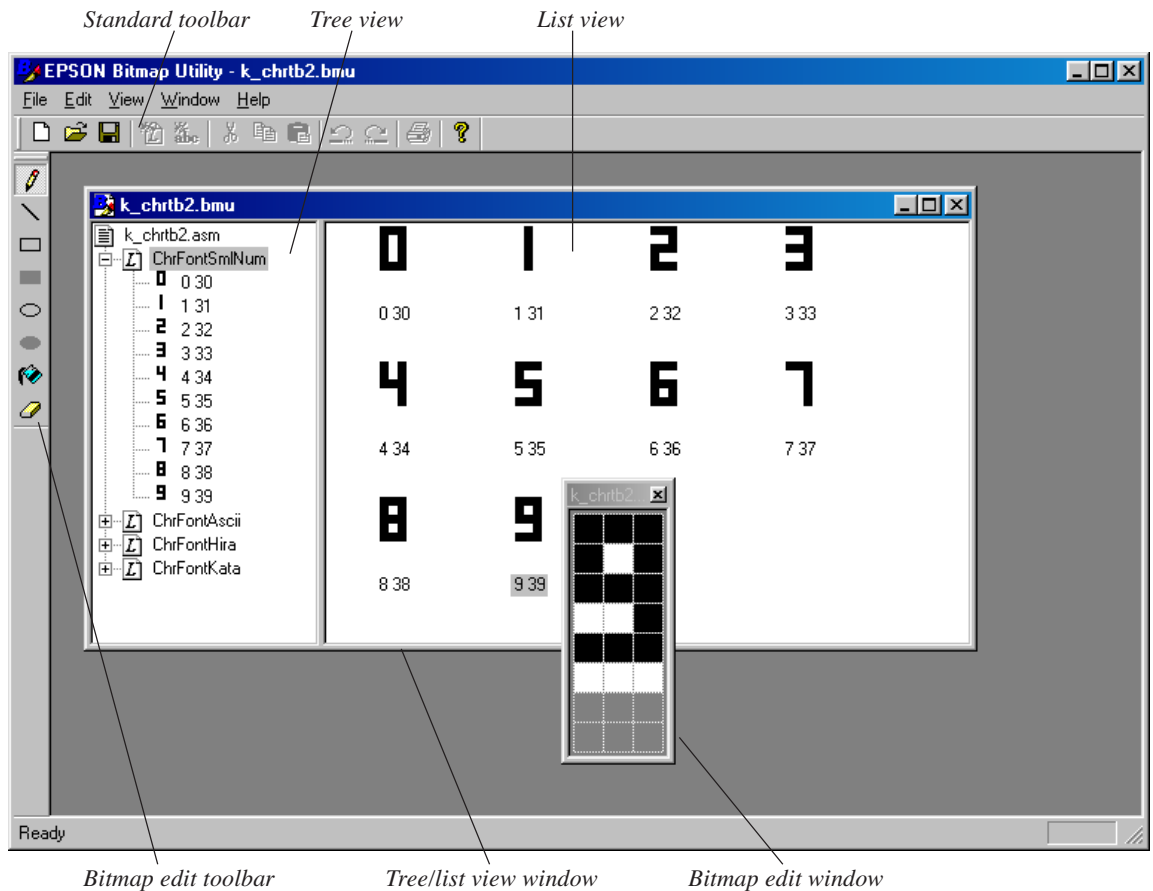
BmpUtil.exe

Double-click this icon to start BmpUtil.

To exit BmpUtil, select [Exit] from the [File] menu.

6.4 Window

The configuration of the BmpUtil windows is shown below:



Tree/list view window

This window is provided for each assembly list file. This window appears when you create a new file or read a bitmap definition file. You can open multiple files for editing. This window is divided into two parts: the tree view pane and list view pane. You can resize these areas by dragging their boundaries.

Tree view

This area displays the tree structure for three hierarchical levels, namely, the name of the assembly source file to be output, its labels, and the data defined in each label. You can use this area to select desired items, to rename a file or label, and to add new or delete current labels or data.

List view

This area displays the list of data defined in the label currently selected in tree view. The name of data is used as the comment added to each piece of data when it is output to the assembly source file. Use this area to select desired data, to rename the data, and to add new or delete current data.

Bitmap edit window

This window is displayed when you double-click a bitmap icon within list view. Use this window to create a new or edit the current bitmap.

Toolbars

Two toolbars, the standard and the bitmap edit toolbars, are provided. The standard toolbar contains commands on the [File] and [Edit] menus, while the bitmap edit toolbar is used to edit bitmap images. The bitmap edit toolbar becomes active when you open a bitmap edit window.

Basic window operations

Closing and activating the window

To close the window, click the [Close] button on the window. The windows being opened are listed on the [Window] menu. Select a window name from this menu to activate the corresponding window, or click anywhere in the desired window. You can also use the [Ctrl] + [Tab] keys to switch the active window from one window to the next. (This window switching technique works only in tree/list view windows.)

Resizing and moving the window

Drag the window boundaries to resize any window. The [Minimize] and [Maximize] buttons work in the same way as in other Windows applications. You can reposition any window within the display by dragging the title bar, but each window can be resized and moved only within the application window. A scroll bar is displayed if the image displayed in a window would otherwise extend beyond the window in any direction.

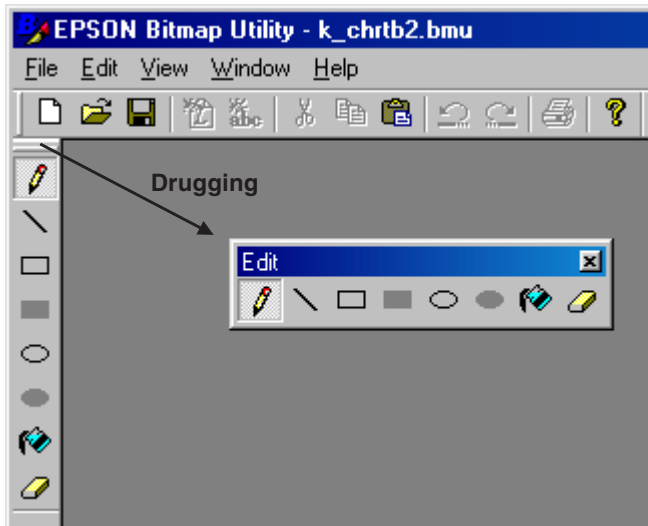
Aligning windows

To align open windows, select [Cascade] or [Tile] from the [Window] menu. (This method of aligning windows works only in tree/list view windows.)

Moving toolbars

To move a toolbar, drag the its heading. Drag the toolbar to the top of the window to arrange toolbar icons horizontally. Drag the toolbar to the extreme left or right of the window to align the toolbar icons vertically. If you drag the toolbar anywhere else in the window (or if you double-click on the heading, the buttons form a floating toolbar.

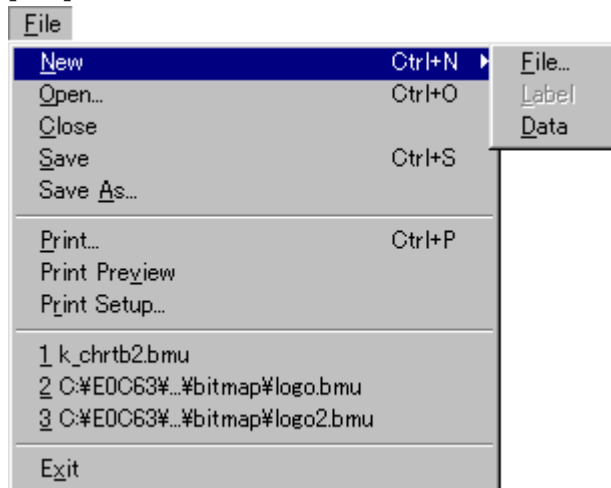
You can hide the toolbars by selecting a command from the [View] menu.



6.5 Menus and Toolbar

6.5.1 Menus

[File] menu



[New] ([Ctrl] + [N])

When the tree/list view window is open

[File...]

Creates a new bitmap definition file and displays a wizard to set new data type and size. A tree/list view window is open when the setting is done.

[Label]

Creates a new label. This option becomes active when the assembly source file name is selected in the tree view.

[Data]

Creates a new bitmap pattern. This option becomes active when one of the labels is selected in the tree view.

[New...] ([Ctrl] + [N]) When the tree/list view window is not open

Creates a new bitmap definition file and displays a wizard to set new data type and size. A tree/list view window is open when the setting is done.

[Open...] ([Ctrl] + [O])

Opens a new bitmap definition file (.bmu).

[Close]

Closes the currently active tree/list view window. This command works in the same way as the [Close] button in the window. If you made changes to the data, this command displays a dialog box that prompts you to save your changes.

[Save] (Ctrl) + [S]

Saves the contents of the active tree/list view window to the bitmap definition file (.bmu) and assembly source file, overwriting the previous files.

[Save As...]

Saves the contents of the active tree/list view window to the bitmap definition file (.bmu) under another name.

[Print...] ([Ctrl] + [P])

Prints all bitmap images, defined in the active tree/list view window, in a list.

[Print Preview]

Displays a print image.

[Print Setup...]

Displays a dialog box to select paper size and printer.

File list

Recently opened files are listed here. You may open a file by selecting from this list.

[Exit]

Exits BmpUtil.

[Edit] menu



[Undo] ([Ctrl] + [Z])

Cancels up to four of the most-recent editing actions performed within the bitmap edit window.

[Redo] ([Ctrl] + [Y])

Executes up to four of most-recent editing actions canceled with [Undo].

[Cut] ([Ctrl] + [X])

Cuts and copies a selected portion to the clipboard.

[Copy] ([Ctrl] + [C])

Copies a selected portion to the clipboard.

[Paste] ([Ctrl] + [V])

Pastes the portion copied to the clipboard.

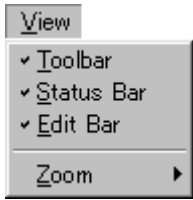
[Delete] ([Ctrl] + [D])

Deletes a selected item.

[Rename]

Renames an item selected in the tree/list view window.

[View] menu



[Toolbar]

Alternately shows or hides the toolbar as you select this menu command.

[Status bar]

Alternately shows or hides the status bar as you select this menu command.

[Edit Bar]

Alternately shows or hides the bitmap edit toolbar as you select this menu command.

[Zoom]

Zooms the dot display size of the bitmap edit window.

[Window] menu



[Cascade]

Aligns the open tree/list view windows, overlapping them diagonally.

[Tile Horizontally]

Aligns the open tree/list view windows by placing them horizontally.

[Tile Vertically]

Aligns the open tree/list view windows by placing them vertically.

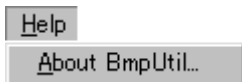
[Arrange Icons]

Arranges minimized tree/list view window icons at the lower space of the application window.

Window list

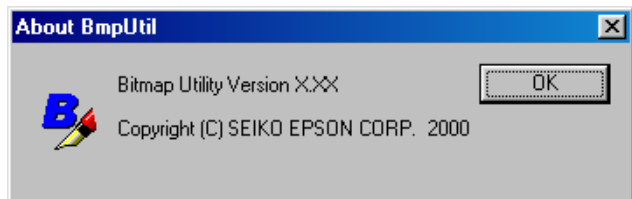
Lists the currently open tree/list view windows. Select a window to activate the corresponding tree/list view window.

[Help] menu

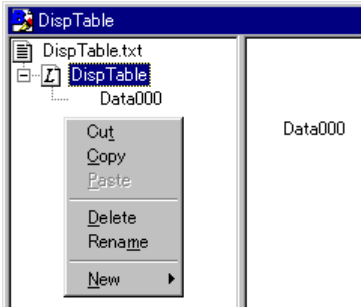


[About BmpUtil...]

Displays the version information for BmpUtil.



Pop-up menu



To display a pop-up menu, right-click with the mouse pointer positioned within the tree view or list view pane. Depending on the item selected, different commands in the menu become active. Commands in the menu work the same way as the others already mentioned.

6.5.2 Standard Toolbar Buttons



[New File] button

Creates a new bitmap definition file and displays a wizard to set new data type and size. A tree/list view window is open when the setting is done.



[Open] button

Opens a bitmap definition file (.bmu).



[Save] button

Saves the contents of the active tree/list view window to the bitmap definition file (.bmu) and assembly source file, overwriting the previous files.



[New Label] button

Creates a new label. This button becomes active when the assembly source file name is selected in the tree view.



[New Data] button

Creates a new bitmap pattern. This button becomes active when one of the labels is selected in the tree view.



[Cut] button

Cuts and copies a selected portion to the clipboard.



[Copy] button

Copies a selected portion to the clipboard.



[Paste] button

Pastes the portion copied to the clipboard.



[Undo] button

Cancels up to four of the most recent editing actions performed within the bitmap edit window.



[Redo] button

Executes up to four of the most recent editing actions canceled with [Undo].



[Print] button

Prints all bitmap images defined in the active tree/list view window in a list.



[About] button

Displays version information for BmpUtil.

6.5.3 Bitmap Edit Toolbar Buttons

Use these buttons to create new bitmap images or to correct current bitmap images. They become active when the bitmap edit window is open.



[Pen] button

Sets or resets dots to white or black.



[Line] button

Traces a straight line.



[Rectangle] button

Draws a rectangular frame.



[Filled Rectangle] button

Draws a filled rectangle.



[Ellipse] button

Draws an elliptical frame.



[Filled Ellipse] button

Draws a filled ellipse.



[Fill] button

Fills an area in which all dots are the same color with the selected color.



[Erase] button

Erases a selected area by filling it with white.


6.6 Creating Bitmap Data

This section explains how to create a bitmap.

6.6.1 New Data Wizard

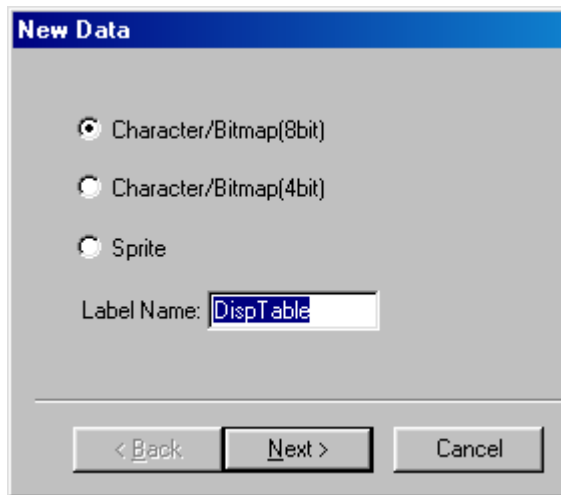
Starting the new data wizard

To create new bitmap data, select [New...] (or [New-File]*) from the [File] menu or click the [New File] button. A [New Data] dialog box appears.

 [New File] button

* The [File] menu shows [New...] when a tree/list view window is not open. The same menu shows [New], indicating that there is a subordinate menu, when tree/list view window is open.

[New Data] dialog box



Use this dialog box to select the type of data you want to create and to specify a label.

Data type

Select [Character/Bitmap(4bit)] when creating bitmap data for the S1C63 Family.

For the S1C88 Family, either [Character/Bitmap(8bit)] or [Sprite] can be selected.

Character/Bitmap

Select this option to create a bitmap image for the dot matrix LCD or font data for the character generator. Be sure to select "(4bit)" for the S1C63 Family or "(8bit)" for the S1C88 Family.

Sprite

Select this option to create a sprite data of 16 (16 dots pixels consisting of bitmap and clear planes.

