

User's Manual

ID78K0

Integrated Debugger

Guide (Windows™ based operation)

Target device **78K/0 series**

Document No. U11649EJ1V2UM00 (1st edition)
Date Published March 1998 J CP(K)

© NEC Corporation 1996
Printed in Japan

IBM PC/AT is a trademark of International Business Machines Corporation.

i386 and i486 are trademarks of Intel Corporation.

MS-DOS and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Windows is an abbreviation of Microsoft™ Windows™ Operating System.

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or of others.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)

Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH

Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.

Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.

Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH

Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.

Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.

Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH

Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.

Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.

Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.

United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.

Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.

Cumbica-Guarulhos-SP, Brasil
Tel: 011-6465-6810
Fax: 011-6465-6829

Preface

Thank you for purchasing the ID78K0 integrated debugger.

Conventional debuggers are used by entering commands directly. The ID78K0 integrated debugger, on the other hand, runs under Windows to provide a friendly, easy-to-use GUI (Graphical User Interface). Its operation is mouse-based, and operation is possible without having to refer to the manual. Also, frequently used commands are represented as buttons, allowing their activation simply by clicking the button with the mouse.

«Purpose»

The purpose of this manual is to provide the user with a brief explanation of how to use the ID78K0 integrated debugger. This manual should be read together with the "ID78K0 Integrated Debugger User's Manual (Reference)." For a detailed explanation of each window, refer to the "ID78K0 Integrated Debugger User's Manual (Reference)."

«Files supplied with the integrated debugger»

Files used with the integrated debugger

File name	Explanation
ID78K0.EXE	Debugger main section. The debugger is started by executing this file.
ID78K0P.DLL	Contains the libraries used for link processing with Project Manager.
DB78K0.DLL	Contains libraries for file and symbol processing.
AS78K0.DLL	Contains libraries for assembly and disassembly.
EX78K0.DLL	Contains libraries for communication with the in-circuit emulator.
EX78K0.OM0	Downloaded into the in-circuit emulator when the debugger starts.
ID78K0.HLP	Help file.
EXPC.INI	Initial file. Used to specify a set point and an interrupt address for the PC interface board.

Sample programs

File name	Explanation
SAMPLE.C	Sample program written in C.
SUB.C	Sample program written in C. Contains the subroutines of SAMPLE.C.
SAMPLE.LNK	Load module file for sample programs SAMPLE.C and SUB.C. Compiled by μ PD78014.

«Target device»

The device which is to be the target of debugging by the integrated debugger is called a target device. The table below lists target devices, their associated device files, microprograms, and the names of the CPUs which select the target devices.

Target device	CPU name	Device file
μ PD78014	78014	D014.78K
μ PD78044	78044	D044.78K
μ PD78054	78054	D054.78K
μ PD78064	78064	D064.78K

Note: For details of other devices, contact your NEC sales representative or authorized dealer.

«In-circuit emulator»

An in-circuit emulator and dedicated interface board are required to use the integrated debugger.

The table below lists the in-circuit emulator boards and interface boards that can be connected to host machines.

In-circuit emulator

Product name	Explanation
IE-78000-R-A	In-circuit emulator main board
IE-78xxx-R-EM(Note 1)	Product type dependent board

Note 1. For details, contact your NEC sales representative or authorized dealer.

Interface boards

Product name	Explanation
IE-70000-98-IF-A	Interface board for PC-9801 and 9821 Series (C bus)
IE-70000-98-IF-B	Interface board for PC-9801 and 9821 Series (C bus)
IE-70000-98N-IF(Note 2)	Interface board for 98NOTE (110-pin expansion bus)
IE-70000-PC-IF-B(Note 3)	Interface board for IBM-PC/AT Series (ISA bus)

Note 2. The IE-70000-98N-IF is corrected to the expansion bus (110-pin type) of 98NOTE.

Note 3. The IE-70000-PC-IF-A cannot be used.

«Host machine»

The integrated debugger runs under Windows. The table below lists the requirements for the machine to be used.

Item	Requirement
Host machine	PC-9801, 9821 or IBM-PC/AT Series
CPU	i80386 or above (i80486, 33 MHz or above recommended)
Main memory	4M bytes or more (8M bytes or more recommended)
OS	Windows 3.1 or Windows 95
Screen size	640 x 400 dots or larger (800 x 600 dots or larger recommended)

«Configuration»

- **Chapter 1 Overview**

Explains general operations of the integrated debugger.

- **Chapter 2 Basic Operations**

Explains the relationships between windows and other information by purpose.

- **Chapter 3 Advanced Use of ID78K0**

Describes the terms used in the explanation of the integrated debugger.

«Conventions»

The following explains the conventions used throughout this manual.

	: Indicates a key to be pressed.
+	: Indicates keys which must be pressed at the same time.
“ ”	: Indicates a character string.
‘ ’	: Indicates a character.
[]	: Indicates an optional parameter.
GRPH key	: Representation of a key featured by the PC-9801 and 9821 Series.
The Alt key of the IBM-PC/AT Series has the same function.	

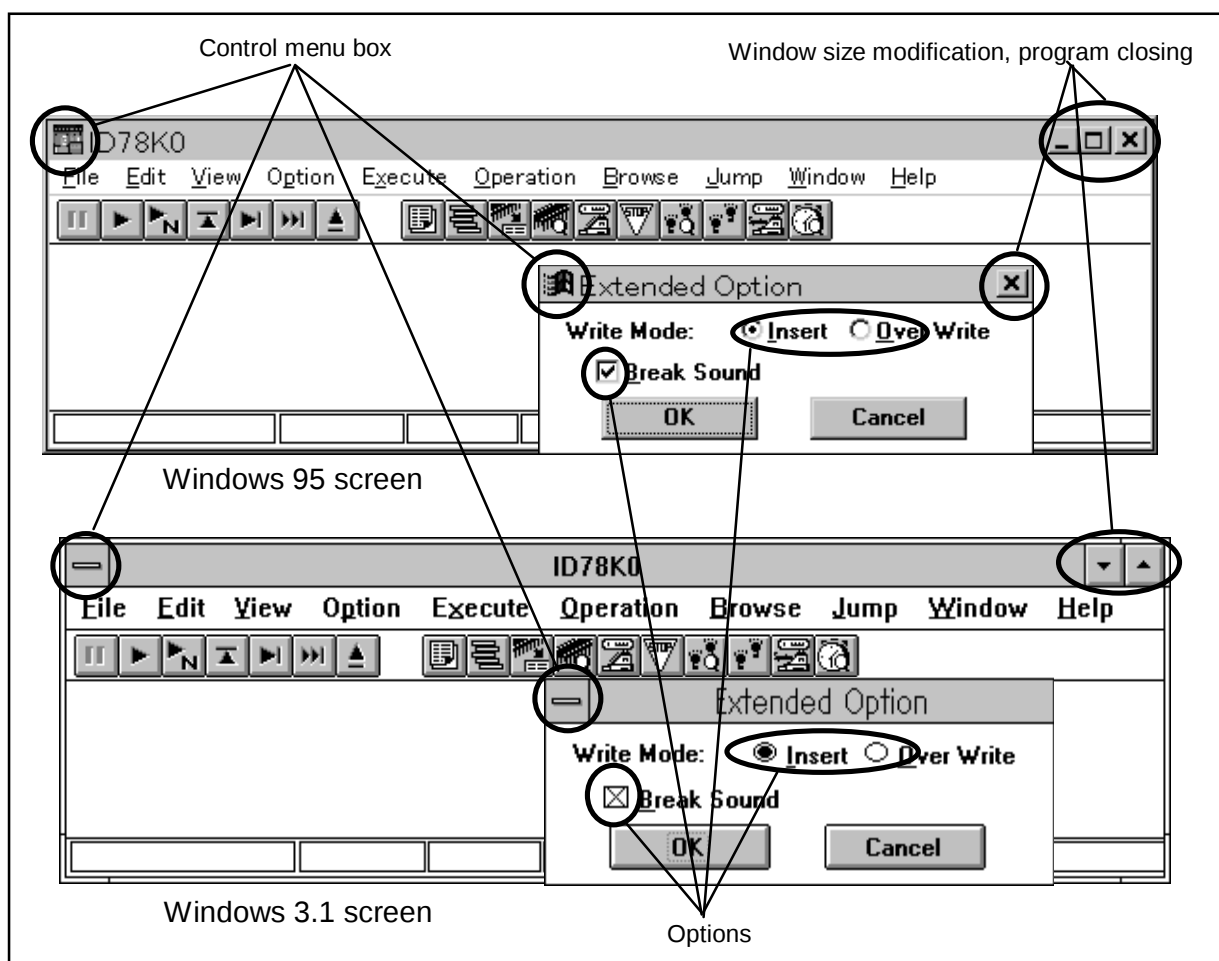
All representations of keys in this manual are for the PC-9801 and 9821 Series. When using an IBM-PC/AT Series computer as a host machine, see **Appendix B**.

«Screen»

The descriptions in this manual refer to Windows 95 screens unless specified otherwise.

The differences between Windows 3.1 screens and Windows 95 screens are as described below.

	Windows 3.1	Windows 95	Remarks
Control menu box			Displays the control menu. With Windows 95, an icon or the Windows logo is displayed.
Window size modification			Minimizes the window.
			Maximizes the window.
			Restores the window to its original size.
Close button	(None)		Closes the window.
Option	 	 	Multiple options can be selected.
	 	 	Only one of the multiple options can be selected.



«Cautions»

- To perform source debugging, add options for creating debug information whenever compiling, assembly, or linking is performed. Otherwise, source debugging may not be possible.
- When creating your own startup routine in C, add the symbols given below. Failing to do so may result in part of the step execution not being performed correctly.

Where to add	Symbol to be added
Start of startup routine	__@cstart
End of startup routine	__@cend

«Related Documents»

The documents (user's manuals) related to this manual are listed below:

Document name		Document number	
		Japanese	English
ID78K0 Integrated Debugger User's Manual, Reference		U11539J	U11539E
RA78K Series Assembler Package	Language	EEU-815	EEU-1399
	Operation	EEU-809	EEU-1404
RA78K Series Structured Assembler Preprocessor		EEU-817	EEU-1402
CC78K Series C Compiler	Language	EEU-655	EEU-1280
	Operation	EEU-656	EEU-1284
78K/0 Series User's Manual, Instructions		IEU-849	EEU-1372
μPD78014, 78014Y Sub-Series		U-10085JJ	EEU-1343

Note: The above documents may be revised without notice. Use the latest versions when designing an application system.

Chapter 1	Overview-----	1
1.1	Starting and Terminating the Debugger -----	2
1.1.1	Starting -----	2
1.1.2	Terminating-----	3
1.2	Making Maximum Use of the Main Window -----	4
1.2.1	Main Window Functions-----	4
1.2.2	Making Maximum Use of Menus-----	5
1.2.3	One-Touch Tool Bar Operation-----	5
1.2.4	Using Information Provided by the Status Bar -----	7
Chapter 2	Basic Operations -----	8
2.1	Establishing the Environment -----	9
2.1.1	Selecting a Device-----	10
2.1.2	Selecting a CPU Clock -----	11
2.1.3	Mapping-----	12
2.1.4	Specifying a Stack Area-----	13
2.1.5	Setting the Alternate Software Operation Clock -----	14
2.1.6	Setting Memory Banks -----	16
2.1.7	Loading/Saving the Debugging Environment -----	18
2.2	Source Level Debugging-----	20
2.2.1	Notes on Compilation, Assembly, and Linking-----	20
2.2.2	Downloading a Program-----	21
2.2.3	Displaying a Source -----	22
2.2.4	Functions Supported by the Source Window -----	23
2.2.5	Jump from the Source Window-----	24
2.3	Instruction Level Debugging-----	25
2.3.1	Assembly Language Display and Online Assembly -----	26
2.3.2	Saving and Referencing Displayed Assembly Language Code -----	27
2.3.3	Functions Supported by the Assemble Window -----	28
2.3.4	Jump from the Assemble Window -----	29
2.4	Manipulating Memory -----	30
2.4.1	Displaying and Modifying Memory Data-----	30
2.4.2	Basic Memory Data Operations -----	31
2.4.3	Saving and Referencing Displayed Memory Data -----	32
2.4.4	Functions Available in the Memory Window-----	33
2.4.5	Jumping from the Memory Window-----	34
2.5	Manipulating Registers -----	35
2.5.1	Displaying and Modifying Registers -----	35
2.5.2	Saving and Referencing Displayed Register Data -----	36
2.5.3	Functions Available in the Register Window -----	37
2.5.4	Functions Available in the SFR Window -----	37
2.5.5	Jumping from the Register Window -----	38
2.6	Creating Events -----	39
2.6.1	Setting and Referencing Events in the Source Window and Assemble Window-----	40
2.6.2	Creating Event Conditions -----	41
2.6.3	Setting Events -----	43
2.6.4	Saving and Restoring Event Conditions-----	44
2.6.5	Functions Available in the Event Manager-----	45
2.6.6	Jumping to an Event Setting Address-----	46

2.7	Manipulating Symbols (Variables) -----	47
2.7.1	Displaying and Modifying Variables -----	48
2.7.2	Saving and Referencing Symbol Data -----	50
2.7.3	Functions Available in the Variable Window and Local Variable Window -----	51
2.8	Using the Tracer Effectively -----	52
2.8.1	Displaying Trace Results -----	53
2.8.2	Saving and Referencing Trace Results -----	54
2.8.3	Effective Trace Memory Usage 1 (Trace Mode Setting) -----	55
2.8.4	Effective Trace Memory Usage 2 (Trace Full Break, Snapshot Trace) -----	59
2.8.5	Inter-Window Connection Functions (Window Connection Function, Jump Function) -----	61
2.9	Measuring the Execution Time -----	63
2.9.1	Measuring Program Execution Time -----	63
2.9.2	Time Measurement Using the Tracer -----	64
Chapter 3 Advanced Use of ID78K0 -----		65
3.1	Verifying the Validity of Evaluation -----	66
3.1.1	Coverage -----	66
3.1.2	Verifying the Validity of Evaluation Based on Coverage -----	67
3.1.3	Notes on Coverage Results -----	69
3.2	Using External Sense Clips -----	70
3.2.1	Tracing External Data -----	71
3.2.2	Trigger Output -----	72
3.2.3	Real-Time RAM Output -----	73
3.2.4	Creating an Event by ANDing a Data Condition -----	74
3.3	Measuring Time by Setting Conditions -----	75
Appendix A Error Messages -----		77
Appendix B Key Functions -----		87
B.1	Functions of Special Function Keys -----	87
B.2	Functions of Special Function Keys (CTRL + Key) -----	88
Appendix C Menus -----		89

[MEMO]

Chapter 1 Overview

This chapter outlines the debugger.