The data thus created can only be used with the S1C88 Family models equipped with sprite-capable display controllers.

Label

Type a label in the [Label Name] text box. This is the label that will be used by the application program to access data. Since the label will be written in the assembly source file to be output, be sure to use characters and symbols that the assembler can identify. Avoid exceeding the maximum number of characters provided for the assembler.

Note that the entered name will also be used as the assembly source file name.

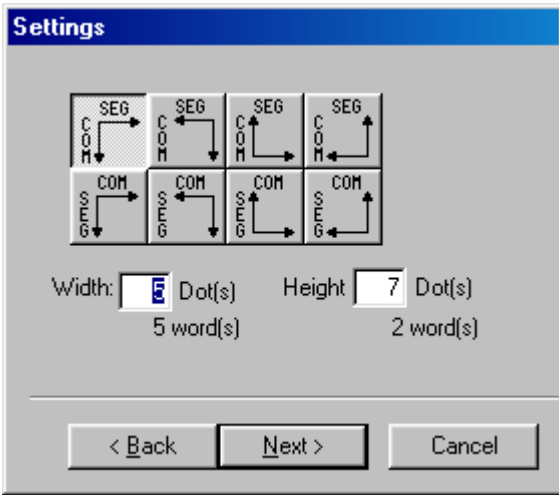
Each label or assembly source file name specified in this dialog box can be changed later in tree view. You can add labels at a later point. In other words, you can define multiple labels for a single assembly source file.

Click the [Next>] button after making the required selections.

If you select [Sprite], you do not need to specify any other information, such as bitmap size. Click the [Finish] button — which appears in place of the [Next>] button — to exit the New Data wizard. The tree/list view pane is displayed.

[Settings] dialog box (when [Character/Bitmap] is selected)

Select [Character/Bitmap] and click the [Next>] button in the [New Data] dialog box to display the [Settings] dialog box. If you change your mind and wish to select [Sprite], click the [<Back] button to return to the [New Data] dialog box.



(Example for the S1C63 Family)

Use this dialog box to specify the bitmap image size and COM/SEG layout.

COM/SEG layout

Use one of the eight buttons to select a COM/SEG layout according to the COM/SEG port assignment for the dot-matrix LCD so that the created bitmap is properly oriented when displayed. Keep in mind that you cannot change this option once the data is created.

Bitmap image size

Type a horizontal and vertical number of dots for the bitmap image respectively in the [Width] and [Height] text boxes.

Under each text box, you can find the number of words required in each direction. Multiply these numbers to calculate the number of words required for each piece of data.

Example of settings

Figure 6.6.1.1 shows an example of creating character generator data of 5 × 7 dots. (default settings)

COM/SEG layout: Width: 5 dots, Height: 7 dots

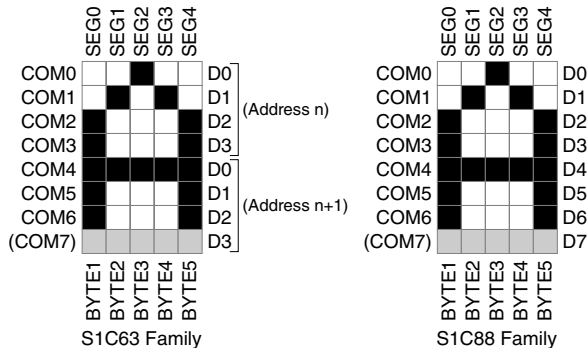


Fig. 6.6.1.1 Example of bitmap image size settings

The COM and SEG numbers that appear in the example merely indicate that an arrangement in ascending order in each of the directions indicated by arrows on the selected button. These numbers can vary, depending on the display memory addresses to which data is actually written.

The data created is assigned to data bits by ascending order of COM numbers.

In the S1C63 Family, since data is assigned in the COM direction in units of four bits, some data bits will become invalid if you specify a number of dots that is not a multiple of four for this direction. In the example, the data is assigned to COM0 through 6 while D3 corresponding to COM7 is masked.

In the S1C88 Family, since data is assigned in the COM direction in units of eight bits, some data bits will become invalid if you specify a number of dots that is not a multiple of eight for this direction. In the example, the data is assigned to COM0 through 6 while D7 corresponding to COM7 is masked.

You can change this assignment to mask the D0 corresponding to COM0 using the [Offset] dialog box, shown next.

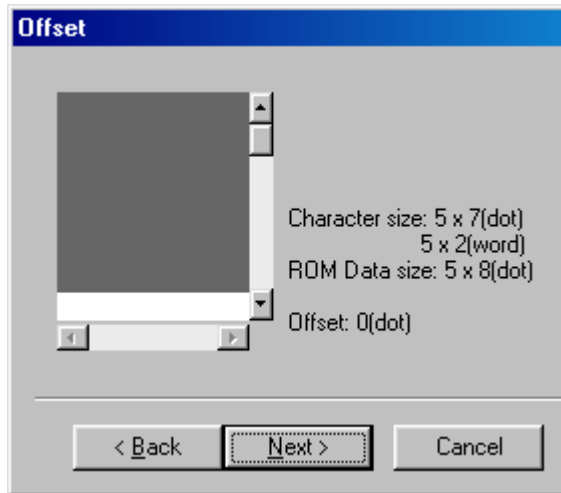
Click the [Next>] button after making the required selections.

Depending on the size you selected, no more dialog boxes may be displayed. In this case, the [Finish] button appears in place of the [Next>] button. Click the [Finish] button to exit the New Data wizard. The tree/list view window is displayed.

[Offset] dialog box (when [Character/Bitmap] is selected)

The [Offset] dialog box appears if you specify a number of dots in the [Settings] dialog box that is not a multiple of four (S1C63 Family) or eight (S1C88 Family) for the COM direction. Click the [<Back] button to return to the [Settings] dialog box.

Note: The [Offset] dialog box does not appear when you specify a number of dots that is a multiple of four (eight) for the COM direction.



(Example for the S1C63 Family)

Use this dialog box to specify the data offset value in the COM direction.

Because display data is assigned in the COM direction in units of four (eight) dots, some COM output data is not used if you specify a number of dots not a multiple of four (eight) for this direction. By default, bitmap assignment starts with COM0. For this reason, the data to be assigned to the largest COM number will be invalid.

Data will be assigned to the gray area within the box enclosed by the scroll bars, while the white area remains unused. Use the scroll bars to move the unused area to COM0. The [Offset] value changes when you move the unused area.

Suppose, for example, that you create data of 5 < 7 dots as described in the previous example. The data arrangement then changes as follows:

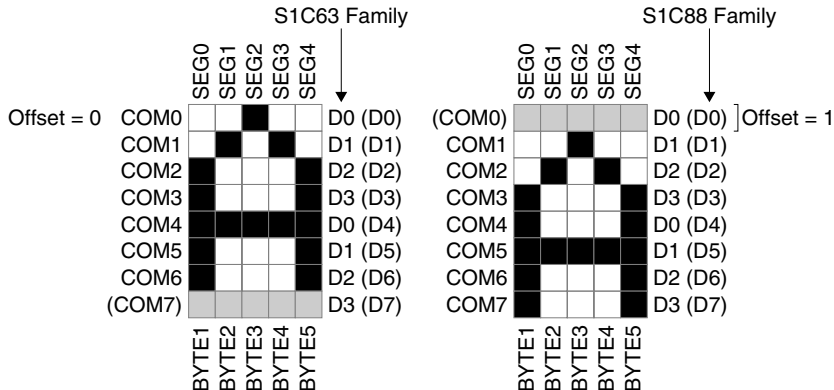


Fig. 6.6.1.2 Example of offset setting

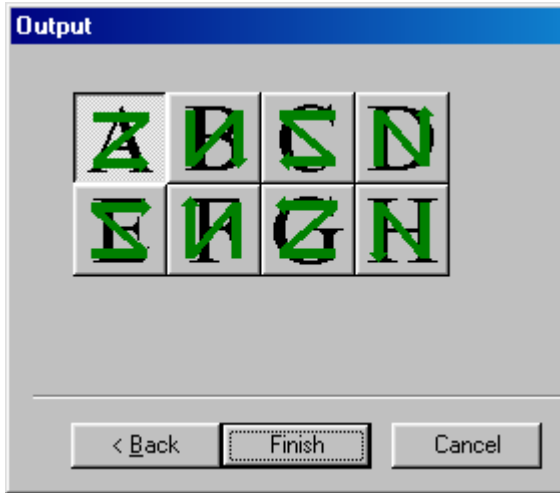
This setting creates final data with 0 selected as the unused bit.

Click the [Next>] button after specifying an offset value.

[Output] dialog box (when [Character/Bitmap] is selected)

The [Output] dialog box appears following the [Settings] or [Offset] dialog box if you specify five dots or more (S1C63 Family) or nine dots or more (S1C88 Family) for the COM direction in the [Settings] dialog box. Click the [<Back] button to return to the previous dialog box.

Note: The [Output] dialog box does not appear if you specify four (eight) or fewer dots for the COM direction.



Use this dialog box to select one data output pattern sequence from the eight options (patterns A to H). Data is output to the assembly source file in the sequence shown on the selected button.

Figure 6.6.1.3 shows an example in which bitmap data of 16 × 16 dots is output.

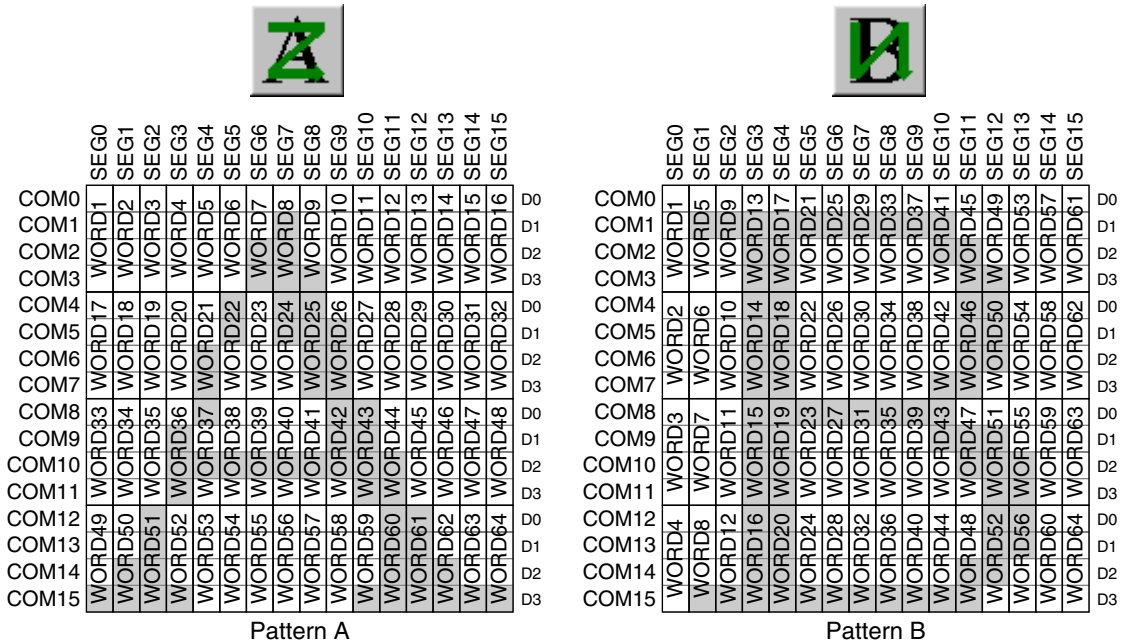


Fig. 6.6.1.3(a) Data output pattern (S1C63 Family)

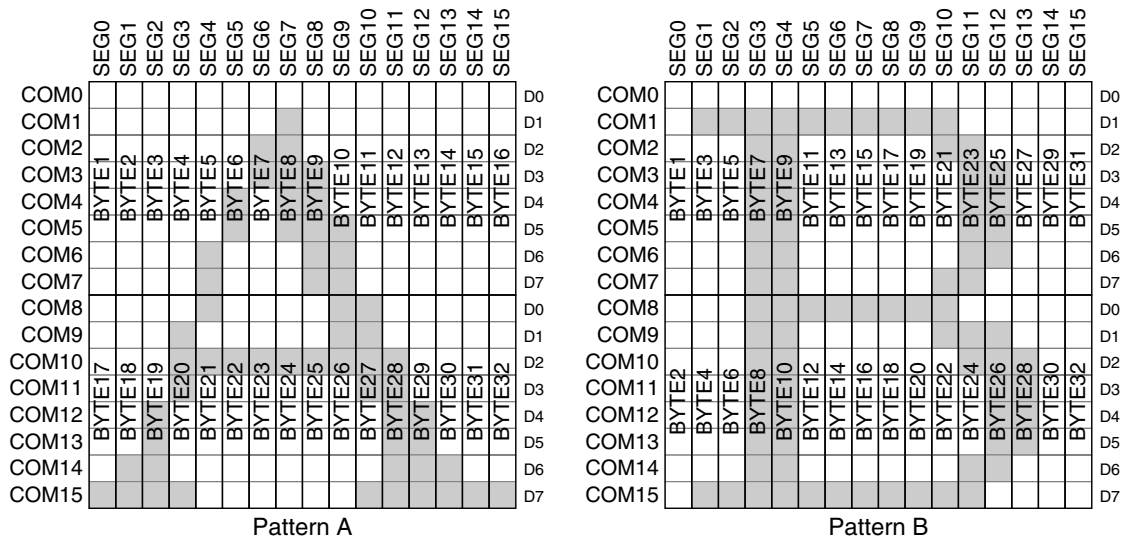


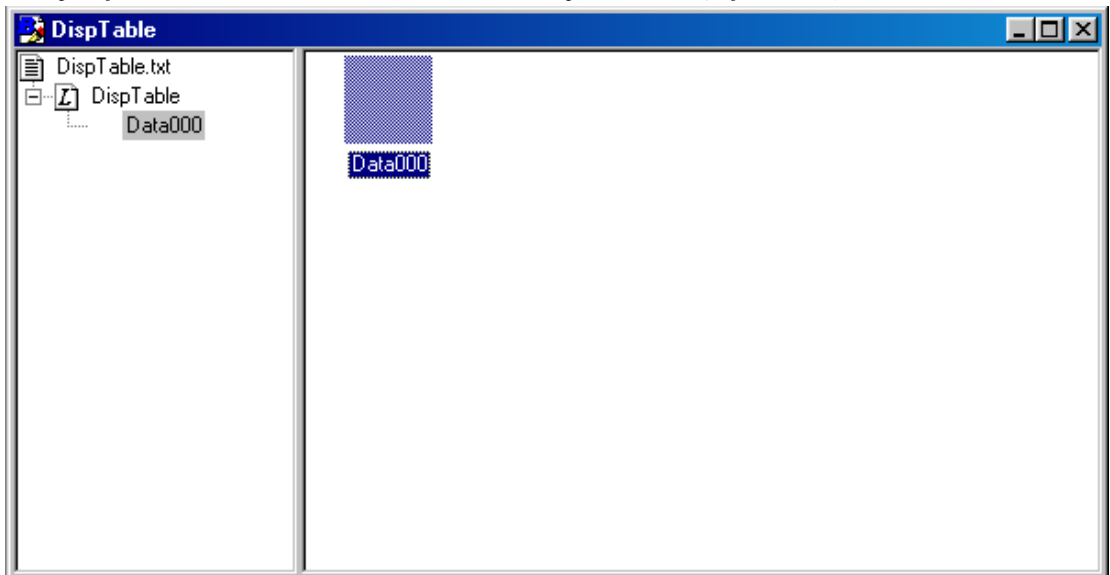
Fig. 6.6.1.3(b) Data output pattern (S1C88 Family)

After selecting a data output pattern, click the [Finish] button to exit the New Data wizard. The tree/list view window is displayed.