1.1 Starting and Terminating the Debugger

This section explains how to start and terminate the debugger.

1.2 Making Maximum Use of the Main Window

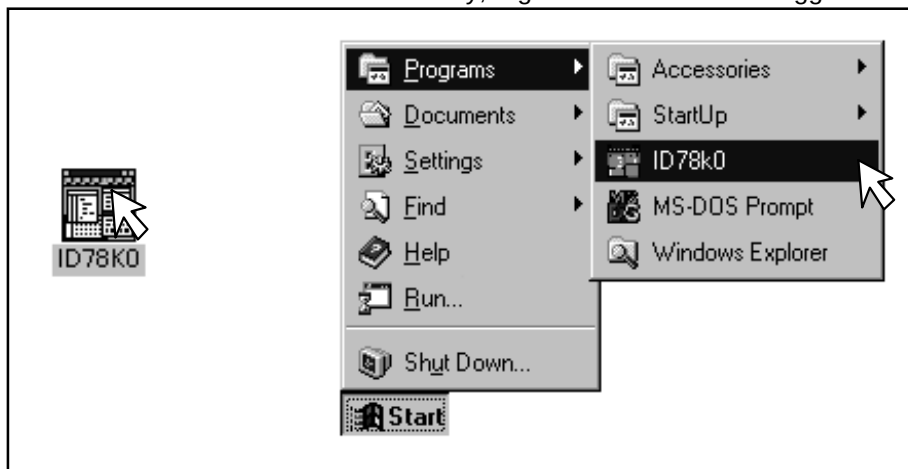
The main window appears when the debugger is started. The main window supports many functions. By making full use of these functions, the efficiency of debugging can be significantly enhanced.

1.1 Starting and Terminating the Debugger

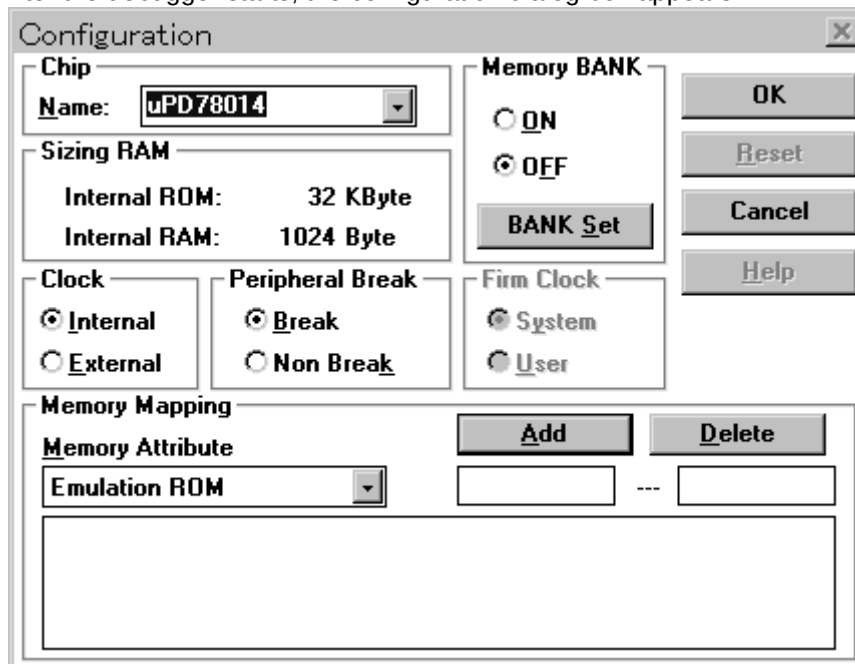
- The debugger can be started and terminated easily.
- To start the debugger, select the icon, shortcut key, or corresponding item in the start menu. These will have been registered when the software was installed.
- To terminate the debugger, select the corresponding item from the menu. When terminating the debugger, you may select saving of the debugging environment. Doing so allows the debugger to be used immediately the next time it is started.

1.1.1 Starting

1. Start Windows.
2. Turn on the in-circuit emulator.
3. Turn on the target, if being used.
4. Double-click the icon or shortcut key, registered when the debugger was installed.

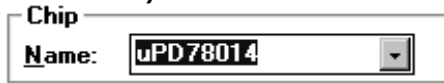


5. After the debugger starts, the configuration dialog box appears.



6. Select a debug target device.

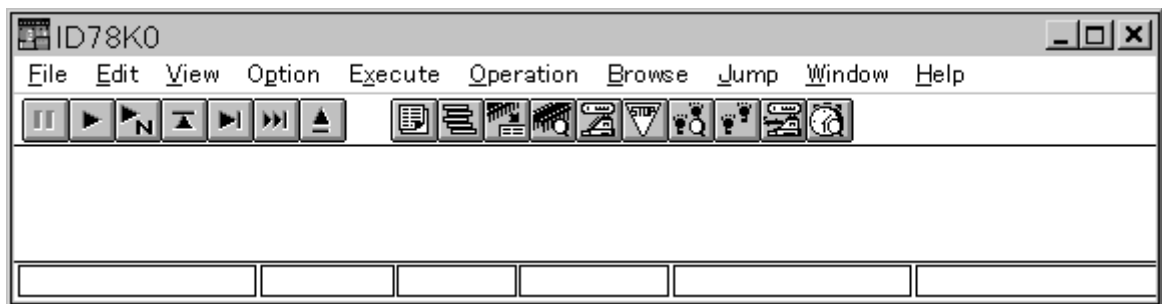
(Note that the debug target device can be selected only when the debugger is being started.)



7. Set the clock source, memory mapping, and other required items.

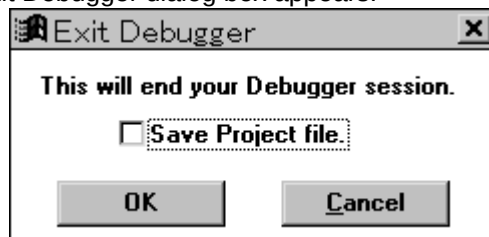
8. Once all the necessary items have been set, click the  button. This completes device initialization and causes the required data to be downloaded to the in-circuit emulator.

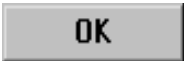
9. Once downloading has been completed, the main window of the debugger opens. The main window is used as the core window for debugging.



1.1.2 Terminating

1. Select File from the menu bar of the main window.
2. Select Exit from the File pull-down menu.
3. The Exit Debugger dialog box appears.



4. Click the  button to terminate the debugger.

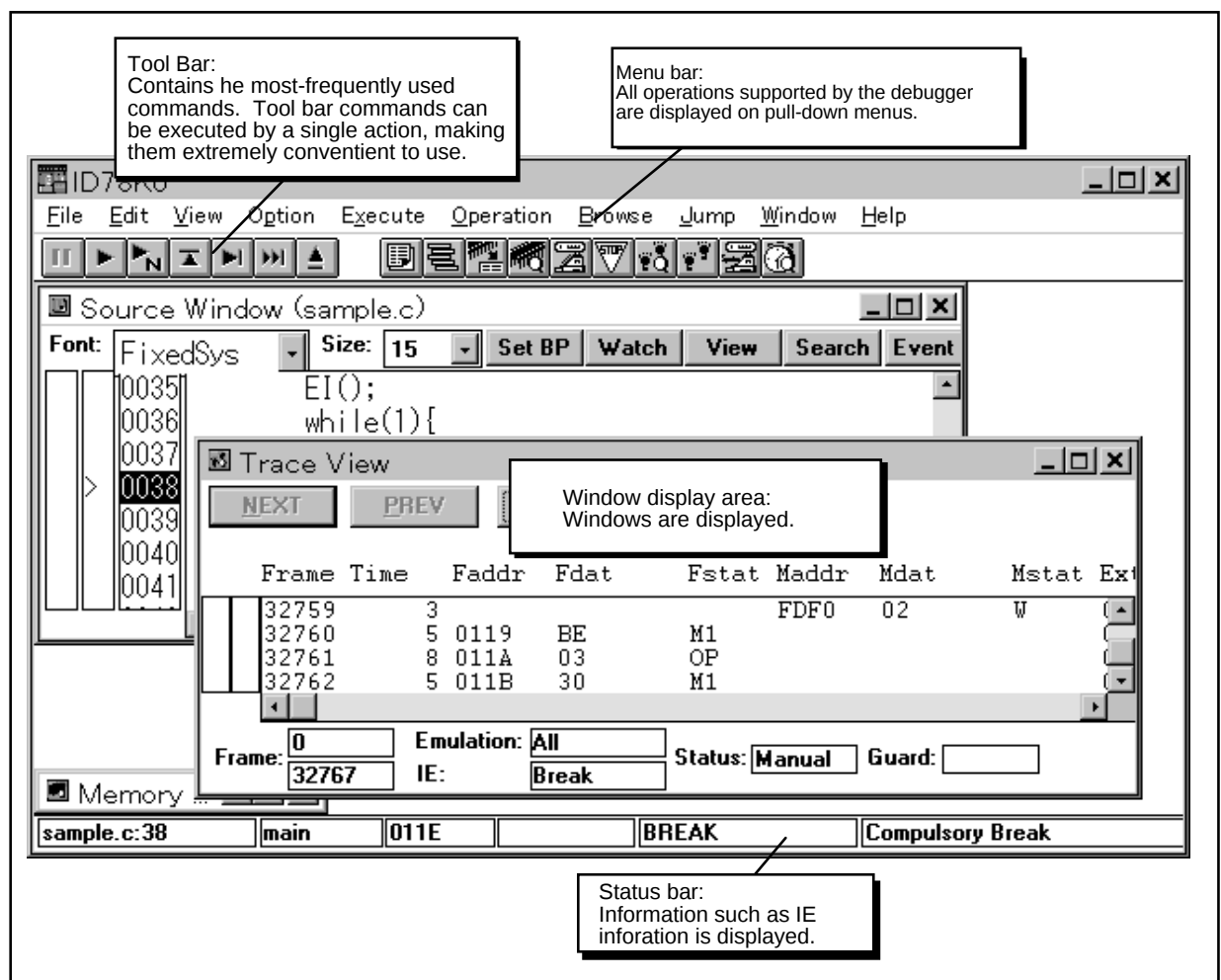
1.2 Making Maximum Use of the Main Window

- All debugger windows are based on the main window.
- The main window supports many functions, all of which are easy to use.

1.2.1 Main Window Functions

- The main window supports four major functions.
- Many debugger operations are performed from the main window. Remember the following four functions.

Function	Description
Menu bar	Contains all the functions supported by the debugger. To perform some operation with the debugger, first check the contents of the menu bar.
Tool bar	Contains the most-frequently used commands. While no target is connected, try clicking each of the buttons, and make a note of the graphic identifying each button.
Window display area	Windows are displayed in this area. These windows include, for example, the Source window and Assemble window, both of which are used whenever debugging is performed.
Status bar	The status of the in-circuit emulator (IE) is displayed in this area. The IE status and break cause are particularly important.



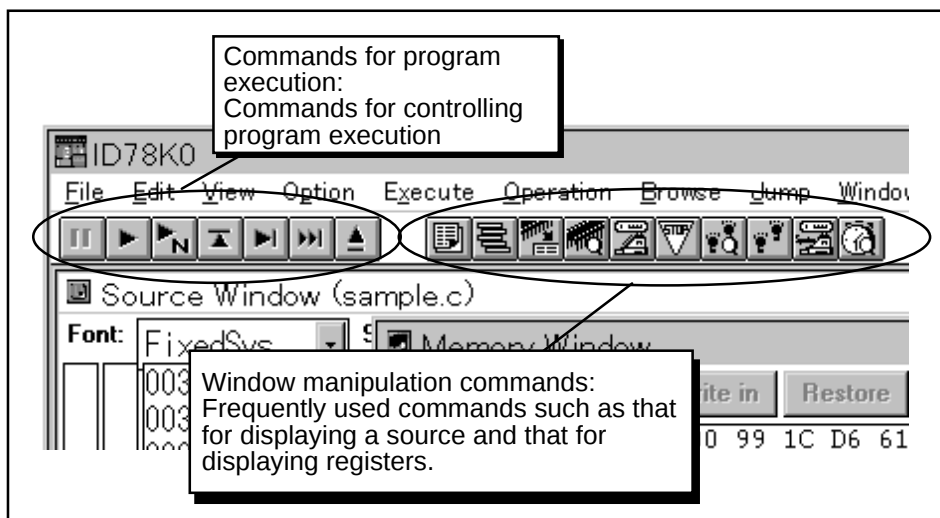
1.2.2 Making Maximum Use of Menus

- The menus contain all the functions supported by the debugger.
- Even when you are not familiar with the debugger's functions, briefly studying each of the pull-down menus will allow you to understand the range of functions available.
- The menus are outlined below.