Tree/list view window displayed when new data is created

Exiting the New Data wizard displays the tree/list view window shown below.

Example of tree/list view window when [Character/Bitmap] is selected (default label)

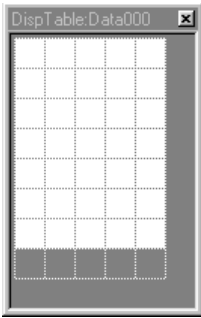


The assembly source file and label appear in the list view with the names specified in the [New Data] dialog box. Under the label, you can see that new data (blank data) has been created with the name of Data000. Display the data by double-clicking the label. See Section 6.6.3, "Editing Functions", for information on changing these names or adding and deleting labels and data.

6.6.2 Creating Bitmap Images

This section explains how to create bitmap images.

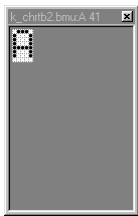
Bitmap edit window



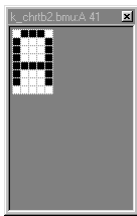
A bitmap edit window is displayed when you double-click a bitmap icon in list view. You can use this window to create and edit bitmap images. Creating a new bitmap image displays a blank bitmap icon. In this case, double-click a desired position above the data name.

The gray area within the window cannot be edited. When you create a new bitmap image, the area that can be edited appears white (dots OFF).

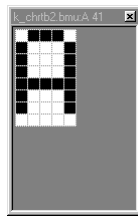
By default, the bitmap initially appears with its dots enlarged at the maximum ratio (500%). You can change sizes using [Zoom] in the [View] menu.



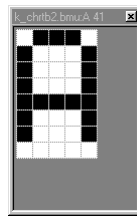
100%



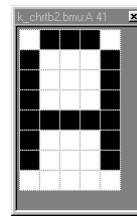
200%



300%



400%



500% (default)

When you place the mouse pointer (which turns into the button selected from the bitmap edit toolbar) anywhere in the bitmap edit window, the pointer position (coordinates with the 0 position set at the upper left corner of the bitmap) is displayed in the status bar in X \ Y format.



After creating a bitmap image using the editing tools described in a later section, click the window [Close] button to complete your bitmap edit task.

Bitmap editing tools

Use the following buttons in the bitmap edit toolbar to edit bitmap images:



[Pen] button

If you click the left mouse button a dot is black. And if you click the right mouse button a dot is white. By dragging the mouse the trace can be erased or comes alive.



[Line] button

Traces a line connecting the start and end points when you drag the mouse. Black is selected as the line color when you use the left mouse button. White is selected as the line color when you use the right mouse button.



[Rectangle] button

Draws a rectangular frame with the start and end points set as two corners of a frame, diagonally opposite each other, when you drag the mouse. Black is selected as the frame color when you use the left mouse button. White is selected as the frame color when you use the right mouse button.



[Filled Rectangle] button

Draws a filled rectangle with the start and end points set as two corners of a rectangle, diagonally opposite each other, when you drag the mouse. Black is selected as the border and fill color when you use the left mouse button. White is selected as the border and fill color when you use the right mouse button.



[Ellipse] button

Draws an elliptical frame passing through the start and end points when you drag the mouse. Black is selected as the frame color when you use the left mouse button. White is selected as the frame color when you use the right mouse button.



[Filled Ellipse] button

Draws a filled ellipse whose border passes through the start and end points when you drag the mouse. Black is selected as the border and fill color when you use the left mouse button. White is selected as the border and fill color when you use the right mouse button.



[Fill] button

Fills an area in which all dots are the same color with a selected color when you click one of the dots in that area. Black is selected as the fill color when you use the left mouse button. White is selected as the fill color when you use the right mouse button.



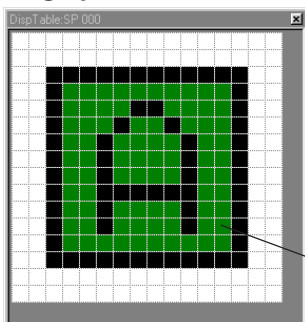
[Erase] button

Fills a rectangular area (with the start and end points set as two corners of the rectangle that are diagonally opposite each other), drawn by dragging the mouse, with white.

All dragging operations other than that for the [Fill] button can be canceled by pressing the [ESC] key before you release the mouse button (as you drag the mouse).

You can also cancel up to four of your most recent actions by selecting [Undo] or [Redo] from the [Edit] menu or the [Undo] or [Redo] button from the standard toolbar.

Editing sprite data



Use the above-mentioned editing tools to edit bitmap images for sprite data. When editing sprite data, you can also set transparent dots. To set transparent dots, first select the desired tool from the above-mentioned editing tools, then enable dots (using the left mouse button) while holding down the [Ctrl] key. Dots specified as transparent dots appear green. To cancel transparent dots, disable dots (using the right mouse button) while holding down the [Ctrl] key.

(Green) clear dots

6.6.3 Editing Functions

This section describes editing functions other than the ones for bitmap image creation described in the previous section.

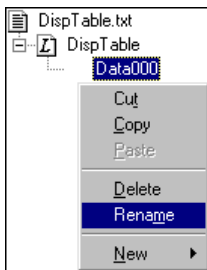
Saving and read files

After creating data, save the data to a file by selecting [Save] or [Save As...] from the [File] menu (or by clicking the [Save] button), as in other Windows applications. BmpUtil outputs a bitmap definition file (.bmu) and assembly source file (name displayed in the tree view). To add data or to make changes to existing data, open the corresponding bitmap definition file by selecting [Open...] from the [File] menu (or by clicking the [Open] button). You cannot open an assembly source file.

Renaming a file

To rename a bitmap definition file (.bmu), select [Save As...] from the [File] menu and save the file under another name.

To rename the assembly source file, a label, or data, follow the steps given below.



1. Click a desired item.
2. Re-click the same item (as opposed to double-clicking the item — which does not have the desired effect.) You can also rename an item by selecting [Rename] from the [Edit] menu or from the pop-up menu which appears by right-clicking.

You can now rename the desired item. To change the entire name, simply begin typing. To change a part of the current name, select that part and type the new characters.



Notes:

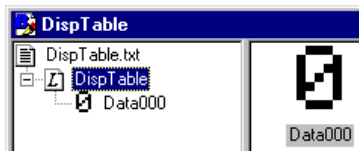
- If you rename the assembly source file, a new assembly source file is created under a new name after you save the first piece of data after the renaming of the assembly source file.

- Use assembler-readable characters for the assembly source file and label names. Avoid exceeding the maximum number of characters provided for the assembler.
- Data names are used merely as comments to identify data within the assembly source file to be output. They are not used to perform control operations within the program.

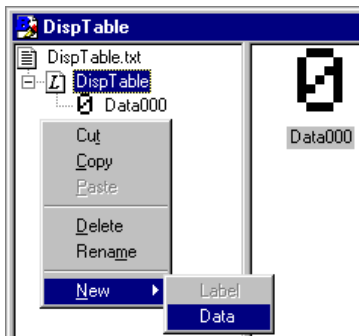
Adding, deleting, and copying data

Adding data

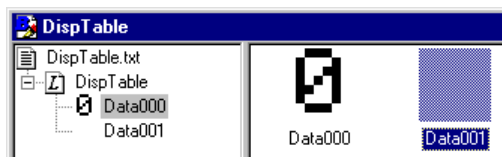
To add data, follow the steps given below.



1. Select a label to which to add data in the tree view. The list of data defined in the selected label is displayed in the list view.



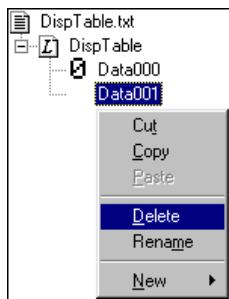
2. Select [New - Data] from the [File] menu, or right-click and select [New - Data] from the pop-up menu. New data is added to the tree view and the list view. Rename the data as necessary.



- * When creating new data based on existing data, you can add this new data by copying and pasting existing data.

Deleting data

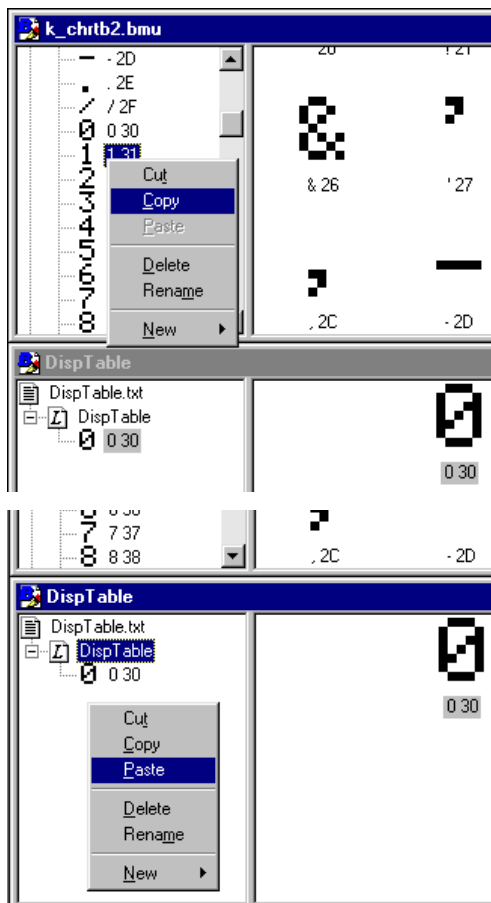
To delete data, follow the steps given below.



1. Select data you wish to delete from the list or the tree views. Use the [Shift] or [Ctrl] key to select multiple pieces of data.
2. Select [Delete] from the [Edit] menu, or right-click and select [Delete] from the pop-up menu. The selected data is deleted from the tree view and the list view.

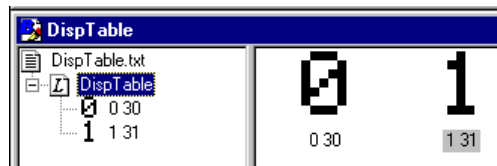
Cutting, copying, and pasting data

To cut (or copy) and paste data, follow the steps given below.



1. Select the data you wish to cut or copy from the list or the tree view. Use the [Shift] or [Ctrl] key to select multiple pieces of data.
2. Select [Cut] (or [Copy]) from the [Edit] menu, or right-click and select [Cut] (or [Copy]) from the pop-up menu. (You can also use the keyboard shortcut [Ctrl] + [X] or [Ctrl] + [C].)
3. Select a label to which to paste the data from the tree view. (You can also paste the data to a label in other window.) Then select [Paste] from the [Edit] menu, or right-click and select [Paste] from the pop-up menu. (You can also use the keyboard shortcut [Ctrl] + [V].)

The data is pasted to the tree and the list views.



If the data size in the destination label differs from the source data size, a confirmation dialog box appears to ask you whether you wish to continue. You can choose to cancel pasting. If you choose to continue and paste the data, blank dots are added to the right-hand and bottom sections of the pasted data if the data size in the destination label exceeds the source data size. If the data size in the destination label is less than the source data size, the pasted data is clipped.

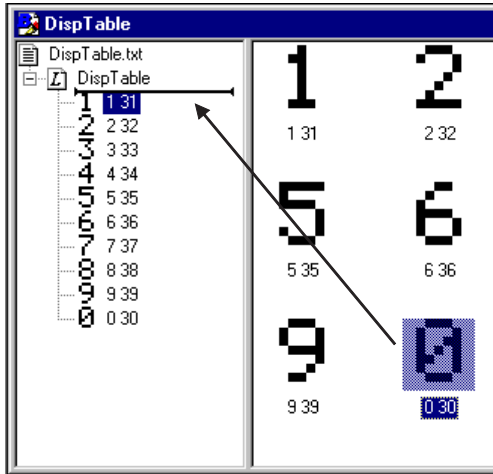
Since the data you cut or copy is saved in the clipboard, you can paste it repeatedly.

When you select a label and paste the data, the data is copied as the last piece of data in that label. To paste the data somewhere between the current pieces of data, select a desired piece of data. The data is pasted before the selected data.

Dragging and dropping data

You can move or copy data by dragging and dropping it, as follows.

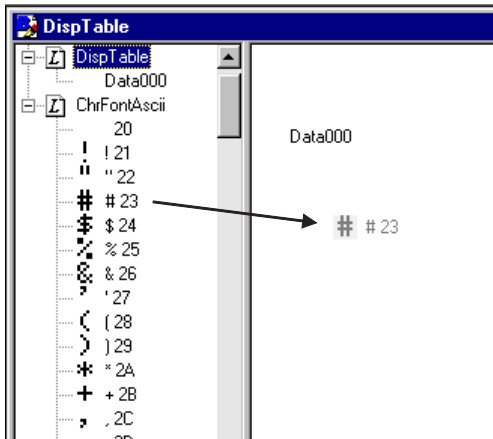
Moving data within the same label (rearranging data)



1. Select a desired data item from the tree view.
2. Drag and drop the data within the same label. As you drag the data, another pointer appears in tree view to indicate locations in tree view where you may position the data.

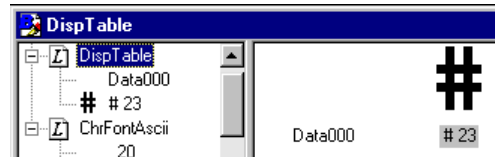
Note: Dropping the data within another label copies rather than moves the data. You are also copying the data if you drag it from the list or the tree view and drop it into another.

Copying data 1 (from the tree view to the list view)

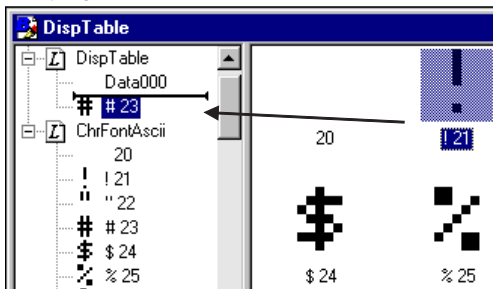


1. In the tree view, select a label to which to copy data. The list of data defined in the destination label is displayed in the list view.
2. Drag the desired data from the tree view to the list view. Drop it into the list view pane to copy the data.

Copying data in this way adds it as the last piece of data in the label.



Copying data 2 (from the list view to the tree view)



1. In the tree view, select a label from which to copy data. The list of data defined in that label is displayed in the list view.
2. Drag the desired data from the list view and drop it into the tree view. As you drag the data, another pointer appears in the tree view to indicate locations in the tree view pane where you can position the data.

Use this technique to copy data to a desired location.

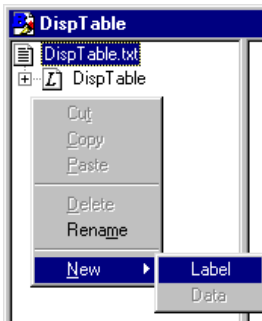
When using data copying technique 1 or 2, if the destination data size differs from the source data size, a confirmation dialog box will prompt you for confirmation to continue. You can choose to cancel copying. If you choose to continue and copy the data, blank dots are added to the right-hand and bottom sections of the copied data if the destination data size exceeds the source data size. If the destination data size is less than the source data size, the copied data is clipped.

When two or more windows are open, you can copy data between windows by dragging and dropping.