Menu	Description
<u>F</u> ile	Contains file operation commands. This menu enables the switching of the source displayed in the Source window, the loading and saving of project files, and other file operations.
<u>E</u> dit	Provides commands for copying and pasting displayed data, as well as commands for memory editing.
<u>V</u> iew	Contains display commands. Using this menu, you can retrieve variables, enter the display start address, and display variables.
<u>O</u> ption	Allows you to display and hide the tool bar, status bar, and buttons in each window, and to establish the debugger environment.
<u>E</u> xecute	Contains execution commands. Also, trace mode setting is performed from this menu.
<u>O</u> peration	Allows you to perform window mode switching, and to specify connection to the trace window.
<u>B</u> rowse	Contains the commands used to open each window. From this menu, you can display windows such as the event and coverage windows.
<u>J</u> ump	Allows you to jump to the source window, assemble window, and memory window.
<u>W</u> indow	Allows you to specify how windows are to be displayed, the arrangement of icons, and also enables switching between windows.
<u>H</u> elp	Displays help information.

1.2.3 One-Touch Tool Bar Operation

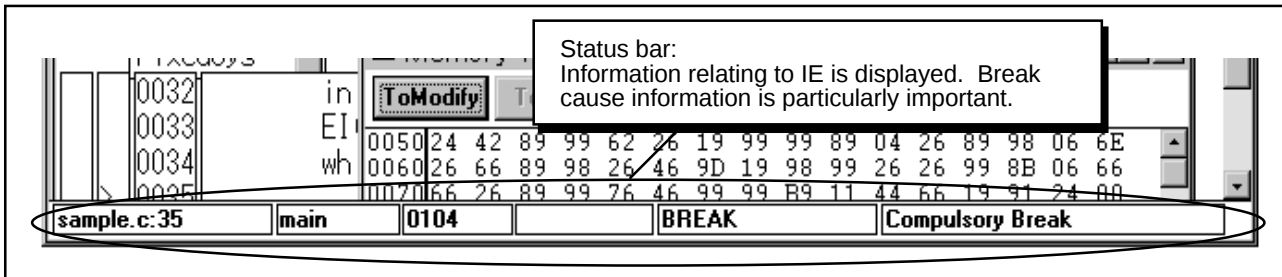
- The tool bar consists of buttons which correspond to frequently used commands. Commands are executed simply by clicking the corresponding button.
- The function of each button is identified by a suitably representative graphic.
- The commands assigned to the tool bar buttons can also be executed from the menu bar.



Display	Description
	Stops user program execution.
	Executes a user program. As soon as the break conditions are satisfied, the user program terminates.
	Executes a user program. Even when break conditions are satisfied, the user program does not terminate.
	Executes the program in real time, until execution returns to the calling function.
	Executes the program, step by step. Every time this button is clicked, one step of the program is executed. For source level debugging, one step corresponds to one line. For instruction level debugging, one step corresponds to one instruction.
	Performs Next step execution of the program. Every time this button is clicked, one step of the program is executed, by means of Next step execution. For source level debugging, one step corresponds to one line. For instruction level debugging, one step corresponds to one instruction.
	Initializes the debugger or emulation CPU. Opens the Reset Debugger dialog box.
	Displays the source text. Opens the Source window.
	Displays the stack contents. Opens the Stack window.
	Displays a disassembled program. Opens the Assemble window.
	Displays the contents of memory. Opens the Memory window.
	Displays the register contents. Opens the Register window.
	Registers and sets break events. Opens the Break dialog box.
	Displays trace results. Opens the Trace View window.
	Registers and sets trace events. Opens the Trace dialog box.
	Displays the SFR contents. Opens the SFR window.
	Displays timer measurement results. Opens the Timer window.

1.2.4 Using Information Provided by the Status Bar

- The status bar displays important data including, for example, the status of the IE and the cause of a break.
- If a break occurs at a point where no break has been set, or if no source appears when a break occurs, for example, check this area first.



- Source file name:** Displays the source file name and source line number corresponding to the indicated PC value. If no file information is available, "---" is displayed.
- Function name:** Displays the function name corresponding to the indicated PC value. If no function information is available, "---" is displayed.
- PC value:** Displays the current PC value.
- CPU status:** Displays the status of the CPU (μ PD780xx: target device).

CPU status	Description
TARGET	The target is on.
HOLD	Bus hold mode
STANDBY	Halt or stop mode
LUTCHUP	Latch-up has been detected. Turn off the target and in-circuit emulator immediately.

- IE status:** Displays the operation status of the in-circuit emulator.

IE status	Description
RUN	Real-time execution in progress
STEP	Step-by-step execution in progress
BREAK	Break status
TRACE	Tracing in progress
TIMER	Timer measurement in progress

- Break cause:** Displays the reason for a break. The table below lists possible break causes.

Cause	Description
Compulsory Break	Normal break (manual break)
Temporally Break	Normal break (break caused by internal processing)
Event Break	Break triggered by an event
Out Of Range Break	Break caused by procedure step termination
Trace Full Break	Break caused by trace full state
Non Map Break	Access to a non-mapped area was attempted.
SFR Illegal	Illegal access to an SFR was attempted.
Stack Overflow	Break caused by stack overflow
Write Protect	An attempt was made to write to a write-protected area.

Chapter 2 Basic Operations

This chapter explains the basic operations of the ID78K0.
Each section clarifies how windows are related to each other.

2.1 Establishing the Environment

Explains how to establish a debugging environment.

2.2 Source Level Debugging

Explains the use of the Source window to debug a source program.

2.3 Instruction Level Debugging

Explains the use of the Assemble window to perform assembler level debugging.

2.4 Manipulating Memory

Explains the use of the Memory window to perform modification, initialization, and other operations on memory.

2.5 Manipulating Registers

Explains the functions of the Register window, used to manipulate general-purpose registers, and those of the SFR window, used to manipulate SFRs.

2.6 Creating Events

Events are very useful for debugging. Events can be used for program and trace control. This section explains how to set an event.

2.7 Manipulating Symbols (Variables)

The debugger supports the input of symbols as data. This section explains how to enter symbols and display variables.

2.8 Using the Tracer Effectively

The IE-78000-R-A contains 32K frames of trace memory. The tracer is used to trace data, making it very useful for detecting program problems. This section explains the use of the tracer.

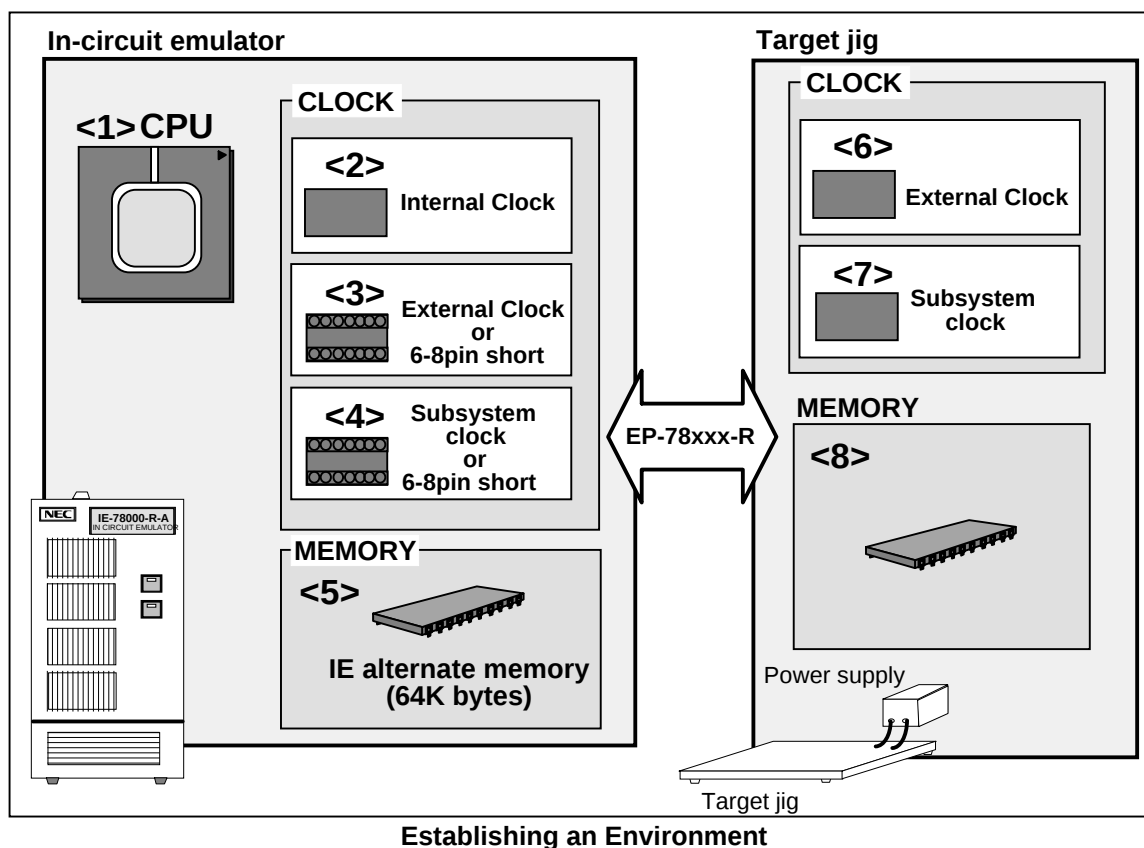
2.9 Measuring the Execution Time

Explains the time required to execute a program from beginning to end, and time tags written in the tracer.

2.1 Establishing the Environment

- Establishing an environment allows the debugger to recognize the configuration of a target system. Establishing a debugging environment enables the maximum utilization of the debugger functions.
- The environment must be established whenever the debugger is started.
- Once an environment has been established, it can be saved to a file, subsequently eliminating the need to newly establish the environment. When the debugger is next started, the environment can be established simply by loading the file (project file).
- For the ID78K0 operating environment, set the following items:

Item	Setting window	Location in environment setup diagram	Remarks
Device	Configuration dialog box	<1> CPU	Can be set only when the debugger is being started
CPU clock		<2> <3> <4> <6> <7> CLOCK	
Peripheral equipment operation		<1> CPU	
Memory bank switching	Configuration dialog box Bank Set dialog box	<1> CPU <8> MEMORY	Can be set at any time
Alternate operation clock	Configuration dialog box	<1> CPU	
Memory mapping	Configuration dialog box	<5> <8> MEMORY	
Mask option	Mask Option dialog box	<1> CPU	



2.1.1 Selecting a Device

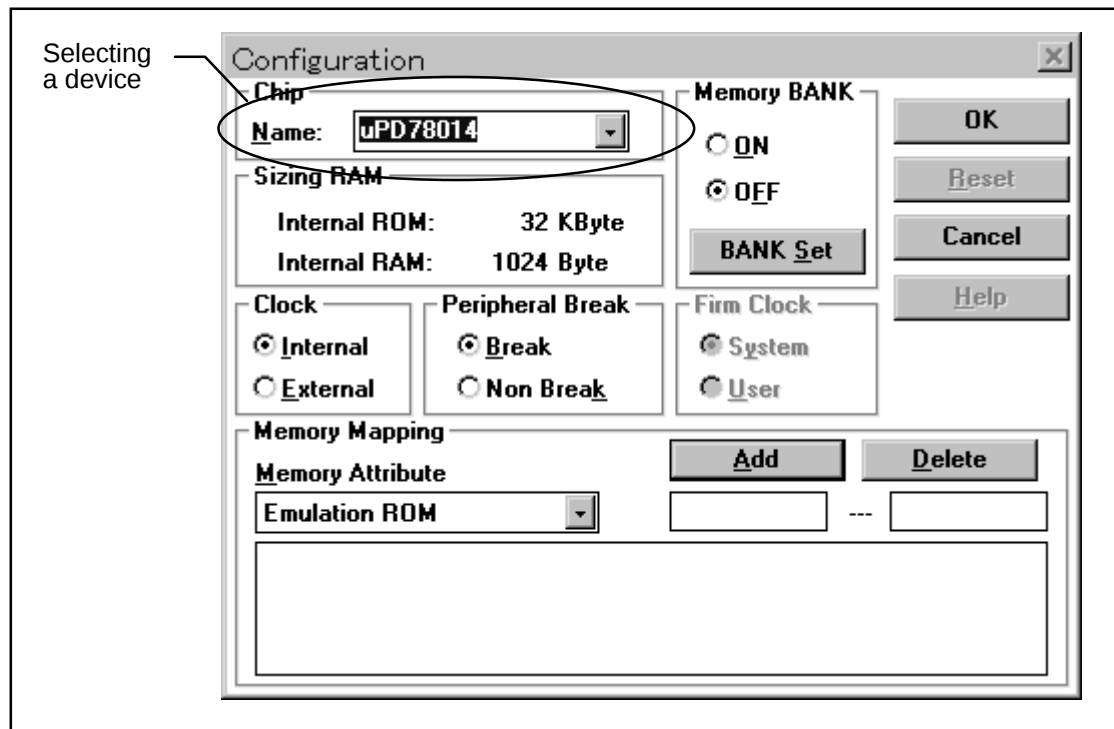
- A device can be selected in the Configuration dialog box that appears when the debugger is started. Note that once the debugger has started, this selection cannot be changed.

Setting in the Configuration dialog box:

1. Start the debugger.
2. Select a device in the Configuration dialog box that appears when the debugger is started.

When the main window has already been opened:

1. Terminate, then restart, the debugger.



2.1.2 Selecting a CPU Clock

- A CPU clock is selected in the Configuration dialog box that appears when the debugger is started. Note that once the debugger has started, the CPU clock cannot be changed.

Setting in the Configuration dialog box:

- Start the debugger.
- Change the CPU clock in the Configuration dialog box that appears when the debugger is started.

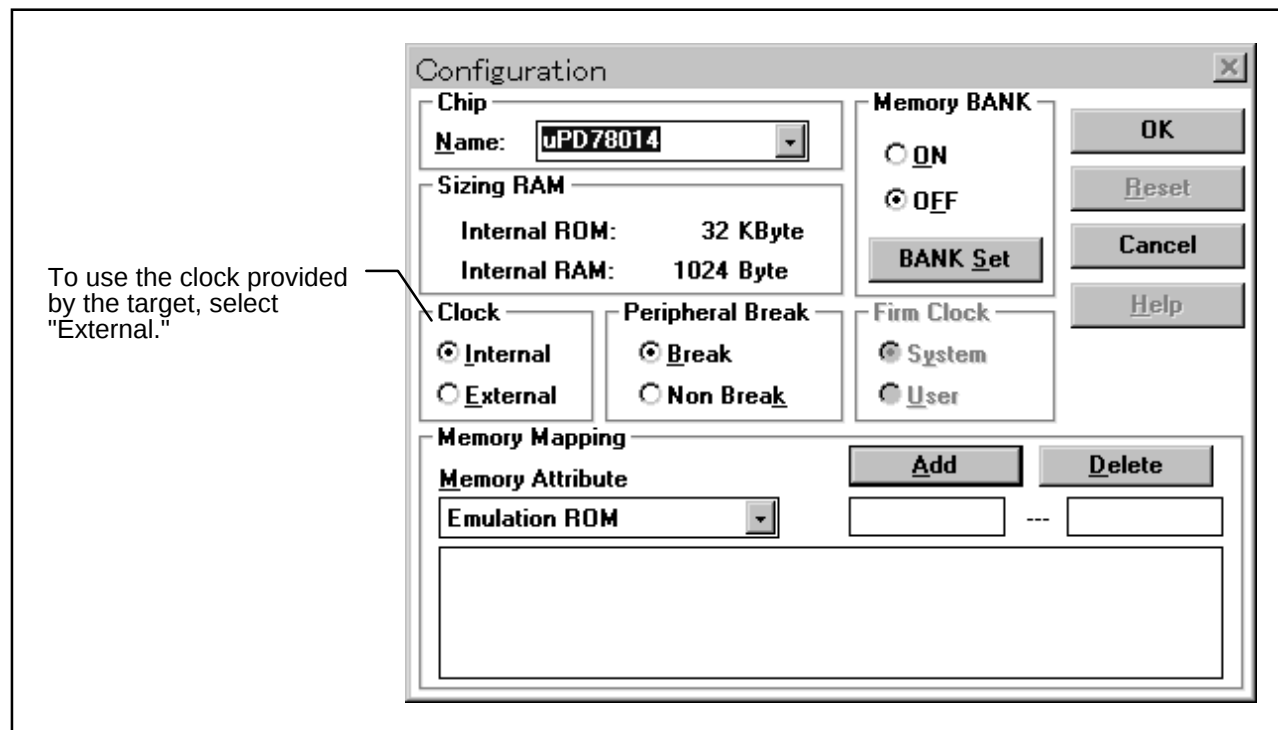
When the main window has already been opened:

- Terminate, then restart, the debugger.

- When "Internal" is selected as the CPU clock, the clock provided by the in-circuit emulator is used as the CPU clock. This clock frequency is determined by the connected emulation board (EM board).

Emulation board(Note)	CPU clock frequency when "Internal" is selected
IE-78014-R-EM	8.38 MHz
IE-78014-R-EM-A	
IE-780208-R-EM	4.19 MHz
IE-78044-R-EM	
IE-78064-R-EM	5.0 MHz
IE-78078-R-EM	
IE-78098-R-EM	6.0 MHz

Note For emulation boards not listed here, refer to the manual provided with the board.



2.1.3 Mapping

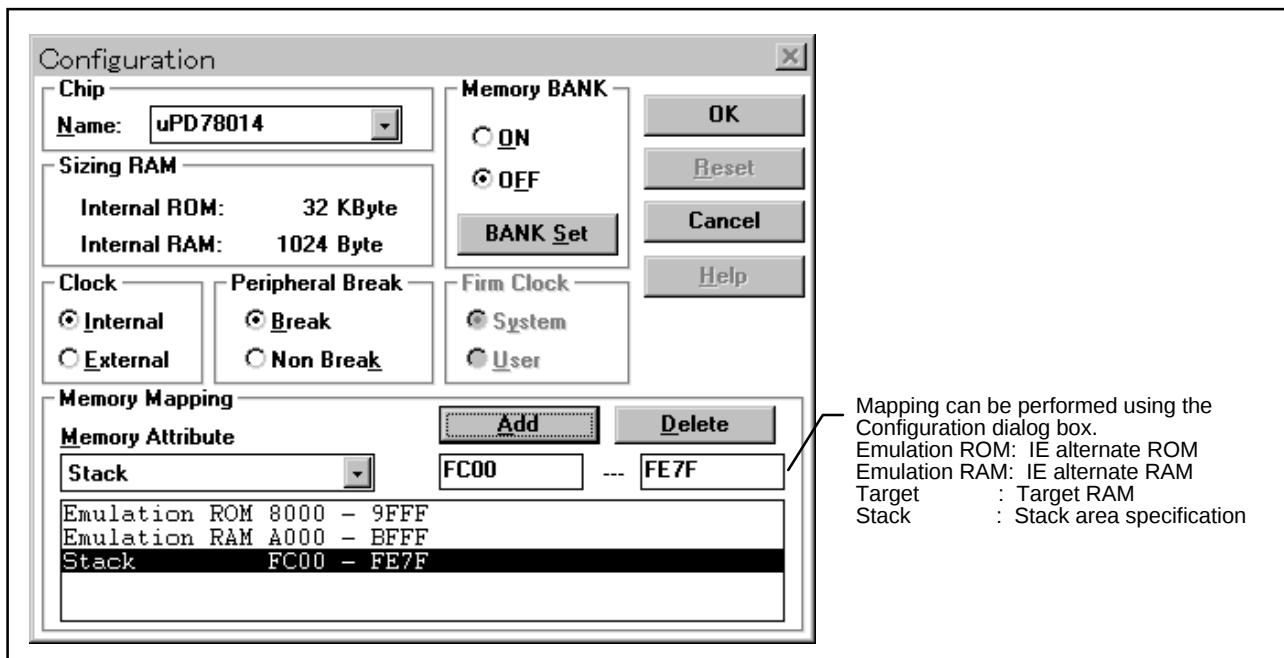
- When external ROM/RAM is used in addition to internal ROM and internal RAM (including SFRs and registers), the area to be mapped must be set.

To add an area to be mapped:

- Open the Configuration dialog box. This dialog box appears when the debugger is started. It can also be displayed by selecting Option -> Configuration... from the menu bar.
- Set the Memory Attribute area, then click the **Add** button.

To delete a mapped area:

- Open the Configuration dialog box. This dialog box appears when the debugger is started. It can also be displayed by selecting Option -> Configuration... from the menu bar.
- Select the mapped area to be deleted, then click the **Delete** button.



2.1.4 Specifying a Stack Area

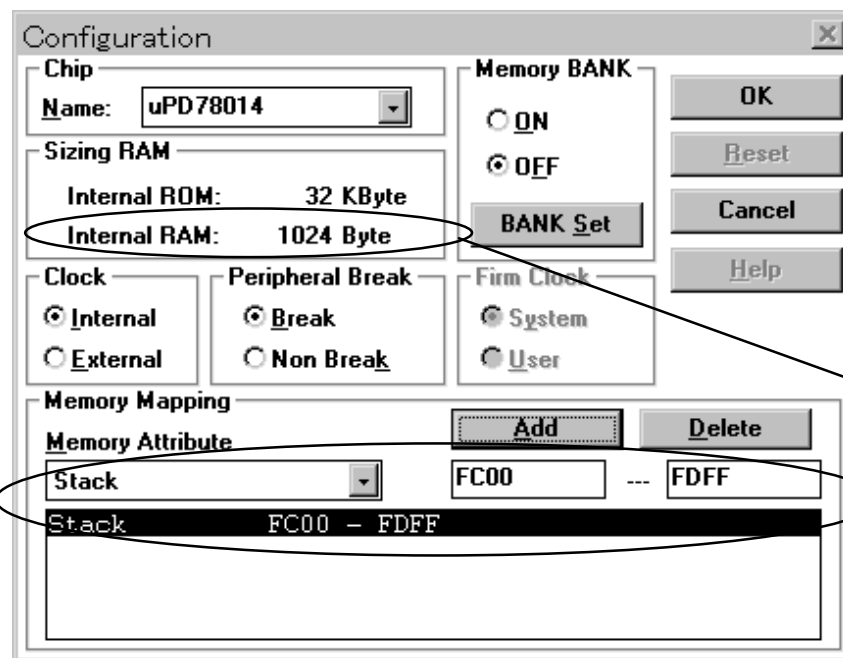
- To monitor stack operation, specify a stack area.
- When a stack area has been specified, any stack operation (CALL, RET, PUSH, POP) performed outside the set area is detected as being an illegal access.
- An area in internal high-speed RAM can be specified as the stack area.
- When no stack area is specified, the entire internal high-speed RAM area is used as the stack area.

To specify a stack area:

1. Open the Configuration dialog box. This dialog box appears when the debugger is started. It can also be displayed by selecting Option -> Configuration... from the menu bar.
2. Set the Memory Attribute area, then click the **Add** button.

To cancel the stack area specification:

1. Open the Configuration dialog box. This dialog box appears when the debugger is started. It can also be displayed by selecting Option -> Configuration... from the menu bar.
2. Select the mapped stack area to be deleted, then click the **Delete** button.



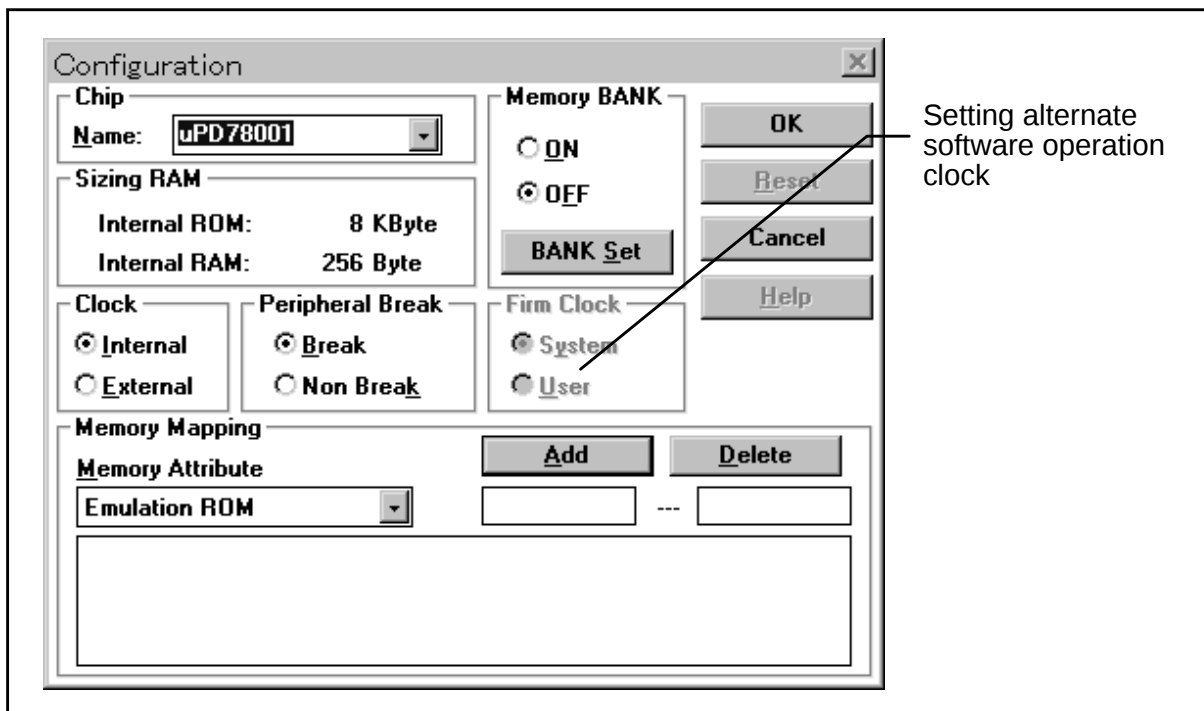
The μ PD78014 contains 1024 bytes of internal high-speed RAM between addresses 0xfb00 and 0xfeff. Therefore, set the area to be mapped within this range.