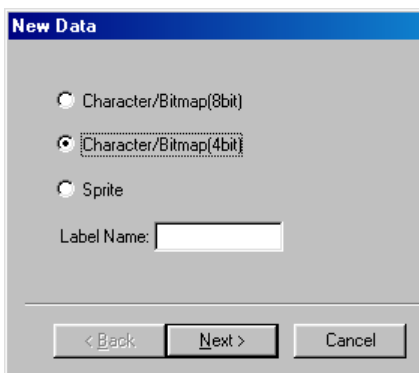
Adding, deleting, and copying a label

Adding a label

To add a label, follow the steps given below.



1. Click the file name displayed at the uppermost line of the tree view pane. No data is currently displayed in the list view.
2. Select [New - Label] from the [File] menu, or right-click and select [New - Label] from the pop-up menu.

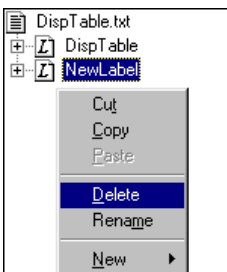


3. A wizard for creating new labels appears. You can use this wizard in the same way as the wizard described in "New Data Wizard" of Section 6.6.1. Specify a label name and select the data size and so on in this wizard. When you finish, a new label containing a single piece of blank data is created.

* When you create a new label based on an existing label (including defined data), you can add a label by copying and pasting the existing label.

Deleting a label

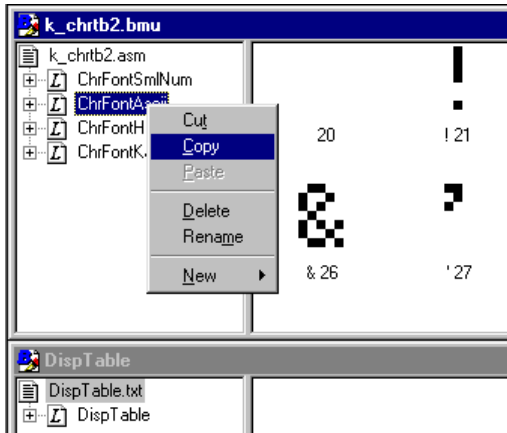
To delete a label, follow the steps given below.



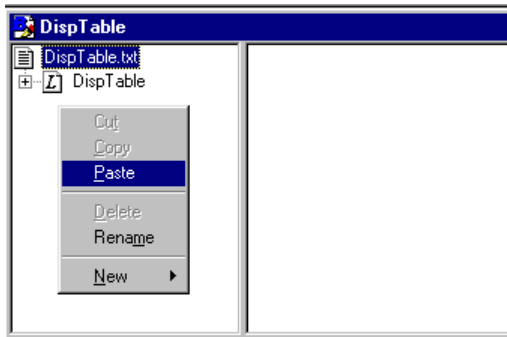
1. In the tree view, select a label you wish to delete.
2. Select [Delete] from the [Edit] menu, or right-click and select [Delete] from the pop-up menu.
The selected label and all data defined in the label are deleted from the tree view pane.

Cutting, copying and pasting a label

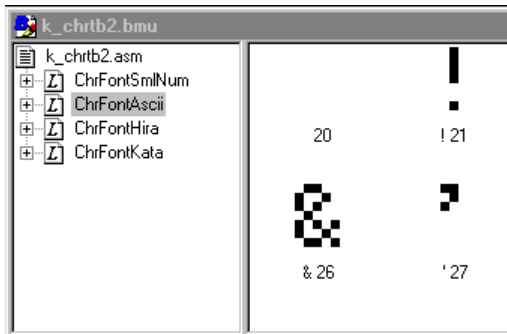
You can cut (or copy) and paste a label, including data defined in that label. To do this, follow the steps given below.



1. In the tree view, select a label you wish to cut or copy.
2. Select [Cut] (or [Copy]) from the [Edit] menu, or right-click and select [Cut] (or [Copy]) from the pop-up menu. (You can also use the keyboard shortcut [Ctrl] + [X] or [Ctrl] + [C].)

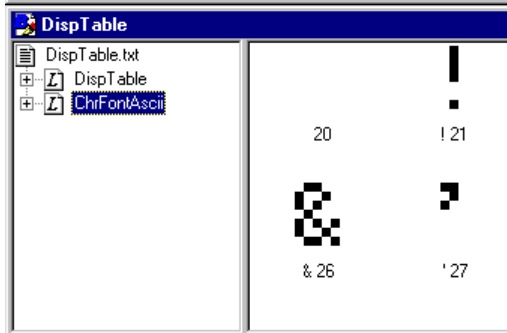


3. In the tree view, select the assembly source file name. (You can also select file names in other windows.) Then select [Paste] from the [Edit] menu, or right-click and select [Paste] from the pop-up menu. (You can also use the keyboard shortcut [Ctrl] + [V].)



The label is pasted at the end of the tree view. The list of data defined in the label is displayed in the list view.

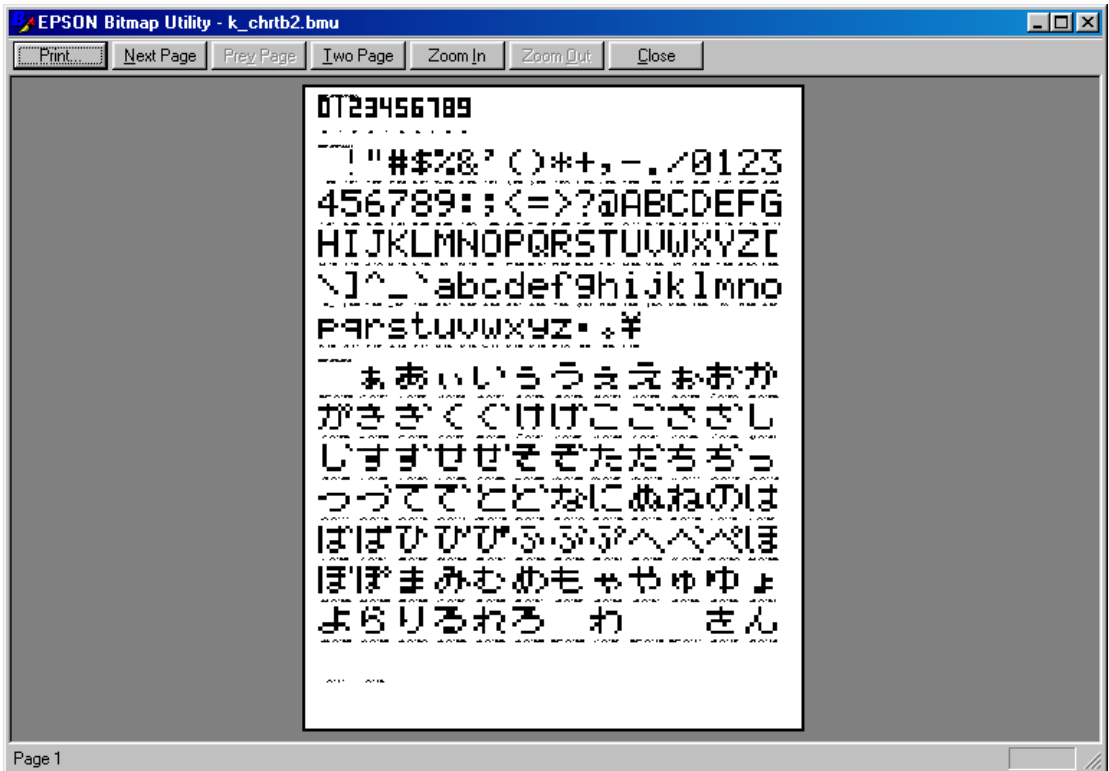
Since the cut or copied label and data are saved in the clipboard, you can paste them repeatedly.



Printing data

Print the created data by selecting [Print] from the [File] menu, or by clicking the [Print] button in the toolbar.

All data defined in the open file are printed. You can view the print image by selecting [Print Preview] from the [File] menu.



6.7 Assembly Source File

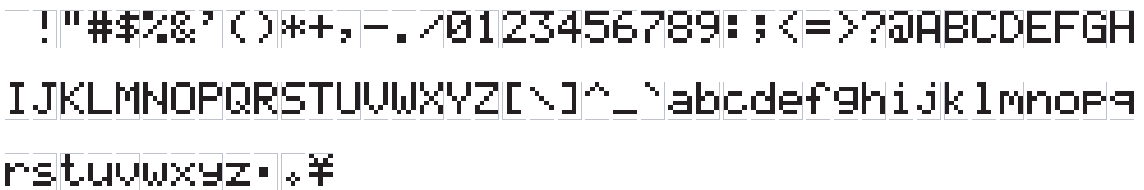
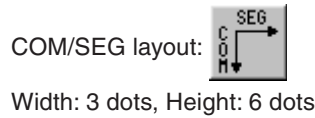
This section shows an example of an assembly source file. This file contains specified labels, each followed by bitmap data (in byte unit) arranged in the specified output sequence. The name assigned to each piece of data is used as a comment. Load the created file into the user program or link it to the program together with other objects after assembling.

Example for S1C63 Family)

0123456789

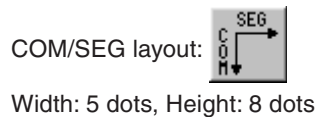
```

ChrFontSmlNum:
    .word 0Fh,01h,0Fh ; 0 30
    .word 01h,01h,01h
    .word 00h,0Fh,00h ; 1 31
    .word 00h,01h,00h
    .word 0Dh,05h,07h ; 2 32
    .word 01h,01h,01h
    .word 05h,05h,0Fh ; 3 33
    .word 01h,01h,01h
    .word 07h,04h,0Fh ; 4 34
    .word 00h,00h,01h
    .word 07h,05h,0Dh ; 5 35
    .word 01h,01h,01h
    .word 0Fh,05h,0Dh ; 6 36
    .word 01h,01h,01h
    .word 01h,01h,0Fh ; 7 37
    .word 00h,00h,01h
    .word 0Fh,05h,0Fh ; 8 38
    .word 01h,01h,01h
    .word 07h,05h,0Fh ; 9 39
    .word 01h,01h,01h
    
```



```

ChrFontAscii:
    .word 00h,00h,00h,00h,00h ; 20
    .word 00h,00h,00h,00h,00h
    .word 00h,00h,0Fh,00h,00h ; ! 21
    .word 00h,00h,04h,00h,00h
    .word 00h,07h,00h,07h,00h ; " 21
    .word 00h,00h,00h,00h,00h
    .word 04h,0Fh,04h,0Fh,04h ; # 23
    .word 01h,07h,01h,07h,01h
    .word 04h,0Ah,0Fh,0Ah,02h ; $ 24
    .word 02h,02h,07h,02h,01h
    .word 03h,03h,08h,04h,02h ; % 25
    .word 02h,01h,00h,06h,06h
    .word 06h,09h,05h,02h,00h ; & 26
    .word 03h,04h,05h,02h,05h
    .word 00h,05h,03h,00h,00h ; ' 27
    .word 00h,00h,00h,00h,00h
    .word 00h,0Ch,02h,01h,00h ; ( 28
    .word 00h,01h,02h,04h,00h
    .word 00h,01h,02h,0Ch,00h ; ) 29
    .word 00h,04h,02h,01h,00h
    .word 04h,08h,0Eh,08h,04h ; * 2A
    .word 01h,00h,03h,00h,01h
    .word 08h,08h,0Eh,08h,08h ; + 2B
    .word 00h,00h,03h,00h,00h
    .word 00h,00h,00h,00h,00h ; , 2C
    .word 00h,05h,03h,00h,00h
    .word 08h,08h,08h,08h,08h ; - 2D
    .word 00h,00h,00h,00h,00h
    
```



.word 00h,00h,00h,00h,00h	; . 2E	.word 00h,00h,00h,00h,00h	; v 56
.word 00h,06h,06h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 00h,00h,08h,04h,02h	; / 2F	.word 00h,00h,00h,00h,00h	; w 57
.word 02h,01h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,09h,05h,0Eh	; 0 30	.word 00h,00h,00h,00h,00h	; x 58
.word 03h,05h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 00h,02h,0Fh,00h,00h	; 1 31	.word 00h,00h,00h,00h,00h	; y 59
.word 00h,04h,07h,04h,00h		.word 00h,00h,00h,00h,00h	
.word 02h,01h,01h,09h,06h	; 2 32	.word 00h,00h,00h,00h,00h	; z 5A
.word 04h,06h,05h,04h,04h		.word 00h,00h,00h,00h,00h	
.word 01h,01h,05h,0Bh,01h	; 3 33	.word 00h,00h,00h,00h,00h	; [5B
.word 02h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 08h,04h,02h,0Fh,00h	; 4 34	.word 00h,00h,00h,00h,00h	; \ 5C
.word 01h,01h,01h,07h,01h		.word 00h,00h,00h,00h,00h	
.word 07h,05h,05h,05h,09h	; 5 35	.word 00h,00h,00h,00h,00h	;] 5D
.word 02h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 0Ch,0Ah,09h,09h,00h	; 6 36	.word 00h,00h,00h,00h,00h	; ^ 5E
.word 03h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 01h,01h,09h,05h,03h	; 7 37	.word 00h,00h,00h,00h,00h	; _ 5F
.word 00h,07h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 06h,09h,09h,09h,06h	; 8 38	.word 00h,00h,00h,00h,00h	; ' 60
.word 03h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 06h,09h,09h,09h,0Eh	; 9 39	.word 00h,00h,00h,00h,00h	; a 61
.word 00h,04h,04h,02h,01h		.word 00h,00h,00h,00h,00h	
.word 00h,06h,06h,00h,00h	; : 3A	.word 00h,00h,00h,00h,00h	; b 62
.word 00h,03h,03h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 00h,06h,06h,00h,00h	; ; 3B	.word 00h,00h,00h,00h,00h	; c 63
.word 00h,05h,03h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 08h,04h,02h,01h,00h	; < 3C	.word 00h,00h,00h,00h,00h	; d 64
.word 00h,01h,02h,04h,00h		.word 00h,00h,00h,00h,00h	
.word 04h,04h,04h,04h,04h	; = 3D	.word 00h,00h,00h,00h,00h	; e 65
.word 01h,01h,01h,01h,01h		.word 00h,00h,00h,00h,00h	
.word 00h,01h,02h,04h,08h	; > 3E	.word 00h,00h,00h,00h,00h	; f 66
.word 00h,04h,02h,01h,00h		.word 00h,00h,00h,00h,00h	
.word 02h,01h,01h,09h,06h	; ? 3F	.word 00h,00h,00h,00h,00h	; g 67
.word 00h,00h,05h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 02h,09h,09h,01h,0Eh	; @ 40	.word 00h,00h,00h,00h,00h	; h 68
.word 03h,04h,07h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,01h,01h,0Eh	; A 41	.word 00h,00h,00h,00h,00h	; i 69
.word 07h,01h,01h,01h,07h		.word 00h,00h,00h,00h,00h	
.word 0Fh,09h,09h,09h,06h	; B 42	.word 00h,00h,00h,00h,00h	; j 6A
.word 07h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,01h,01h,02h	; C 43	.word 00h,00h,00h,00h,00h	; k 6B
.word 03h,04h,04h,04h,02h		.word 00h,00h,00h,00h,00h	
.word 0Fh,01h,01h,02h,0Ch	; D 44	.word 00h,00h,00h,00h,00h	; l 6C
.word 07h,04h,04h,02h,01h		.word 00h,00h,00h,00h,00h	
.word 0Fh,09h,09h,09h,01h	; E 45	.word 00h,00h,00h,00h,00h	; m 6D
.word 07h,04h,04h,04h,04h		.word 00h,00h,00h,00h,00h	
.word 0Fh,09h,09h,09h,01h	; F 46	.word 00h,00h,00h,00h,00h	; n 6E
.word 07h,00h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,09h,09h,0Ah	; G 47	.word 00h,00h,00h,00h,00h	; o 6F
.word 03h,04h,04h,04h,07h		.word 00h,00h,00h,00h,00h	
.word 0Fh,08h,08h,08h,0Fh	; H 48	.word 00h,00h,00h,00h,00h	; p 70
.word 07h,00h,00h,00h,07h		.word 00h,00h,00h,00h,00h	
.word 00h,01h,0Fh,01h,00h	; I 49	.word 00h,00h,00h,00h,00h	; q 71
.word 00h,04h,07h,04h,00h		.word 00h,00h,00h,00h,00h	
.word 00h,00h,01h,0Fh,01h	; J 4A	.word 00h,00h,00h,00h,00h	; r 72
.word 02h,04h,04h,03h,00h		.word 00h,00h,00h,00h,00h	
.word 0Fh,08h,04h,02h,01h	; K 4B	.word 00h,00h,00h,00h,00h	; s 73
.word 07h,00h,01h,02h,04h		.word 00h,00h,00h,00h,00h	
.word 0Fh,00h,00h,00h,00h	; L 4C	.word 00h,00h,00h,00h,00h	; t 74
.word 07h,04h,04h,04h,04h		.word 00h,00h,00h,00h,00h	
.word 0Fh,02h,0Ch,02h,0Fh	; M 4D	.word 00h,00h,00h,00h,00h	; u 75
.word 07h,00h,00h,00h,07h		.word 00h,00h,00h,00h,00h	
.word 0Fh,04h,08h,00h,0Fh	; N 4E	.word 00h,00h,00h,00h,00h	; v 76
.word 07h,00h,00h,01h,07h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,01h,01h,0Eh	; O 4F	.word 00h,00h,00h,00h,00h	; w 77
.word 03h,04h,04h,04h,03h		.word 00h,00h,00h,00h,00h	
.word 0Fh,09h,09h,09h,06h	; P 50	.word 00h,00h,00h,00h,00h	; x 78
.word 07h,00h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 0Eh,01h,01h,01h,0Eh	; Q 51	.word 00h,00h,00h,00h,00h	; y 79
.word 03h,04h,05h,02h,05h		.word 00h,00h,00h,00h,00h	
.word 0Fh,09h,09h,09h,06h	; R 52	.word 00h,00h,00h,00h,00h	; z 7A
.word 07h,00h,01h,02h,04h		.word 00h,00h,00h,00h,00h	
.word 00h,00h,00h,00h,00h	; S 53	.word 00h,00h,00h,00h,00h	; · 7B
.word 00h,00h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 00h,00h,00h,00h,00h	; T 54	.word 00h,00h,00h,00h,00h	; □ 7C
.word 00h,00h,00h,00h,00h		.word 00h,00h,00h,00h,00h	
.word 00h,00h,00h,00h,00h	; U 55	.word 00h,00h,00h,00h,00h	; ¥ 7D
.word 00h,00h,00h,00h,00h		.word 00h,00h,00h,00h,00h	