2.1.5 Setting the Alternate Software Operation Clock

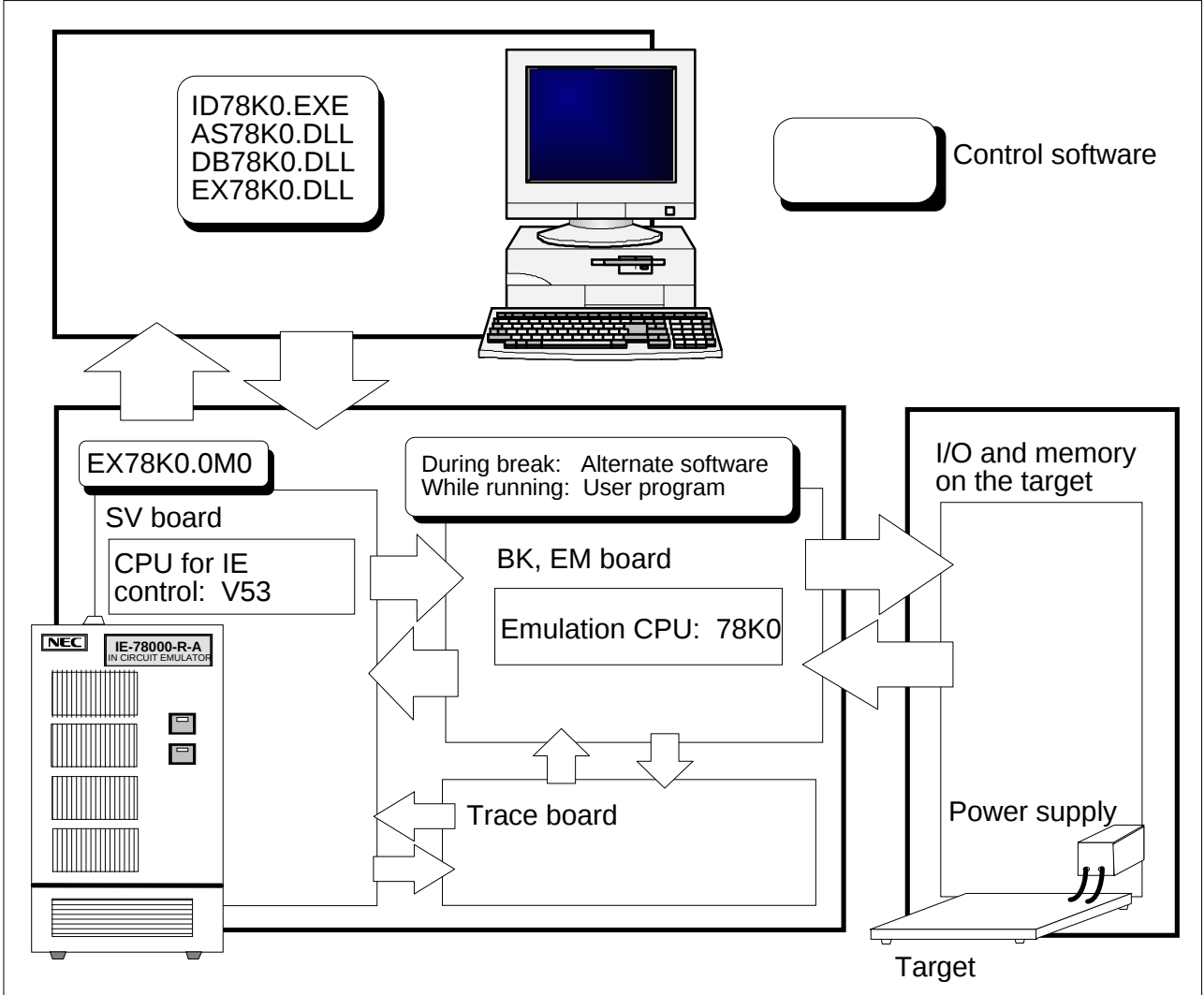
- The alternate software is control software that runs on the 78K0 device. It controls the resources (register values, SFRs, and memory) of the target while the in-circuit emulator is in break mode.
- The alternate software accesses the target resources directly.
- The operation of the alternate software uses the same clock as the user program. If, therefore, the user program uses a low-speed clock while the in-circuit emulator is in break mode, the alternate software will also operate slowly, thus lowering the overall speed of debugger operation. To avoid this, specify the use of the alternate software operation clock.

To set the operation clock:

1. Open the Configuration dialog box. This dialog box appears when the debugger is started. It can also be displayed by selecting Option -> Configuration... from the menu bar.
2. Select the operation clock in the alternate software operation clock selection area.



The alternate software operating environment is illustrated below.

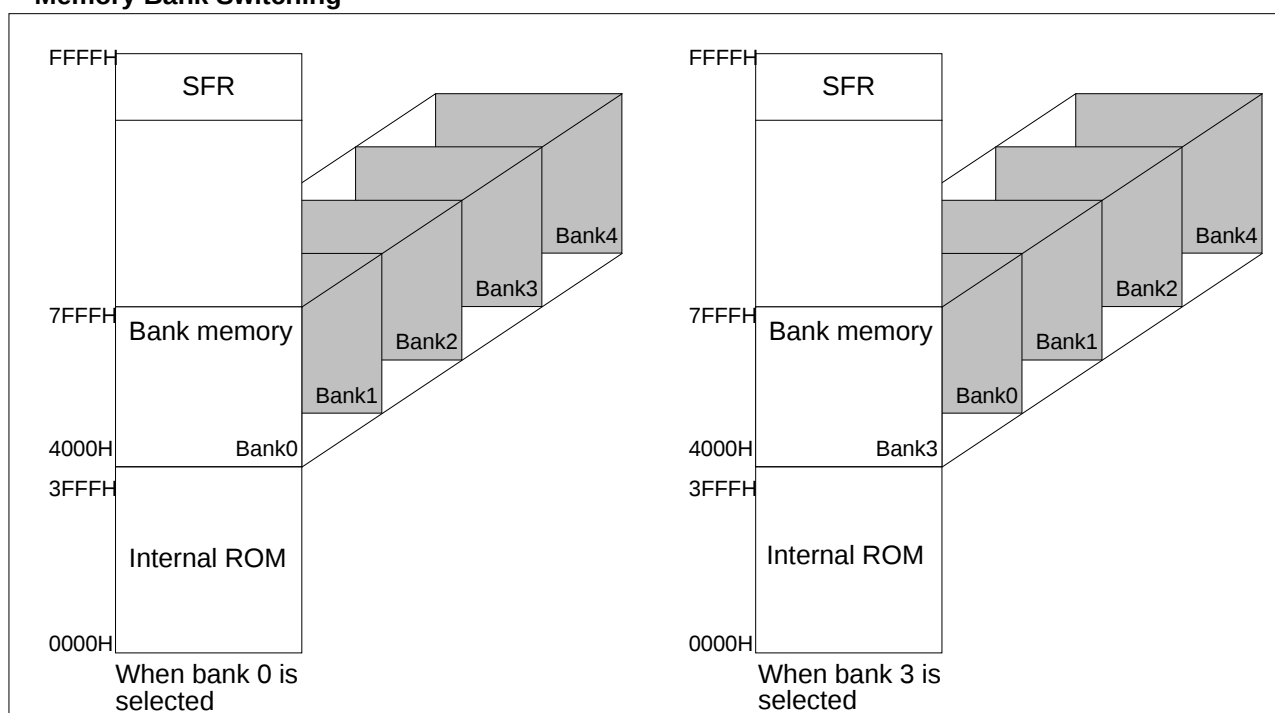


2.1.6 Setting Memory Banks

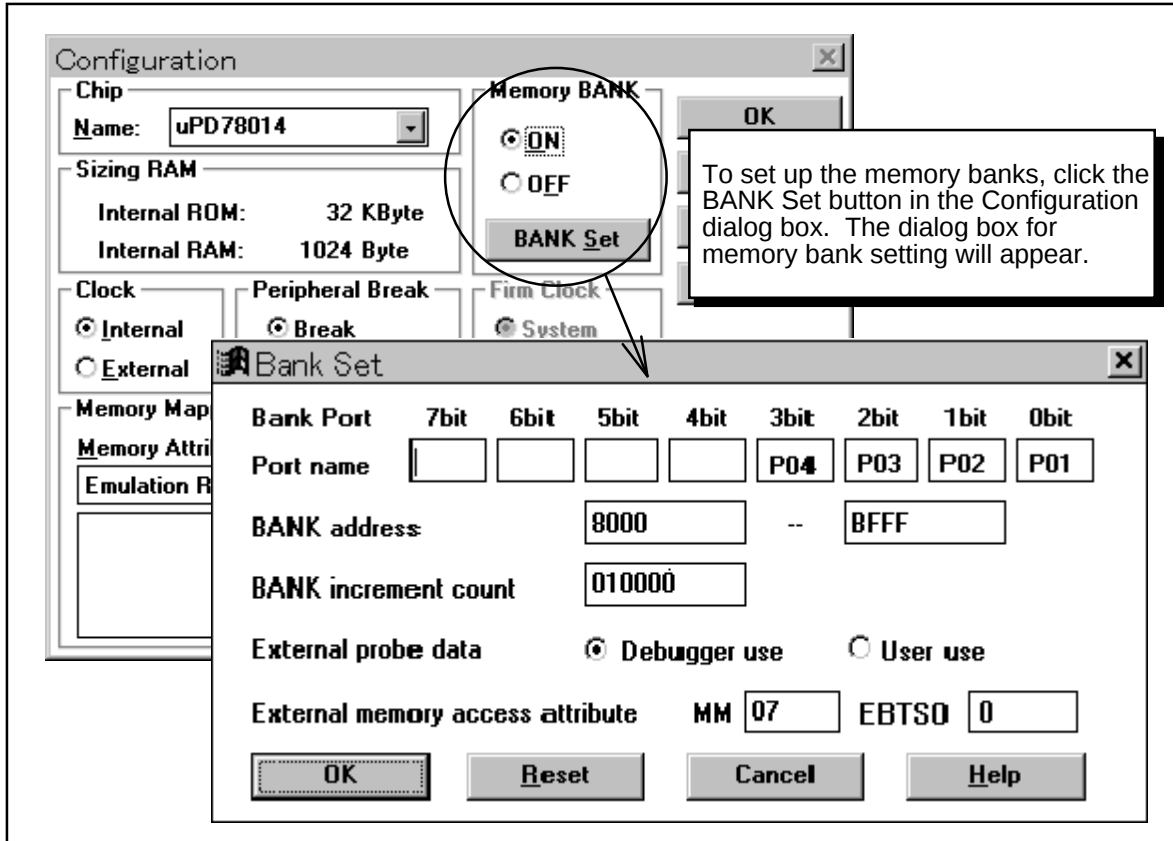
- The 78K0 series has an address space consisting of up to 64K bytes between addresses 0 and 0xffff.
- A program of 64K bytes or more can be run by switching part or all of the program area between addresses 0 and 0xffff.
- The structure of the memory banks is shown below.

In the following figure, the 16K-byte space between addresses 4000h and 7fffh is used for the memory banks. Five banks, 0 to 4, are used. When bank 0 is selected, data in bank 0 can be accessed at addresses 4000h to 7fffh, a linear space existing between address 0h and 7fffh. When bank 0 is selected, the spaces corresponding to banks 1 to 4 cannot be accessed. When bank 3 is selected, the data in bank 3 can be accessed between addresses 4000h and 7fffh. The other banks cannot be accessed.

Memory Bank Switching



- To switch between memory banks, program the generation of the upper address by using, for example, ports.
- So that the debugger can control the memory banks effectively, set the ports and other data to be used for bank switching when establishing the environment.

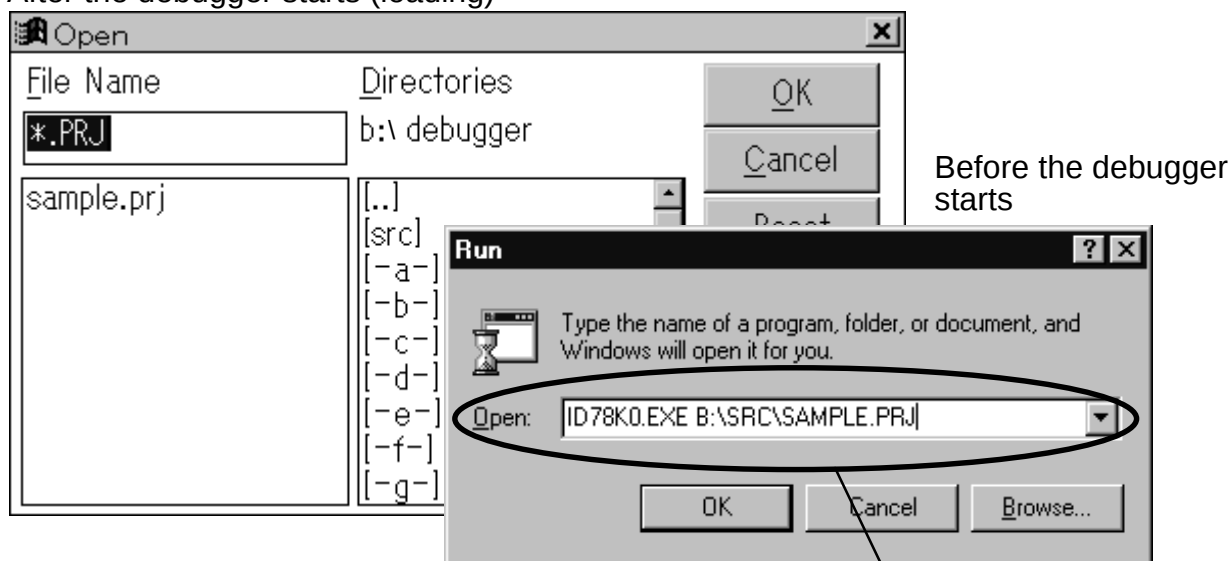


2.1.7 Loading/Saving the Debugging Environment

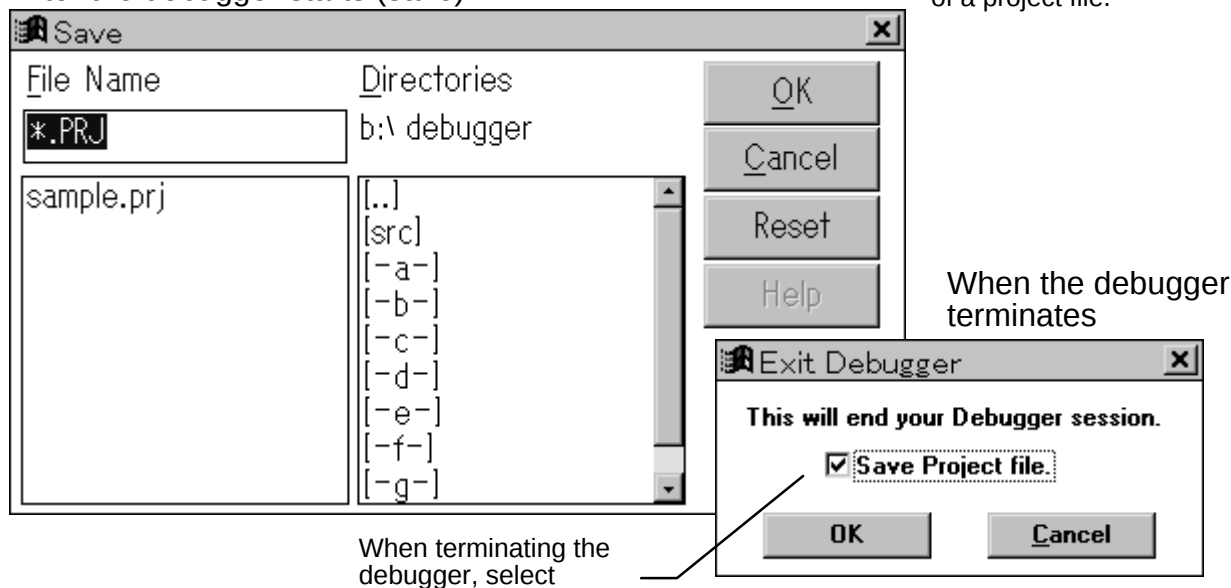
- Saving the debugging environment into a project file enables subsequent debugging to be performed in exactly the same environment.

Debugging environment		Method
Load	At start	Specify a project file to be read, using its full path name, as a start option.
	After start	Load a project file by using the Project file load dialog box.
Save	After start	Save a project file using the Project file save dialog box.
	At exit	Select "Save Project File" in the Exit Debugger dialog box, then terminate the debugger.

After the debugger starts (loading)



After the debugger starts (save)



Data to be loaded/saved

Window	Data
Configuration dialog box	All items
Bank Set dialog box	All items
Main window	Setting information
Load Module dialog box	File information downloaded
Extended Option dialog box	Setting information
Mask Option dialog box	Setting information
Source Path dialog box	Source path information
Source window	Window display information, font information
Assemble window	Window display information, display start address
Memory window	Window display information, display start address
Stack window	Window display information
SFR window	Window display information
Local Variable window	Window display information
Trace View window	Window display information
Show Trace dialog box	Setting information
Snap Trace dialog box	Setting information
Event Manager	Window display information, all event information
Event Link dialog box	Window display information
Break dialog box	Window display information
Trace dialog box	Window display information
Snap-Shot dialog box	Window display information
Event Set dialog box	Window display information
Register window	Window display information, displayed bank
Variable window	Window display information, displayed variable information
Coverage window	Window display information

2.2 Source Level Debugging

- ID78K0 can set breakpoints and display variables for a source.
- Many source level operations are supported, thus greatly enhancing debugging efficiency.
- Source level debugging can be performed by loading a file containing source information.
- Source level debugging is particularly effective for debugging programs written in C or structured assembly language.

2.2.1 Notes on Compilation, Assembly, and Linking

- When source level debugging is performed, the file to be loaded must contain source debugging information.
- Source debugging information is included in the object by specifying the option for adding debugging information at assembly or compile time.
- The following shows how to set options at compilation, assembly, and linking:

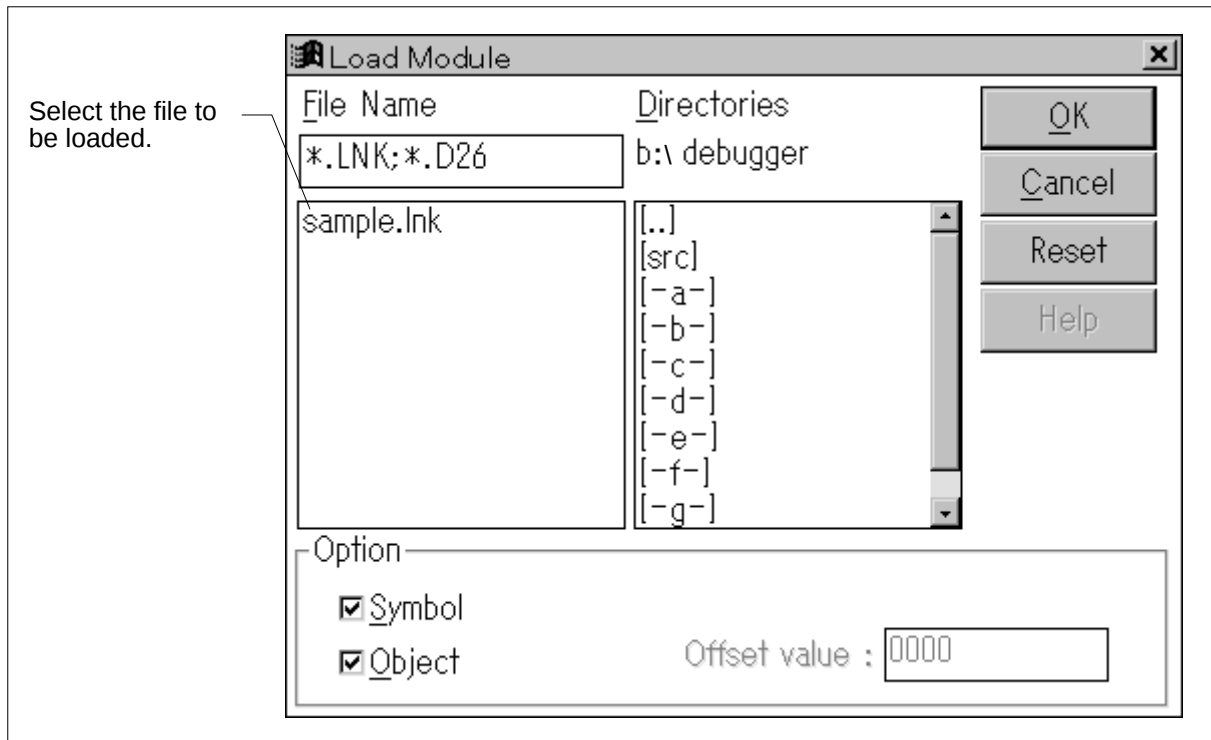
Type of source to be debugged		Required action
C program	Without in-line assembly description	Specify the -G option at compile time.
	With in-line assembly description	1. At compile time, specify the -a option to output an assembly source file. 2. Assemble the source generated in 1, above, without specifying any debug options (-GA, -NGA).
Structured assembly language program		1. Specify the -GS option at structured assembly. 2. Assemble the source generated in 1, above, without specifying any debug options (-GA, -NGA).
Assembly language program		Specify the -GA option at assembly.
Link		Specify the -G option at linking.

2.2.2 Downloading a Program

- Load module files and hexadecimal files can be downloaded.
- When a downloaded file contains source debugging information, source level debugging can be performed.

To perform downloading:

1. Select File -> Download... from the menu bar to open the Load Module dialog box.
2. Load the desired file.



2.2.3 Displaying a Source

- After a load module file containing source debugging information has been downloaded, the source can be displayed.
- If the source file is stored in a directory other than that containing load module file, or if the source file is stored in more than one directory, source path information must be provided to the debugger.

To display a source:

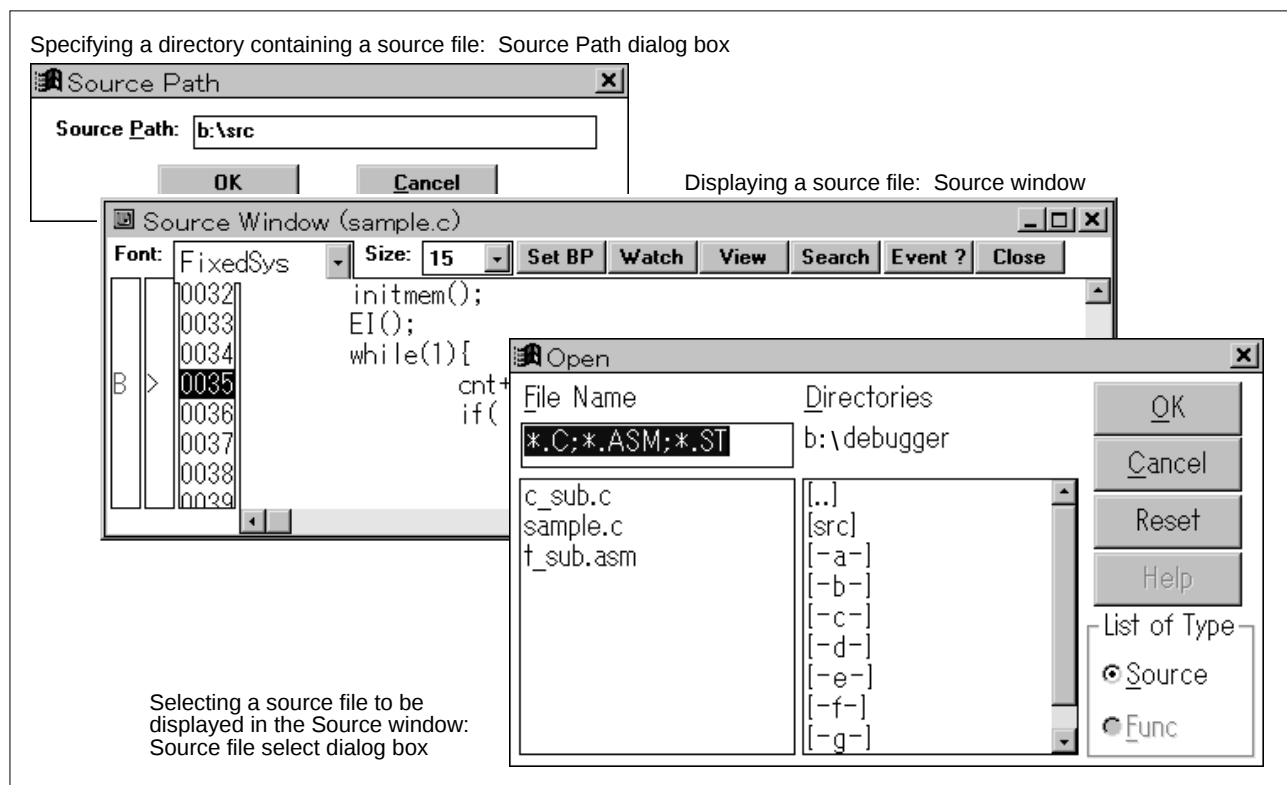
1. Select **B**rowse -> **S**ource Text... from the menu bar or click the  button to open the Source window.

To change the source file displayed in the Source window:

1. Activate the Source window.
2. Select **F**ile -> **O**pen... from the menu bar to open the Source file select dialog box.