6 BITMAP UTILITY

Example for S1C88 Family)

0123456789

ChrFontSmlNum:

```
DB 01Fh,011h,01Fh ; 0 30
DB 000h,01Fh,000h ; 1 31
DB 01Dh,015h,017h ; 2 32
DB 015h,015h,01Fh ; 3 33
DB 007h,004h,01Fh ; 4 34
DB 017h,015h,01Dh ; 5 35
DB 01Fh,015h,01Dh ; 6 36
DB 001h,001h,01Fh ; 7 37
DB 01Fh,015h,01Fh ; 8 38
DB 017h,015h,01Fh ; 9 39
```


COM/SEG layout: 

Width: 3 dots, Height: 6 dots

!"#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIH
 IJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
 rstuvwxyz.~

ChrFontAscii:

```
DB 000h,000h,000h,000h,000h; 20
DB 000h,000h,04Fh,000h,000h; ! 21
DB 000h,007h,000h,007h,000h; " 22
DB 014h,07Fh,014h,07Fh,014h; # 23
DB 024h,02Ah,07Fh,02Ah,012h; $ 24
DB 023h,013h,008h,064h,062h; % 25
DB 036h,049h,055h,022h,050h; & 26
DB 000h,005h,003h,000h,000h; ' 27
DB 000h,01Ch,022h,041h,000h; ( 28
DB 000h,041h,022h,01Ch,000h; ) 29
DB 014h,008h,03Eh,008h,014h; * 2A
DB 008h,008h,03Eh,008h,008h; + 2B
DB 000h,050h,030h,000h,000h; , 2C
DB 008h,008h,008h,008h,008h; - 2D
DB 000h,060h,060h,000h,000h; . 2E
DB 020h,010h,008h,004h,002h; / 2F
DB 03Eh,051h,049h,045h,03Eh; 0 30
DB 000h,042h,07Fh,040h,000h; 1 31
DB 042h,061h,051h,049h,046h; 2 32
DB 021h,041h,045h,04Bh,031h; 3 33
DB 018h,014h,012h,07Fh,010h; 4 34
DB 027h,045h,045h,045h,039h; 5 35
DB 03Ch,04Ah,049h,049h,030h; 6 36
DB 001h,071h,009h,005h,003h; 7 37
DB 036h,049h,049h,049h,036h; 8 38
DB 006h,049h,049h,029h,01Eh; 9 39
DB 000h,036h,036h,000h,000h; : 3A
DB 000h,056h,036h,000h,000h; ; 3B
DB 008h,014h,022h,041h,000h; < 3C
DB 014h,014h,014h,014h,014h; = 3D
DB 000h,041h,022h,014h,008h; > 3E
DB 002h,001h,051h,009h,006h; ? 3F
DB 032h,049h,079h,041h,03Eh; @ 40
DB 07Eh,011h,011h,011h,07Eh; A 41
DB 07Fh,049h,049h,049h,036h; B 42
DB 03Eh,041h,041h,041h,022h; C 43
DB 07Fh,041h,041h,022h,01Ch; D 44
DB 07Fh,049h,049h,049h,041h; E 45
```

COM/SEG layout: 

Width: 5 dots, Height: 8 dots

```

DB 07Fh,009h,009h,009h,001h ; F 46
DB 03Eh,041h,049h,049h,07Ah ; G 47
DB 07Fh,008h,008h,008h,07Fh ; H 48
DB 000h,041h,07Fh,041h,000h ; I 49
DB 020h,040h,041h,03Fh,001h ; J 4A
DB 07Fh,008h,014h,022h,041h ; K 4B
DB 07Fh,040h,040h,040h,040h ; L 4C
DB 07Fh,002h,00Ch,002h,07Fh ; M 4D
DB 07Fh,004h,008h,010h,07Fh ; N 4E
DB 03Eh,041h,041h,041h,03Eh ; O 4F
DB 07Fh,009h,009h,009h,006h ; P 50
DB 03Eh,041h,051h,021h,05Eh ; Q 51
DB 07Fh,009h,019h,029h,046h ; R 52
DB 046h,049h,049h,049h,031h ; S 53
DB 001h,001h,07Fh,001h,001h ; T 54
DB 03Fh,040h,040h,040h,03Fh ; U 55
DB 01Fh,020h,040h,020h,01Fh ; V 56
DB 03Fh,040h,038h,040h,03Fh ; W 57
DB 063h,014h,008h,014h,063h ; X 58
DB 007h,008h,070h,008h,007h ; Y 59
DB 061h,051h,049h,045h,043h ; Z 5A
DB 000h,07Fh,041h,041h,000h ; [ 5B
DB 002h,004h,008h,010h,020h ; \ 5C
DB 000h,041h,041h,07Fh,000h ; ] 5D
DB 004h,002h,001h,002h,004h ; ^ 5E
DB 040h,040h,040h,040h,040h ; _ 5F
DB 000h,001h,002h,004h,000h ; ' 60
DB 020h,054h,054h,054h,078h ; a 61
DB 07Fh,048h,044h,044h,038h ; b 62
DB 038h,044h,044h,044h,020h ; c 63
DB 038h,044h,044h,048h,07Fh ; d 64
DB 038h,054h,054h,054h,018h ; e 65
DB 008h,07Eh,009h,001h,002h ; f 66
DB 00Ch,052h,052h,052h,03Eh ; g 67
DB 07Fh,008h,004h,004h,078h ; h 68
DB 000h,044h,07Dh,040h,000h ; i 69
DB 020h,040h,044h,03Dh,000h ; j 6A
DB 07Fh,010h,028h,044h,000h ; k 6B
DB 000h,041h,07Fh,040h,000h ; l 6C
DB 07Ch,004h,018h,004h,078h ; m 6D
DB 07Ch,008h,004h,004h,078h ; n 6E
DB 038h,044h,044h,044h,038h ; o 6F
DB 07Ch,014h,014h,014h,008h ; p 70
DB 008h,014h,014h,014h,07Ch ; q 71
DB 07Ch,008h,004h,004h,008h ; r 72
DB 048h,054h,054h,054h,020h ; s 73
DB 004h,03Fh,044h,040h,020h ; t 74
DB 03Ch,040h,040h,020h,07Ch ; u 75
DB 01Ch,020h,040h,020h,01Ch ; v 76
DB 03Ch,040h,030h,040h,03Ch ; w 77
DB 044h,028h,010h,028h,044h ; x 78
DB 00Ch,050h,050h,050h,03Ch ; y 79
DB 044h,064h,054h,04Ch,044h ; z 7A
DB 000h,018h,018h,000h,000h ; · 7B
DB 000h,020h,050h,020h,000h ; □ 7C
DB 015h,016h,07Ch,016h,015h ; ¥ 7D

```

6.8 Precautions

For the restrictions, support functions and bug information of the latest version, refer to rel_utility_E.txt of S5U1C88000Q.

7 PORT SETTING UTILITY

7.1 Overview

The port setting utility (PrtUtil.exe, hereafter referred to as PrtUtil) allows the simulator (sim63.exe or sim88.exe) to simulate key inputs on the target. It is used to assign ports to the push keys or key matrix and to set the PC keys used in simulation. The set data is written out to a port setting file (.prt) in text format. When this file is loaded into the simulator, key inputs on the target can be simulated using a PC keyboard.

PrtUtil is an model-independent, common tool.

7.2 Input/Output Files

Figure 7.2.1 shows the input/output files for PrtUtil.

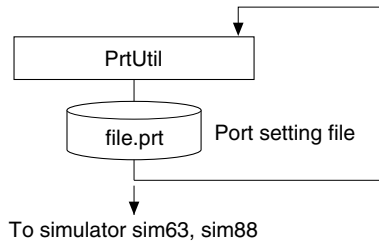


Fig. 7.2.1 Input/output files for PrtUtil

Port setting file (file_name.prt)

This text file defines the relationship between the push keys or key matrix and the ports and is loaded into the simulator to simulate key inputs. Once created, the port setting file can be modified in PrtUtil by reloading it into the utility.

7.3 Starting and Terminating PrtUtil



Double-click this icon to start PrtUtil.

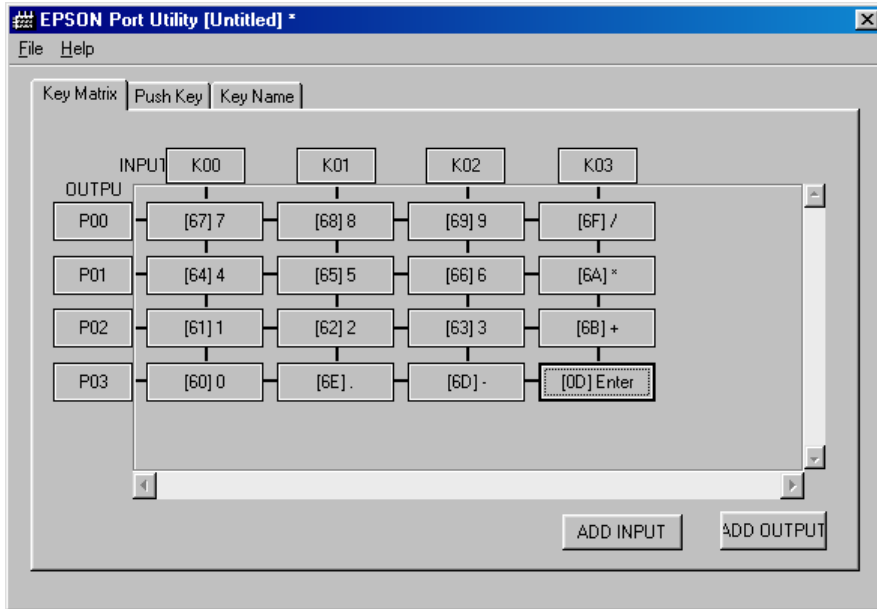
PrtUtil.exe

To quit, select [Exit] from the [File] menu.

7.4 Window

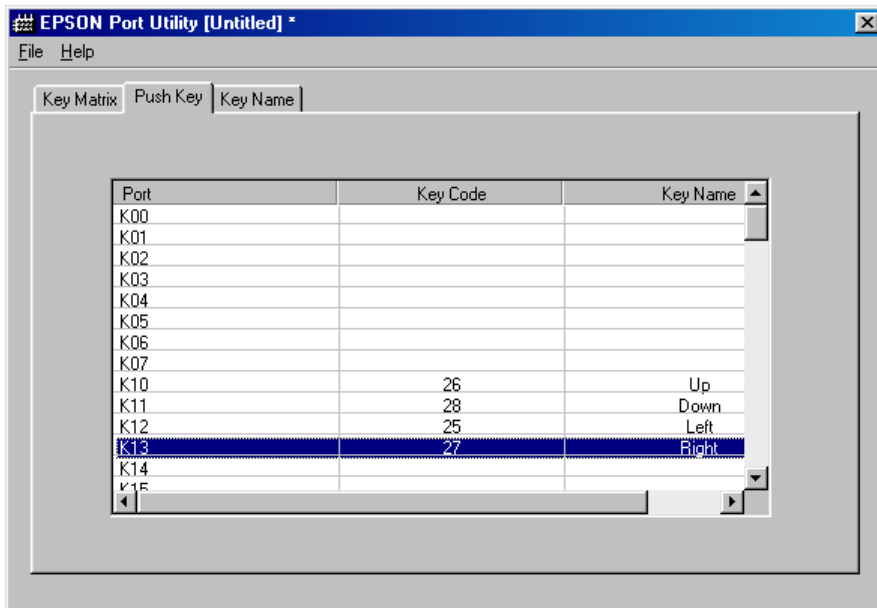
The PrtUtil window has three tabs: [Key Matrix], [Push Key], and [Key Name]. Select the tab appropriate for the items that need to be set.