When a source file is stored in another directory or in more than one directory:

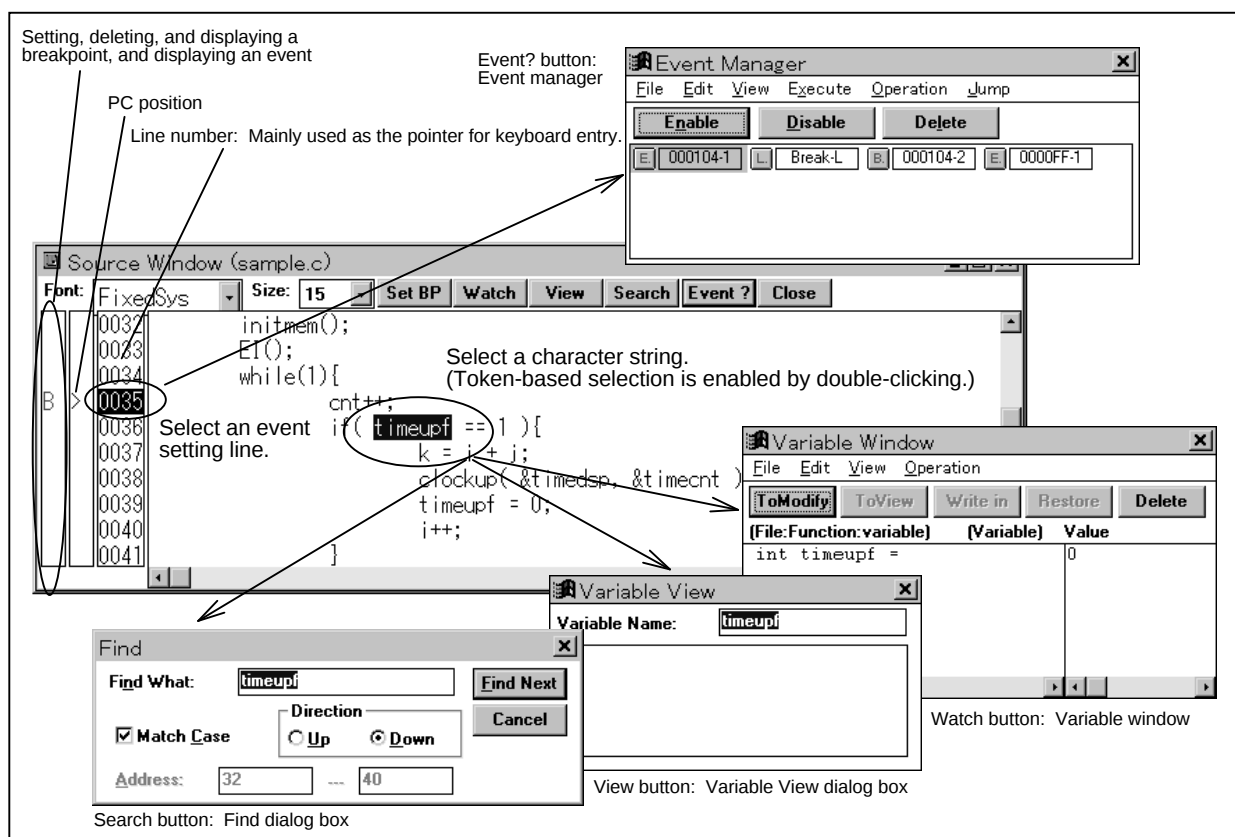
1. Select **O**ption -> **S**ource **P**ath... from the menu bar to open the Source Path dialog box.



2.2.4 Functions Supported by the Source Window

- The Source window Supports a wide range of functions, such as the setting of breakpoints and the addition of variables to be displayed.
- The supported functions are listed below:

Function	Procedure	
	Using the mouse	From the keyboard
Setting/deleting a breakpoint	Click the point mark area.	1. Select a line number (with the mouse). 2. Select Execute -> S et B P from the menu bar. (CTRL + B)
Setting PC	---	1. Select a line number (with the mouse). 2. Select Execute -> S et P C from the menu bar. (CTRL + E)
Displaying a variable	1. Select a variable. 2. Click the Watch button then the View button.	1. Select a variable (with the mouse). 2. Select V iew -> W atch Variable... or V iew Variable... from the menu bar.
Retrieving a character string	1. Select a variable. 2. Click the Search button.	1. Select a variable (with the mouse). 2. Select V iew -> S earch... from the menu bar.
Checking an event	1. Select an event line. 2. Click the Event ? button.	1. Select an event line (with the mouse). 2. Select V iew -> E vent? from the menu bar.



2.2.5 Jump from the Source Window

- Jump from the Source window to the Assemble window and Memory window is supported.
- Using the jump function, it is easy to check the source text assemble results.
- Select a source line number as the jump destination. Then, the start address of the selected source line is set as the jump pointer.

Jump destination	Procedure
Assemble window	1. Select a source line number. 2. Select <u>J</u> ump -> <u>A</u> ssemble... from the menu bar.
Memory window	1. Select a source line number. 2. Select <u>J</u> ump -> <u>M</u> emory... from the menu bar.

The jump destination is the address of the source line, displayed in reverse video: In the following example, the start address of the 38th line is set as the jump pointer.

The screenshot displays three debugger windows:

- Source Window (sample.c):** Shows C code with line 0038 selected. The code is:


```

0034 while(1){
0035     cnt++;
0036     if( timeupf == 1 ){
0037         k = i + j;
0038         clockup( &timedsp, &timecnt );
0039         timeupf = 0;
      
```
- Assemble Window (0132):** Shows the assembly code corresponding to the selected source line. The address 0132 is selected. The assembly code is:

Event Adr.	Label	Data	Mnemonic
0132		1400FB	MOVW DE, #0FB00H
0135		B5	PUSH DE
0136		1480FE	MOVW DE, #0FE80H
0139		B5	PUSH DE
013A		9A0002	CALL !_clockup
013D		B0	POP AX
- Memory Window:** Shows a memory dump. The address 0130 is selected. The memory dump is:

Address	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex
0100	7A	1E	AE	02	30	AE														
0110	01	82	BE	01	30	97														
0120	01	00	BD	3C	AE	08														
0130	BE	06	14	00	FB	B5														
0140	00	00	03	BA	FB	AE	0A	30	AE	0B	80	BE	0B	30	BE	0A				
0150	02	0C	FB	EA	01	00	BD	21	AE	0A	61	01	30	AE	0B	27				
0160	BE	05	30	BE	04	9A	03	04	10	00	00	03	0C	FB	AE	08				

Arrows indicate the jump from the Source window to the Assemble window and from the Memory window to the Assemble window.

Jump to the Memory window

Jump to the Assemble window

2.3 Instruction Level Debugging

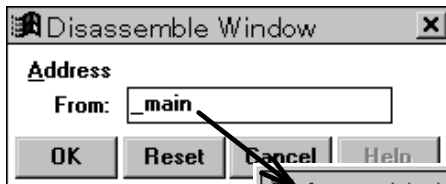
- The contents of memory can be displayed, modified, and retrieved in assembly language.
- Instruction level debugging supports a higher level of precision than source level debugging.
- Assembly language code can be displayed in the Assemble window and Trace View window. This section mainly explains the operations supported by the Assemble window.

2.3.1 Assembly Language Display and Online Assembly

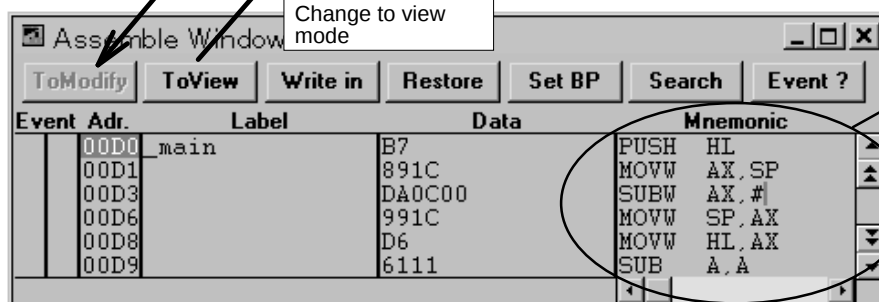
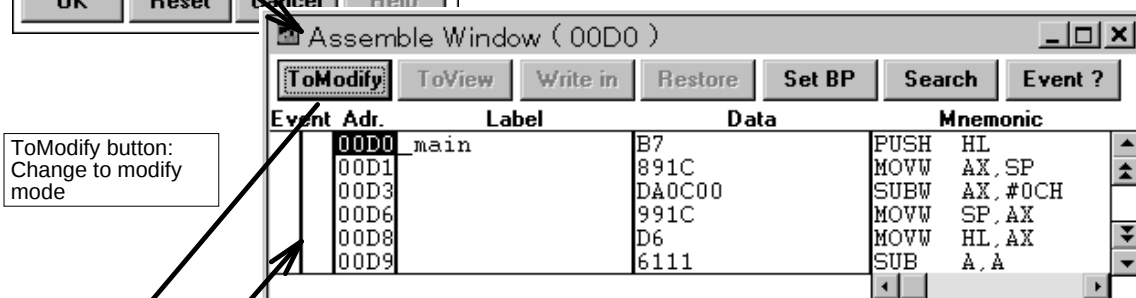
- The Assemble window allows you to view assembly language code and perform online assembly.
- With the online assemble function, patching can be performed. Simple bugs can be corrected and confirmed immediately.

Assemble		Procedure
Display	To select a displayed address	Open the Addressing dialog box in either of the following two ways: 1. Select B rowse -> A ssemble... from the menu bar. 2. Click the  button.
	To display instructions starting from an address selected in another window (such as the Source, Memory, or Register window)	1. Select an address to be used as the display pointer. 2. Select J ump -> A ssemble... from the menu bar.
Modification		1. Open the Assemble window. 2. Click the  button to enter modify mode. 3. Position the cursor to the mnemonic display/modification area, then correct the program. 4. After completing the correction of the program, click the,  button to rewrite the program. 5. Click the  button to enter view mode.

Specifying the start address for display: Addressing dialog box



Assembly language display: Assemble window (display mode)



Program correction: Assemble window (modify mode)

2.3.2 Saving and Referencing Displayed Assembly Language Code

- The displayed assembly language code can be saved to a file. The saved file can subsequently be referenced.
- The file is saved in text format, such that any commercially available editor can be used to view its contents.

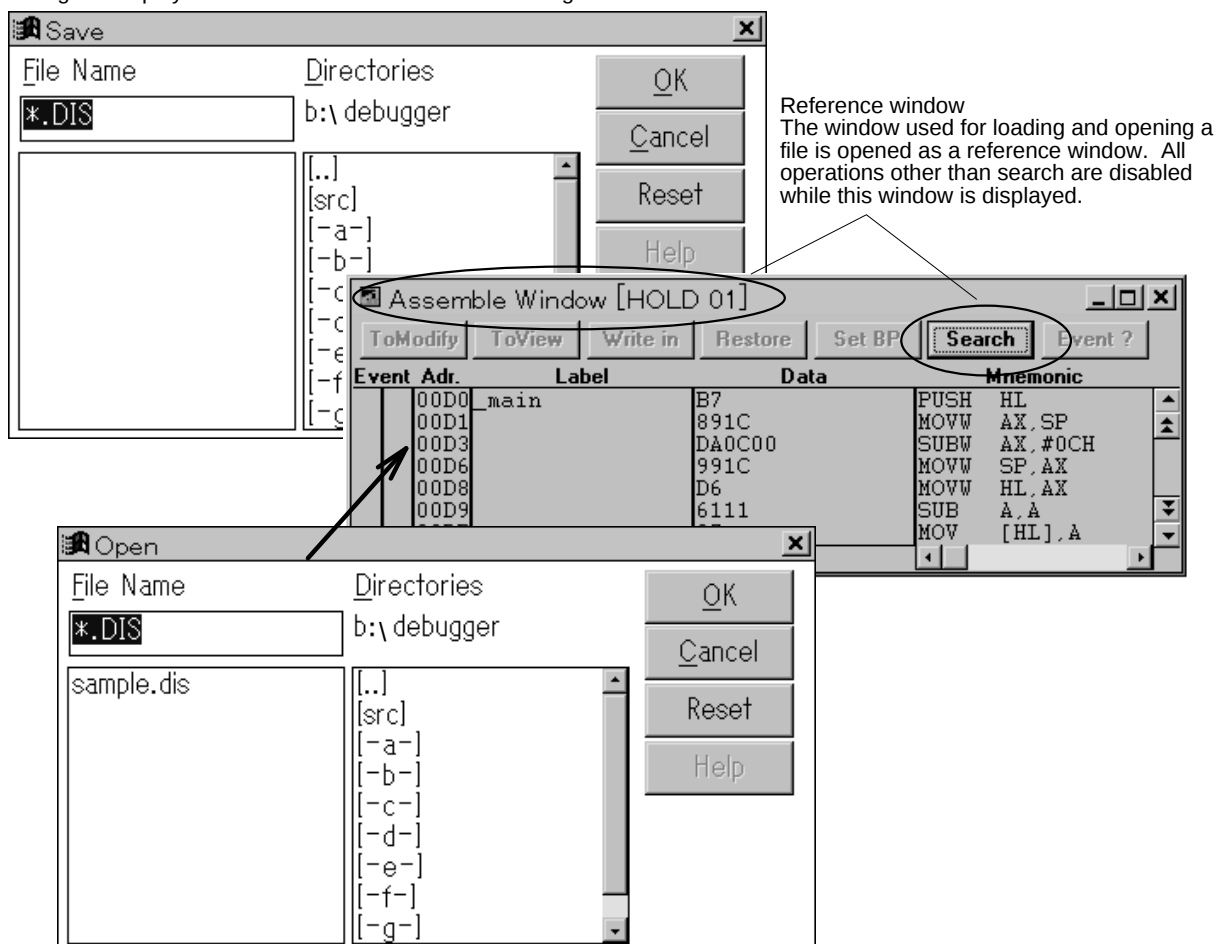
To save displayed assembly language code to a file:

1. Activate the Assemble window.
2. Select File -> Save As... from the menu bar.
3. Save the displayed assembly language code using the View file save dialog box.

To open and reference the saved file:

1. Activate the Assemble window.
2. Select File -> Open... from the menu bar.
3. Load the file to be referenced using the View file load dialog box.