[Key Matrix]

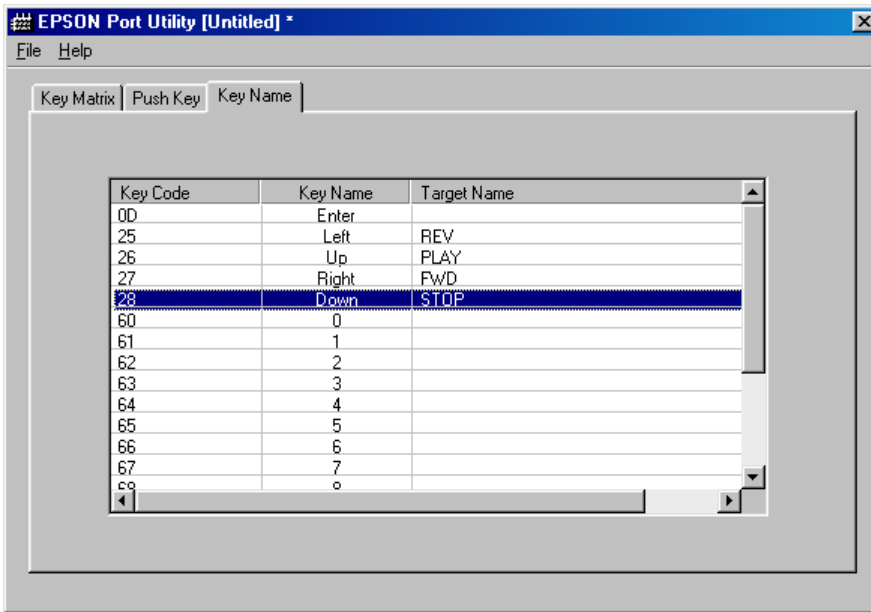


Use this tab to specify the input/output ports comprising the key matrix and the PC keys used for key input simulation.

[Push Key]



Use this tab to specify the input ports used for push key input and the PC keys used for key input simulation.

[Key Name]

Use this tab to define the relationship between the key names on the target and the PC keys used for key input simulation that are assigned the key names.

7.5 Menu

[File] menu**[New]**

Starts a new port setting.

[Open...]

Opens a port setting file (.prt).

[Save]

Saves (overwrites) the contents currently being edited to the port setting file (.prt).

[Save as...]

Saves the contents currently being edited to the port setting file (.prt), specifying a new file name.

[Print...]

Prints the current settings.

[Exit]

Quits PrtUtil.

[Help] menu**[About PrtUtil...]**

Displays version information for PrtUtil.

7.6 Creating Port Setting Data

The simulator supports a function that simulates key inputs for the key matrix using the S1C63xxx/S1C88xxx's input/output ports and the push keys connecting to the input ports. PrtUtil allows the following three types of settings to be made separately:

1. Specify the ports comprising the key matrix and the PC keys used for simulation.
2. Specify the input ports that have push keys connected and the PC keys used for simulation.
3. Define the relationship between the PC keys and the target system key names, which are used to handle the PC keys in the simulator.

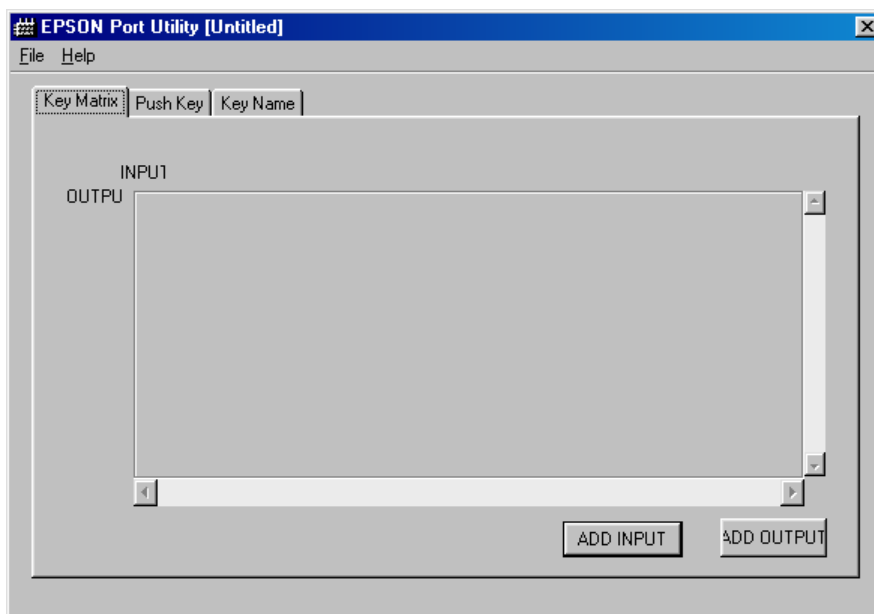
Unnecessary items do not need to be set.

The set data is saved collectively in a single port setting file.

The procedure for creating data is described below.

7.6.1 Creating New Data

On startup, PrtUtil displays the following window:



The user can begin creating new data here.

To create new data while or after editing data, select [New] from the [File] menu. All set data is cleared, and PrtUtil is in the same state as immediately following startup. If you select [New] before the most recent data is saved, a message will prompt you to save this data. Either save or discard the data, or stop creating new data.

7.6.2 Editing the Existing Data

To modify the existing data, select [Open...] from the [File] menu and load the port setting file to edit. If you select [Open...] before the most recent data is saved, a message will prompt you to save this data. Either save or discard the data, or stop loading the file.

When the file is loaded, the set contents are displayed in the window. Start by making the necessary additions or corrections. Always save the created or edited data, using [Save] or [Save as...] from the [File] menu.

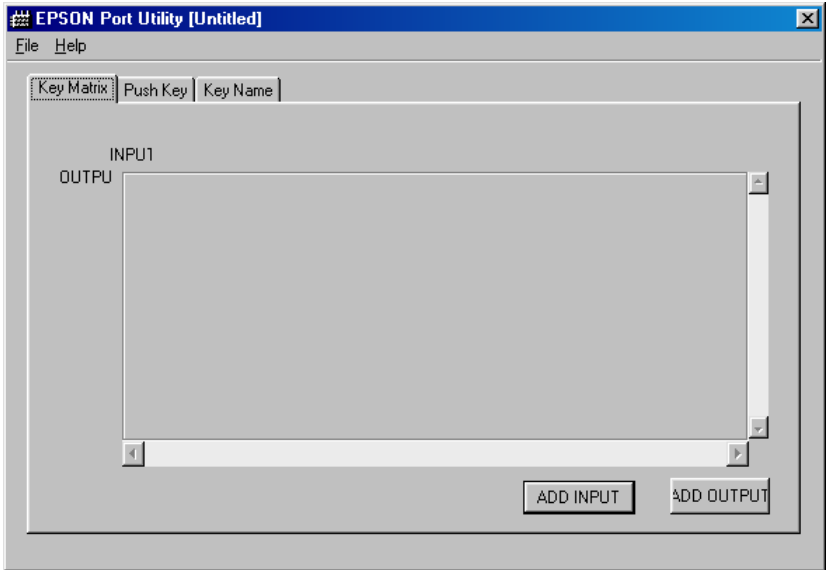
7.6.3 Setting Key Matrix Data

This section describes how to create or modify key matrix data. If the target system does not have a key matrix, no data needs to be set here.

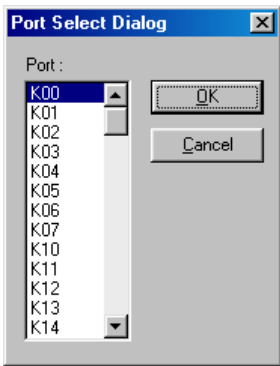
Creating new key matrix data

The procedure for creating new key matrix data is described below.

- 1) Click the [Key Matrix] tab to display the key matrix edit screen shown below.



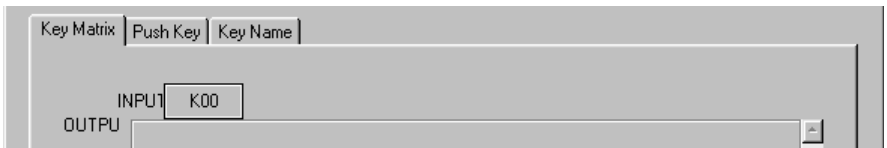
- 2) Click the [ADD INPUT] button. The following [Port Select] dialog box appears.



The input ports (Kxx, Pxx) are listed in this dialog box.

Note: Because PrtUtil can be used with all models in the S1C63/S1C88 Family, some of the port names displayed may not actually be available for some models.

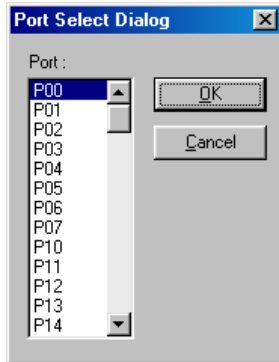
- 3) Select an input port according to the key matrix specifications. Click [OK].



The selected port name is displayed in the edit screen. Select all input ports comprising the key matrix in the same way.



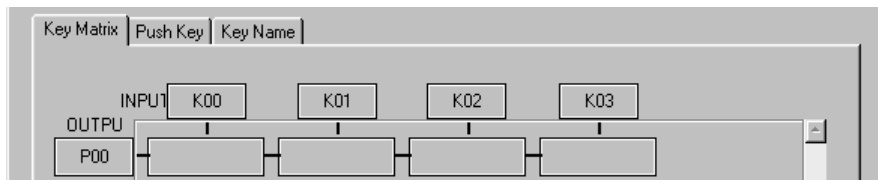
- 4) Click the [ADD OUTPUT] button. The following [Port Select] dialog box is displayed.



Output ports (Pxx, Rxx) are listed in this dialog box.

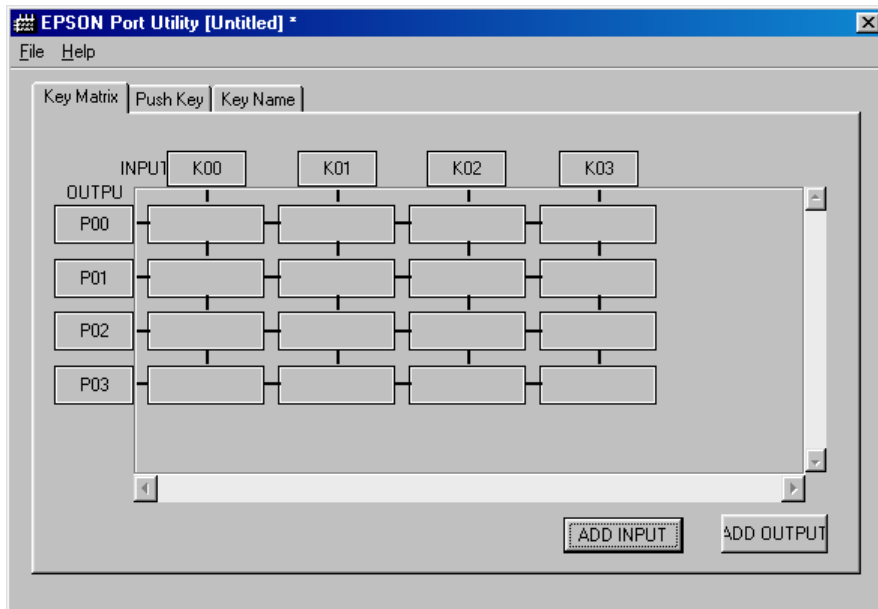
Note: Because PrtUtil can be used with all models in the S1C63/S1C88 Family, some of the port names displayed may not actually be available for some models.

- 5) Select an output port to use according to the key matrix specifications. Click [OK].

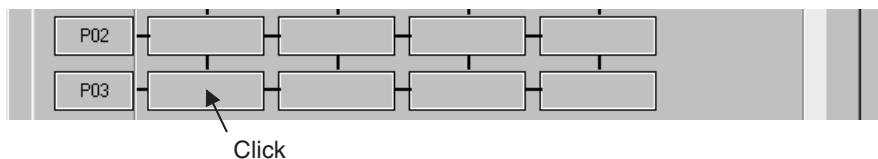


The selected port name is displayed in the edit screen.

Select all output ports comprising the key matrix in the same way.

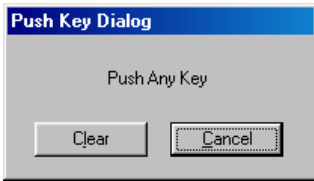


- 6) Click the box to which to assign a key on the PC keyboard.

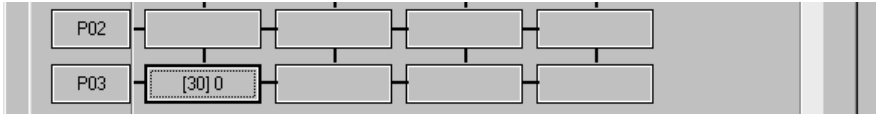


7 PORT SETTING UTILITY

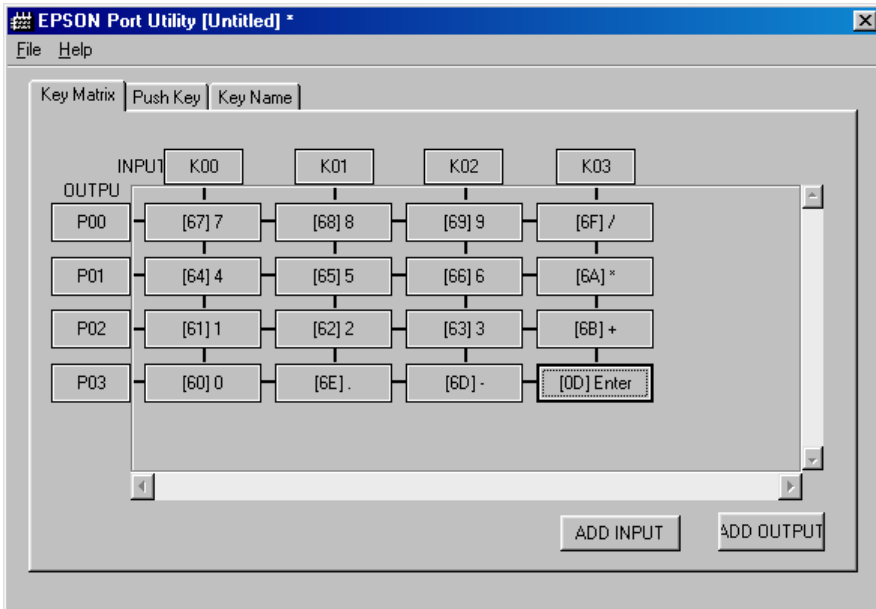
The following [Push Key] dialog box is displayed.



- 7) Press any key on the PC keyboard that will be used during simulation. This closes the dialog box, and the code and the character or name for the pressed key are displayed in the box.



Assign all other keys in the same way.



Note: Despite representing the same number or symbol internally, the keys on the standard keyboard and those on the numeric keypad have different codes.

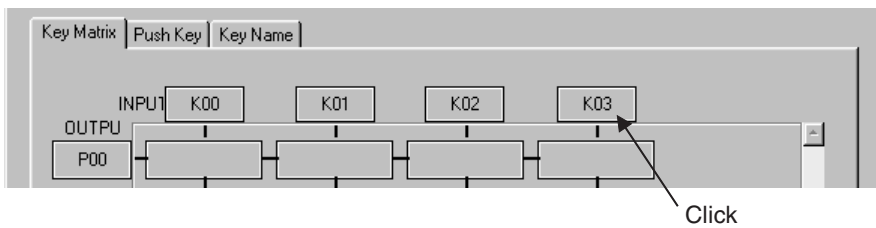
- 8) Select [Save] or [Save as...] from the [File] menu to save the data.

Modifying the key matrix data

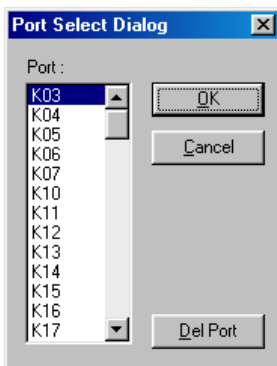
The following describes how to modify the key matrix data created.

Changing the ports

- 1) Click the input port or output port name to change or remove.



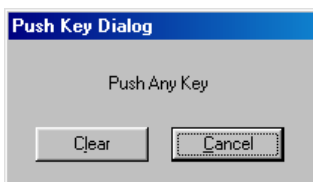
The following [Port Select] dialog box is displayed.



- 2) To replace the port with another port, select a new port name from the list and click [OK].
- 3) To remove the port, click [Del Port]. One row of the key matrix is deleted.
- 4) To stop modifying the key matrix, click [Cancel].

Changing the PC keys

- 1) Click the box whose PC key is to be changed.
The following [Push Key] dialog box is displayed.



- 2) To assign another key to the box, press that key.
- 3) To remove a PC key assignment, click [Clear].
- 4) To stop modifying the PC keys, click [Cancel].

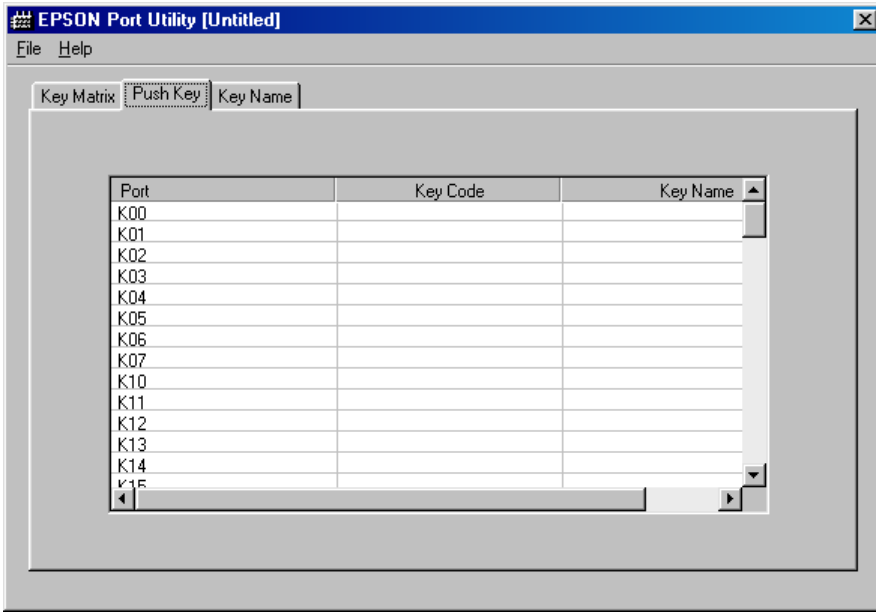
7.6.4 Setting Push Key Data

This section describes how to create or modify push key data. If the target system does not have push keys, no data needs to be set here.

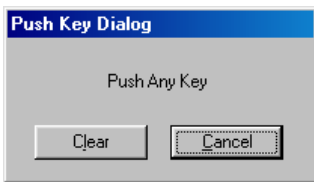
Creating new push key data

The procedure for creating new push key data is described below.

- 1) Click the [Push Key] tab to display the push key edit screen shown below.

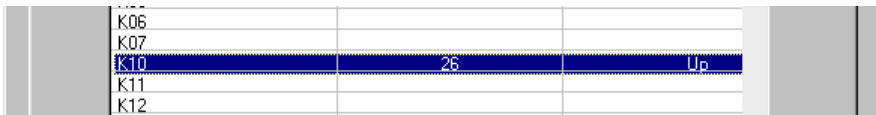


- 2) Double-click the input port line to which a push key is to be connected. The following [Push Key] dialog box is displayed.

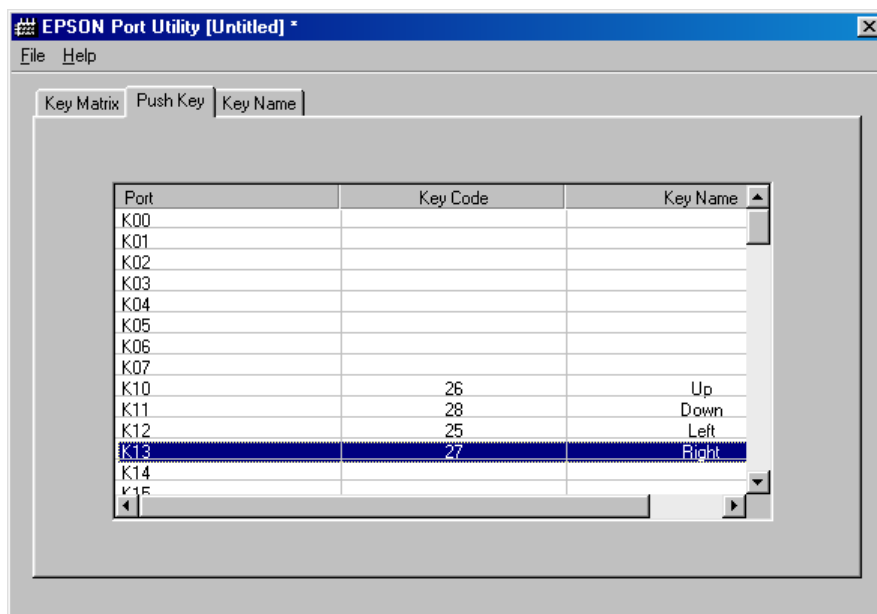


Note: Because PrtUtil can be used with all models in the S1C63/S1C88 Family, some of the port names displayed may not actually be available for some models.

- 3) Press any key on the PC keyboard that will be used during simulation. This closes the dialog box, and the code and the character or name for the pressed key are displayed in the selected port line.



Set all other input ports to which the push keys are to be connected in the same way.



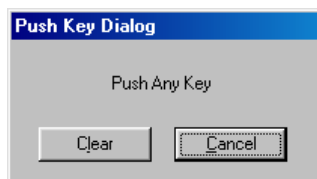
Note: Despite representing the same number or symbol internally, the keys on the standard keyboard and those on the numeric keypad have different codes.

- 4) Select [Save] or [Save as...] from the [File] menu to save the data.

Modifying the push key data

The procedure for modifying the push key data created is described below.

- 1) Click the input port line to be modified.
The following [Push Key] dialog box is displayed.

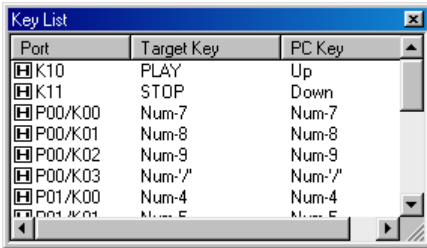


- 2) To assign the input port another key, press that key.
- 3) To remove PC key assignment, click [Clear].
- 4) To stop modifying push key data, click [Cancel].

7.6.5 Setting Target Key Names

This section describes how to define the relationship between the PC keys and the target system key names, which are used to handle the PC keys in the simulator. The target key names set here are displayed along with the PC key names in the simulator's [Key List] window.

The settings made here are intended to assist with debugging, and in no way affect the functionality or behavior of the simulation.

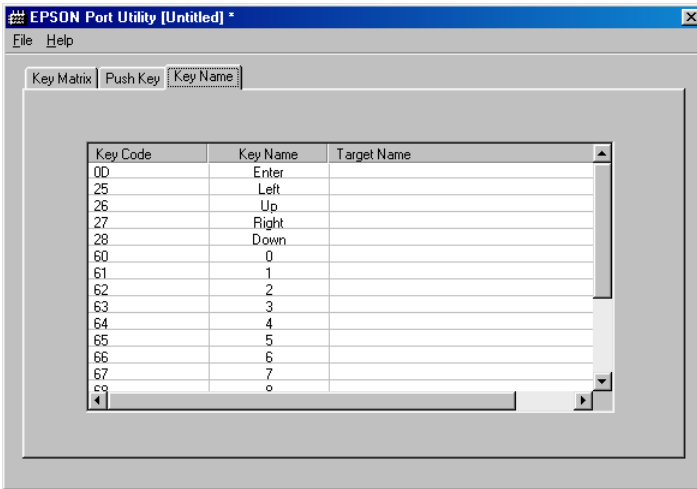


(Example display on the simulator's [Key List] window)

The undefined key names on the target assume the same names as the PC keys when displayed in this window.

The procedure for defining the target key names is described below.

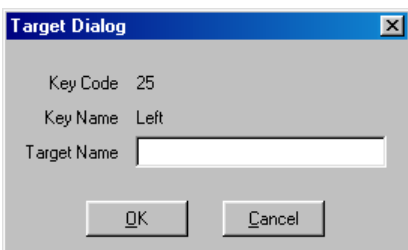
- 1) Before proceeding, make sure the key matrix and the push keys have been set.
- 2) Click the [Key Name] tab to display the key name edit screen shown below.



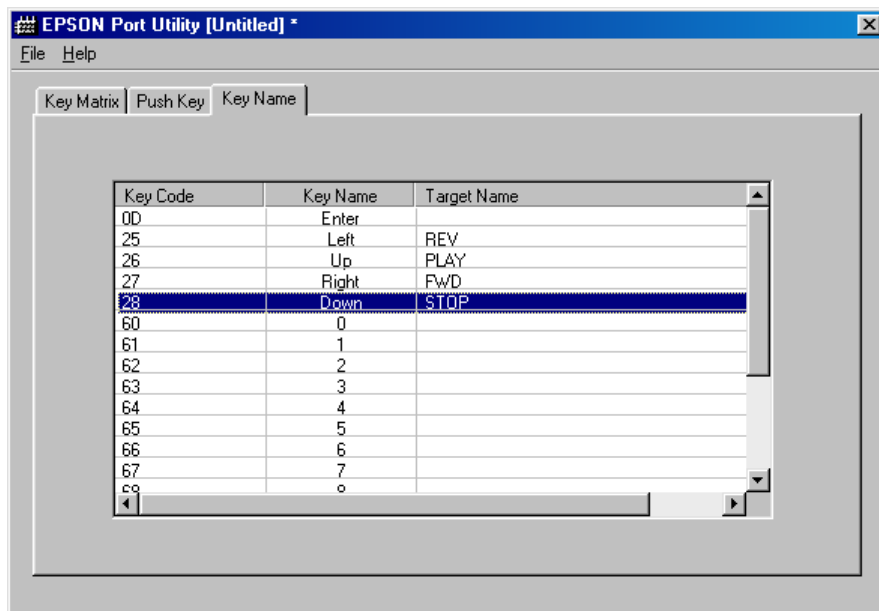
The key codes and the key names for the PC keys assigned during key matrix and push key settings are displayed in this screen.

Note: Unless the key matrix or push keys are assigned PC keys, no key codes and key names are shown here.

- 3) Double-click the PC key line in which you want to define a target key name. The following [Target] dialog box is displayed.



- 4) Enter a target key name in the [Target Name] box and click [OK].
This closes the dialog box and displays the entered key name in the selected PC key line.



- 5) Repeat steps 3 and 4 for a number of times equal to the number of keys to be defined.
- 6) If the key names set need to be modified, also follow steps 3 and 4. To clear any defined key name, delete the characters in the [Target] dialog box's [Target Name] box for that key and click [OK].

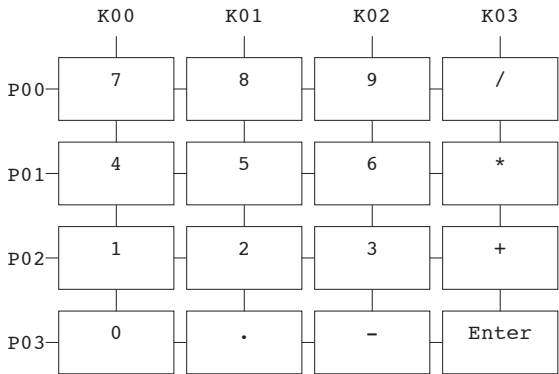
7.6.6 Printing

Select [Print...] from the [File] menu to print the created data.

Example print:

```
[Push Key]
K10=Up [PLAY]
K11=Down [STOP]
K12=Left [REV]
K13=Right [FWD]
```

[Key Matrix]



7.7 Precautions

- Despite representing the same number of symbol internally, the keys on the PC's standard keyboard and those on the numeric keypad have different codes.

Example: The keys 0 to 9 on the standard keyboard are 30h to 39h,
while those on the numeric keypad are 60h to 69h.

Be especially careful when using the same definition file on a desktop PC and notebook PC.

- The following keys on the PC keyboard cannot be assigned for key input simulation:
ALT, Windows, Print, Screen, Num Lock, Scroll Lock, Pause (break), Caps, Fn (Function), special language keys other than Roman characters.
- Since PrtUtil can be used with all models in the S1C63/S1C88 Family, port names that are not actually available for some models may be displayed. Be careful to avoid selecting ports that are not available.

For the restrictions, support functions and bug information of the latest version, refer to rel_utility_E.txt of S5U1C88000Q.

7.8 Port Setting File (.prt)

The contents of a port setting file are shown below. The contents of the definitions are separated into three sections: [Push Key], [Key Matrix], and [Key Name]. Only sections that are set are written in this file.

Definition of push keys

Subsequent to the section declaration [Push Key], the relationship between ports and key codes (described later) is written.

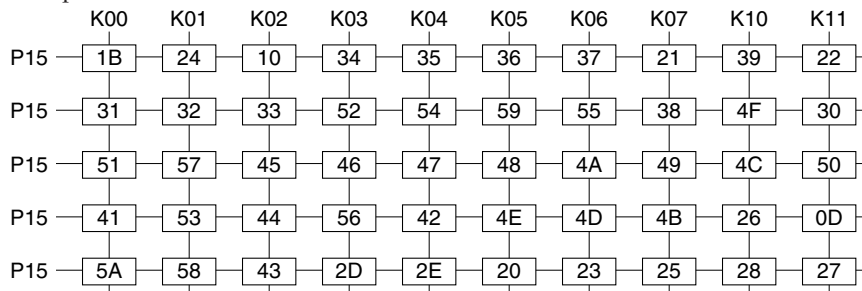
Example:

```
[Push Key]
K00=30
K01=31
K02=32
K03=33
K04=34
K05=35
K06=36
K07=37
K10=38
K11=39
```

Definition of the key matrix

Subsequent to the section declaration [Key Matrix], the input and the output ports used are written. This is followed by descriptions of codes for the PC keys assigned to each input port. If no keys are assigned, the box is blank.

Example:



[Key Matrix]

OUTPUT=P15,P16,P17,R27,R34

INPUT =K00,K01,K02,K03,K04,K05,K06,K07,K10,K11

```
P15 = 1B, 24, 10, 34, 35, 36, 37, 21, 39, 22
P16 = 31, 32, 33, 52, 54, 59, 55, 38, 4F, 30
P17 = 51, 57, 45, 46, 47, 48, 4A, 49, 4C, 50
R27 = 41, 53, 44, 56, 42, 4E, 4D, 4B, 26, 0D
R34 = 5A, 58, 43, 2D, 2E, 20, 23, 25, 28, 27
```

Key names

Subsequent to the section declaration [Key Name], the relationship between the PC keys assigned to the key matrix or push keys and the target key names defined to those PC keys is written.

Example:

[Key Name]

24=Caps	← Home key (key code 24) = Caps
22=On/Off	← PageDown key (key code 22) = On/Off
21=Mode	← PageUp key (key code 21) = Mode
23=Char	← End key (key code 23) = Char
2D=Data	← Insert key (key code 2D) = Data

Key code list (hexadecimal)

These key codes are received by the simulator when keys are entered from the PC keyboard. The numeric keys (NumPad) have independent key codes.

Key	Code
0 to 9	30 to 39
A to Z	41 to 5A
BackSpace	08
Tab	09
Enter	0D
Shift	10
Ctrl	11
Pause	13
Esc	1B
Space	20
PageUp	21
PageDown	22
End	23
Home	24
Left	25
Up	26
Right	27
Down	28
Insert	2D
Delete	2E

(Numeric keypad)

0 to 9	60 to 69
*	6A
+	6B
-	6D
.	6E
/	6F

(Function keys)

F1 to F24	70 to 87
-----------	----------

8 LOGIC ANALYZER

8.1 Overview

The Logic Analyzer (LogAna.exe, hereafter referred to as LogAna) accepts as input input/output log files created in the simulator (Sim63/Sim88)'s [I/O Terminal] window, displaying waveforms representing the input/output port status recorded in those files (serial input/output and A/D conversion logs unsupported). LogAna is a model-independent, common tool. For detailed information on port input/output simulations via the simulator's [I/O Terminal] window, refer to "[I/O Terminal] Window" in Section 3.6.4 (for the S1C63 Family) or Section 4.6.4 (for the S1C88 Family).

8.2 Input Files

LogAna accepts as input [I/O Terminal] input/output files.

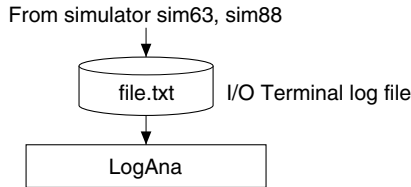


Fig. 8.2.1 Input files for LogAna

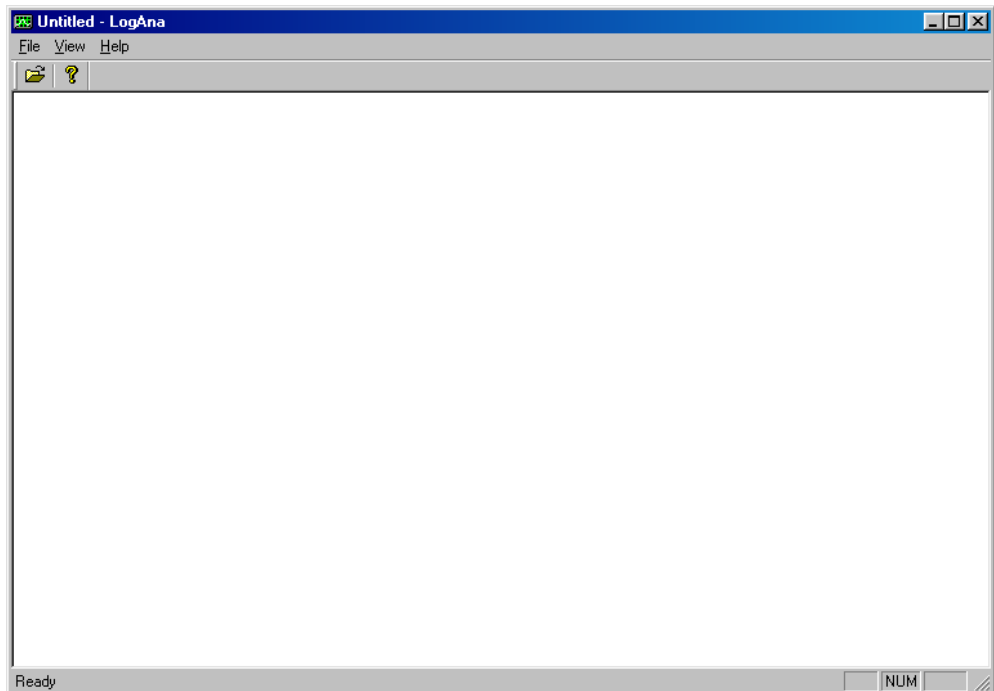
Input/output log files (file_name.txt)

This text file contains the logs displayed on the simulator's [I/O Terminal] window. These log files are created by selecting [Save] from the [I/O Terminal] window's [File] menu.

8.3 Starting and Terminating LogAna



Double-click this icon to start LogAna. The following window appears.



To quit, select [Exit] from the [File] menu.

8.4 Menus and Toolbar

8.4.1 Menus

[File] menu



[Open...] ([Ctrl] + [O])

Opens an input/output log file.

File list

The names of recently opened files are listed here. You can open a file by selecting it from this list.

[Exit]

Quits LogAna.

[View] menu



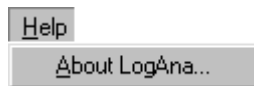
[Toolbar]

Displays or hides the toolbar.

[Status Bar]

Displays or hides the status bar.

[Help] menu



[About LogAna...]

Displays version information for LogAna.

8.4.2 Toolbar Buttons



[Open] button

Opens an input/output log file.



[About] button

Displays version information for LogAna.

8.5 Using LogAna

Described below are procedures for tasks ranging from creating input/output log files to displaying waveforms using LogAna.

- Using a text editor, create the IOT file to be used for input/output simulations in the simulator. For detailed information on creating IOT files, refer to "IOT Files (.iot)" in Section 3.13 (for the S1C63 Family) or Section 4.13 (for the S1C88 Family).

Note that ports whose input/output waveforms are to be displayed must be written using a "watch" statement in this file.

```
Example: [General I/O]
        watch P10,P11,P12,P13,P14,P15,P16,P17
        :
```

Because the input/outputs on the undefined ports are not recorded in the [I/O Terminal] window logs, waveforms for those ports cannot be displayed by LogAna.

- Start the simulator and load the required files, including the target program.
- Click the [I/O] button on the simulator's [LCD] window to display the [I/O Terminal] window.
- Select [Open] from the [I/O Terminal] window's [File] menu and load the IOT file.
- Set breakpoints or break conditions as necessary.
- After resetting the simulator, run the target program.

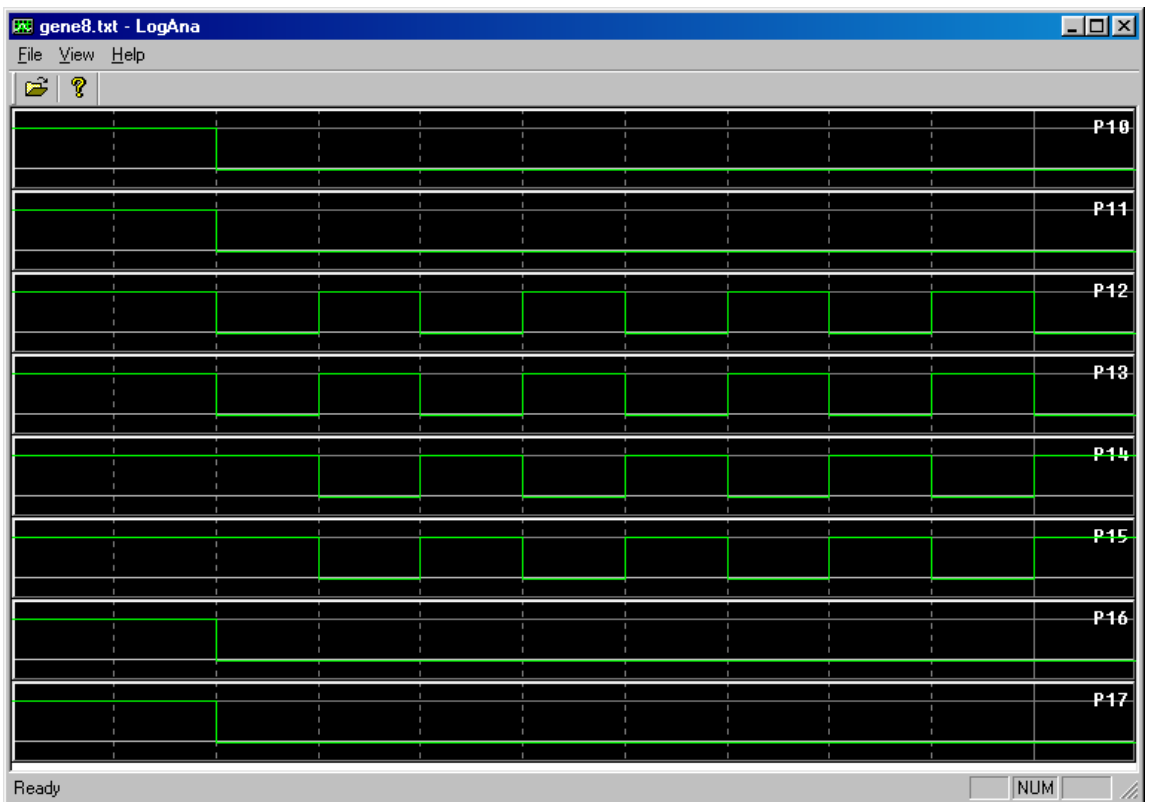
The simulator monitors the status of the specified ports and outputs logs of input/output changes on those ports to the [I/O Terminal] window.
- Wait until the program execution breaks under the break conditions set. Or use the [Key Break] button to halt program execution.
- Select [Save] from the [I/O Terminal] window's [File] menu to save the logs to a file. The file name can have any extension (normally, .txt).

Example log file:

```
P10=1 P11=1 P12=1 P13=1 P14=1 P15=1 P16=1 P17=1

00:00:01.964 P10 >> 0 P11 >> 0 P12 >> 0 P13 >> 0 P16 >> 0 P17 >> 0
00:00:02.964 P12 >> 1 P13 >> 1 P14 >> 0 P15 >> 0
00:00:03.964 P12 >> 0 P13 >> 0 P14 >> 1 P15 >> 1
00:00:04.964 P12 >> 1 P13 >> 1 P14 >> 0 P15 >> 0
00:00:05.964 P12 >> 0 P13 >> 0 P14 >> 1 P15 >> 1
00:00:06.964 P12 >> 1 P13 >> 1 P14 >> 0 P15 >> 0
00:00:07.964 P12 >> 0 P13 >> 0 P14 >> 1 P15 >> 1
00:00:08.964 P12 >> 1 P13 >> 1 P14 >> 0 P15 >> 0
00:00:09.964 P12 >> 0 P13 >> 0 P14 >> 1 P15 >> 1
00:00:10.964 P12 >> 1 P13 >> 1 P14 >> 0 P15 >> 0
```

- Start LogAna.
- Select [Open...] from the [File] menu or use the [Open] button to load the log file. Waveforms are displayed according to the log contents.



(Example waveform displayed from the log file loaded in the preceding page)

In the log, changes in port status are recorded in units of 1 ms. Regardless of whether the clock frequency on the target is set to OSC1 or OSC3, logs are always recorded at this interval time. Therefore, waveforms are also displayed with a resolution of 1 ms.

LogAna does not support the loading of multiple logs. If logs need to be inspected at the same time, multiple units of LogAna must be opened.

8.6 Precautions

For the restrictions, support functions and bug information of the latest version, refer to rel_utility_E.txt of S5U1C88000Q.

9 ROM DATA SETTING UTILITY

9.1 Overview

The ROM Data Setting Utility (Rom88.exe, hereafter referred to as Rom88) loads binary format ROM image data into the S1C88 Family simulator (Sim88). This utility also saves data that has been patched in the Sim88 [Dump] window as a binary format ROM image.

Note: Rom88 is a dedicated utility for the S1C88 Family only. It is not compatible with the S1C63 Family. For information about the operating system (OS) environment, refer to Readme_E.txt of S5U1C88000Q.)

9.2 Input/Output Files

Figure 9.2.1 shows the Rom88 input/output files.

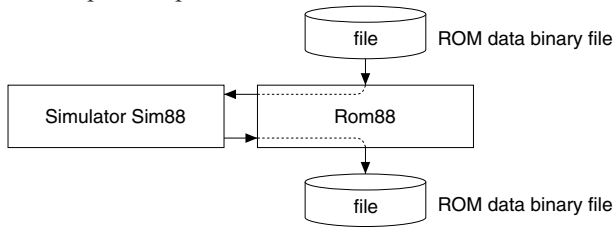


Fig. 9.2.1 Input/output files for Rom88

ROM data binary file

This is the ROM image data used for the S1C88xxx.

9.3 Using Rom88

Given below are instructions on using Rom88.

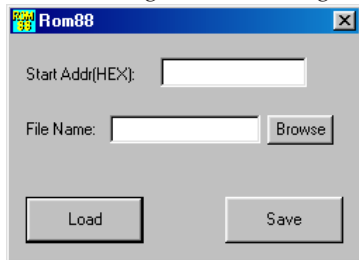
Loading ROM image data

- 1) **IMPORTANT** Always start the simulator Sim88 first. Load the parameter file and all other required files before running Rom88.
- 2) Double-click the icon shown below to start Rom88.



Rom88.exe

The following [Rom88] dialog box is displayed.



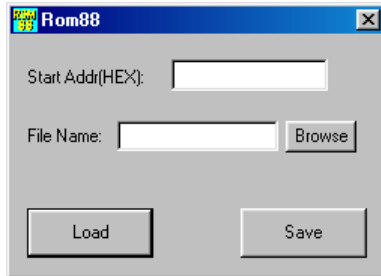
- 3) In the [Start Addr(HEX)] box, enter the start address of the memory to load in hexadecimal notation.
- 4) Use the [Browse] button to display a file select dialog box. Select a file to load in the dialog box, or enter a file name directly into the [File Name] box.
- 5) Click the [Load] button.

When the above is complete, data is loaded from the specified file into the simulator's memory area. Rom88 is terminated upon completion of loading.

Saving ROM image data

- 1) Load ROM image data into the simulator following the procedure described above.
- 2) Edit the data for ROM area in the simulator as necessary.
- 3) Start Rom88.

The following [Rom88] dialog box is displayed.



- 4) In the [Start Addr(HEX)] box, enter the start address of the memory to be saved in hexadecimal notation.
- 5) Use the [Browse] button to display a file select dialog box. Select a file to save in the dialog box, or enter a file name directly into the [File Name] box.
In either case, specify an existing file as the destination to which to save this data.
- 6) Click the [Save] button.

When the above steps are completed, data is written from the specified address to the file. Rom88 terminates upon completion of the save.

9.4 Precautions

Rom88 writes all of the contents of the specified binary file to a file, beginning with the address specified. Make sure the size of the ROM image data to be loaded does not exceed the area specified in the parameter file.

For the restrictions, support functions and bug information of the latest version, refer to rel_utility_E.txt of S5U1C88000Q.

AMERICA

EPSON ELECTRONICS AMERICA, INC.

2580 Orchard Parkway,
San Jose, CA 95131, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

EUROPE

EPSON EUROPE ELECTRONICS GmbH

Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA

EPSON (CHINA) CO., LTD.

7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-6410-6655 FAX: +86-10-6410-7320

SHANGHAI BRANCH

7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5522 FAX: +86-21-5423-5512

SHENZHEN BRANCH

12F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen 518057, CHINA
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

EPSON HONG KONG LTD.

20/F, Harbour Centre, 25 Harbour Road,
Wanchai, Hong Kong
Phone: +852-2585-4600 FAX: +852-2827-4346
Telex: 65542 EPSCO HX

EPSON TAIWAN TECHNOLOGY & TRADING LTD.

14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

EPSON SINGAPORE PTE., LTD.

1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORP.**KOREA OFFICE**

50F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

SEIKO EPSON CORP.**SEMICONDUCTOR OPERATIONS DIVISION****IC Sales Dept.****IC International Sales Group**

421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117