Saving the display contents to a file: View file save dialog box

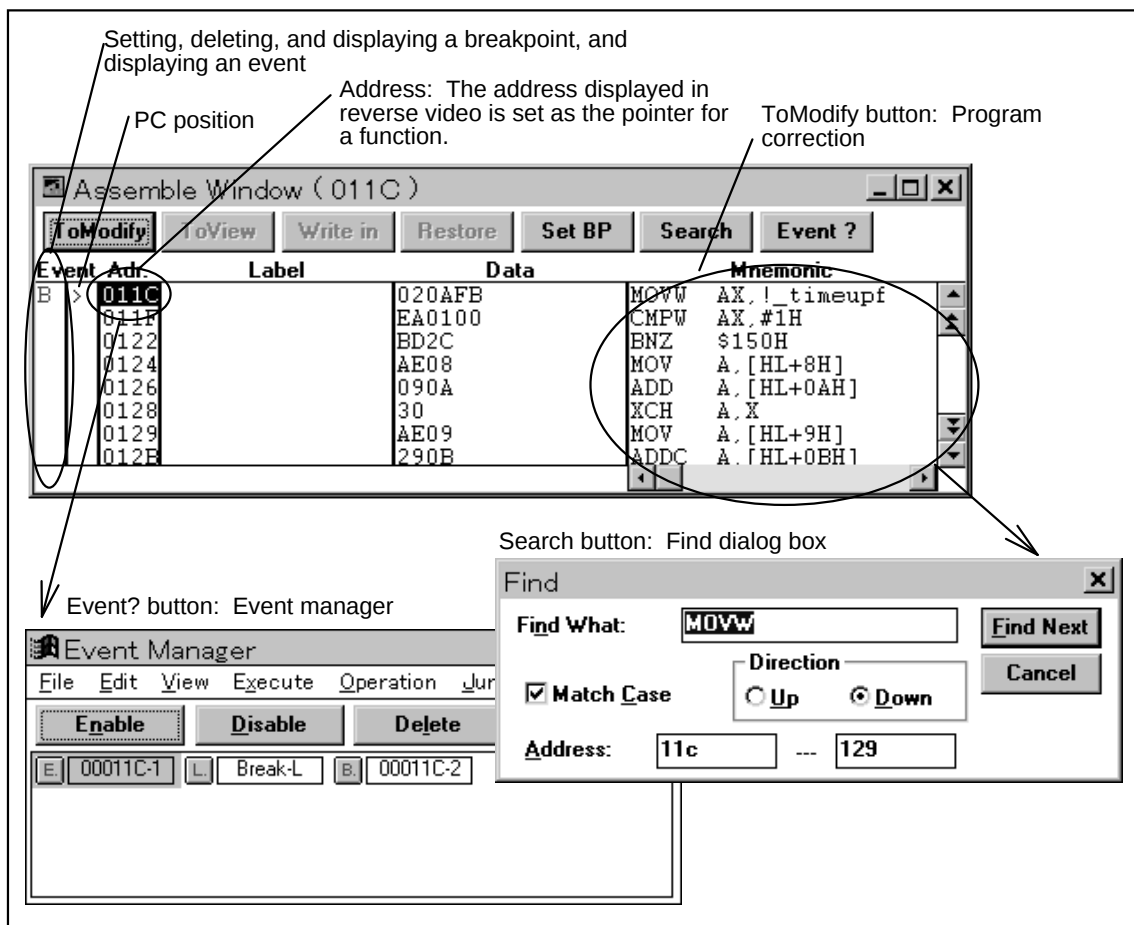


Displaying the contents saved to a file: View file load dialog box

2.3.3 Functions Supported by the Assemble Window

- The Assemble window supports many functions such as the setting of breakpoints and PC setting.
- The supported functions are listed below:

Function	Procedure	
	Using the mouse	Using the keyboard
Setting/deleting a breakpoint	Click the point mark area.	1. Select an address (with the mouse). 2. Select Execute -> Set BP from the menu bar. (CTRL+B)
Setting PC	---	1. Select an address (with the mouse). 2. Select Execute -> Set PC from the menu bar. (CTRL+E)
Retrieving a character string	1. Select a character string. 2. Click the Search button.	1. Select a character string (with the mouse). 2. Select View -> Search... from the menu bar.
Checking an event	1. Select the address at which an event is set. 2. Click the Event ? button.	1. Select the address at which an event is set (with the mouse). 2. Select View -> Event? from the menu bar.



2.3.4 Jump from the Assemble Window

- Jump from a line in the Assemble window to the corresponding source line or memory address is supported.
- Select an address as the jump destination. Then, the selected address is set as the jump pointer.
- When the jump destination is the Source window, a jump is made to a source line including the jump pointer.

Jump destination	Procedure
Source window	1. Select an address. 2. Select <u>J</u>ump -> <u>S</u>ource Text... from the menu bar.
Memory window	1. Select an address. 2. Select <u>J</u>ump -> <u>M</u>emory... from the menu bar.

The jump destination is the address displayed in reverse video: In the following example, address 132H is the jump pointer.

The screenshot shows three windows: Assemble Window (0124), Source Window (sample.c), and Memory Window. In the Assemble window, address 0132 is highlighted. In the Source window, address 0038 is highlighted. In the Memory window, address 0130 is highlighted. Arrows indicate the jump path from 0132 in Assemble to 0038 in Source, and from 0130 in Memory to 0038 in Source.

Assemble Window (0124)

Event	Adr.	Label	Data	Mnemonic
	0130		BE06	MOV [HL+6H], A
	0132		1400FB	MOVW DE, #0FB00H
	0135		B5	PUSH DE
	0136		1480FE	MOVW DE, #0FE80H
	0139		B5	PUSH DE
	013A		9A0002	CALL !_clockup
	013D		B0	POP, &V

Source Window (sample.c)

```

Font: FixedSys Size: 15
0038 clockup( &timedsp, &timecnt );
0039 timeupf = 0;
0040 i++;
0041 }
0042 if( disrupf == 1 ){

```

Memory Window

Address	Value
0100	7A 1E AE 02
0110	01 82 BE 01
0120	01 00 BB 2C
0130	BE 06 14 00 FB B5 14 80 FE B5 9A 00 02 B0 B0 10
0140	00 00 03 0A FB AE 0A 30 AE 0B 80 BE 0B 30 BE 0A
0150	02 0C FB EA 01 00 BD 21 AE 0A 61 01 30 AE 0B 27
0160	BE 05 30 BE 04 9A 03 04 10 00 00 03 0C FB AE 08

Jump to the Memory window

Jump to the Source window
A jump is made to the source line including address 132H in the Source window.

2.4 Manipulating Memory

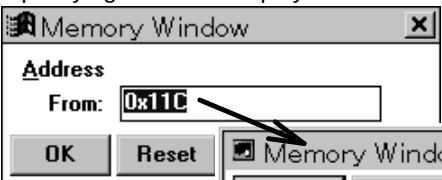
- The user can display, modify, and search for memory data.
- The user can display and modify memory data in ASCII format.

2.4.1 Displaying and Modifying Memory Data

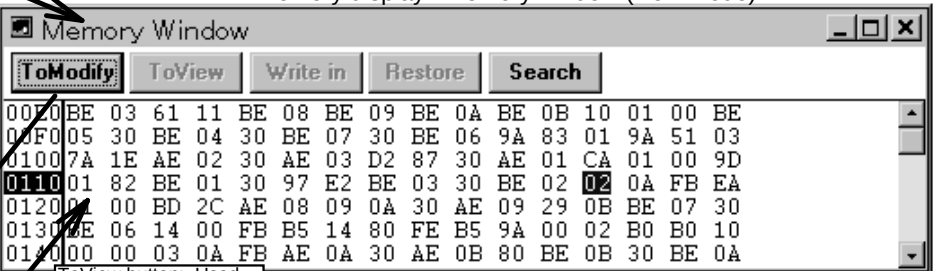
- The user can display and modify memory data in the Memory window.

	Memory	Procedure
Display	When a display address is to be selected	The Addressing dialog box can be opened by means of either of the following two procedures: 1. Select Browse -> Memory... from the menu bar. 2. Select the  button.
	When memory data is to be displayed starting from an address selected in another window (such as the Source window, Assemble window, or Register window)	1. Select an address to act as a display pointer. 2. Select Jump -> Memory... from the menu bar, or press CTRL + M.
Modification		1. Open the Memory window. 2. Switch to modify mode by clicking the  button. 3. Position the cursor to the desired memory display/modification area, then modify the data. 4. After entering the new data, execute the modification by clicking the  button. 5. Switch to view mode by clicking the  button.

Specifying a desired display start address: Addressing dialog box



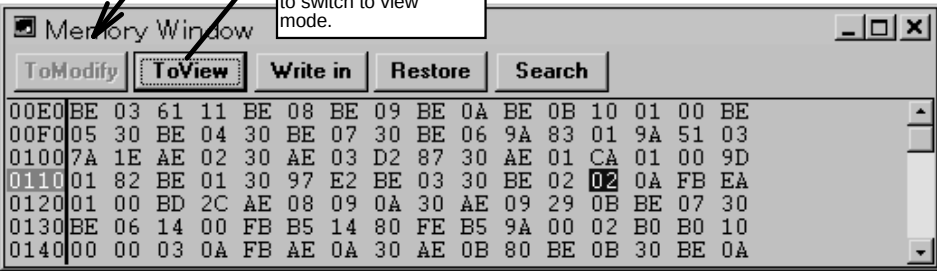
Memory display: Memory window (view mode)



ToModify button: Used to switch to modify mode.

ToView button: Used to switch to view mode.

Memory modification: Memory window (modify mode)



2.4.2 Basic Memory Data Operations

- Basic memory data operations are enabled by activating the Memory window.
- The basic operations include initialization, copy, and comparison.

To initialize memory:

Select **Edit** -> **Memory** -> **Memory Fill...** from the menu bar.

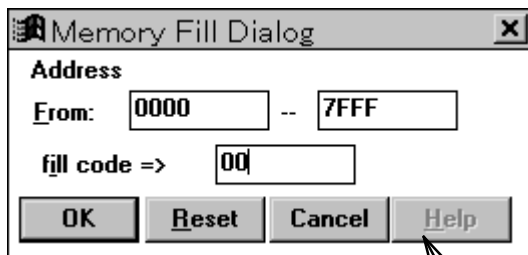
To copy memory data:

Select **Edit** -> **Memory** -> **Memory Copy...** from the menu bar.

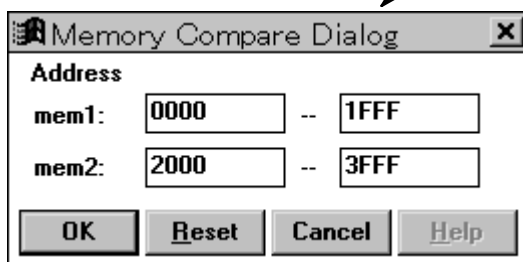
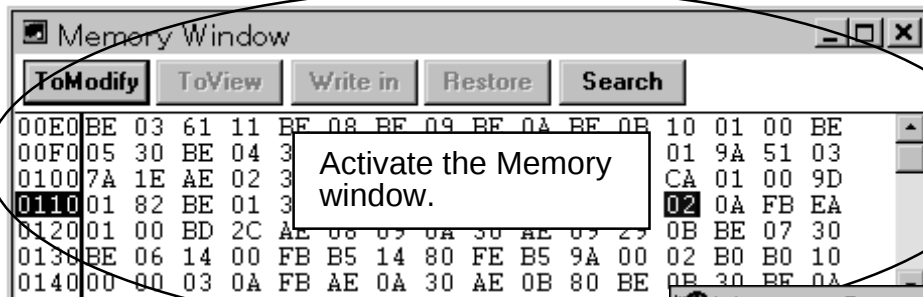
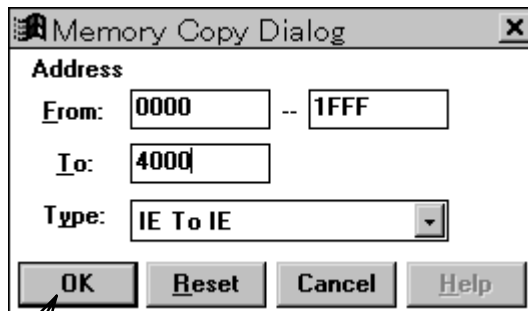
To compare memory data:

Select **Edit** -> **Memory** -> **Memory Compare...** from the menu bar.

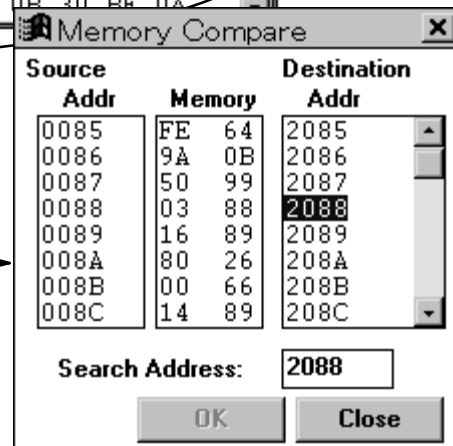
Memory initialization: Memory Fill dialog box
String data of no more than 16 bytes can be specified.



Memory copy: Memory Copy dialog box
A memory copy destination can be specified.



Memory comparison: Memory Compare dialog box



If memory data comparison reveals a mismatch, the Memory Compare result dialog box appears.

2.4.3 Saving and Referencing Displayed Memory Data

- Displayed memory data can be saved to a file. A file containing saved memory data can be referenced.
- Memory data is saved in text format, allowing an editor to be used to reference saved memory data.

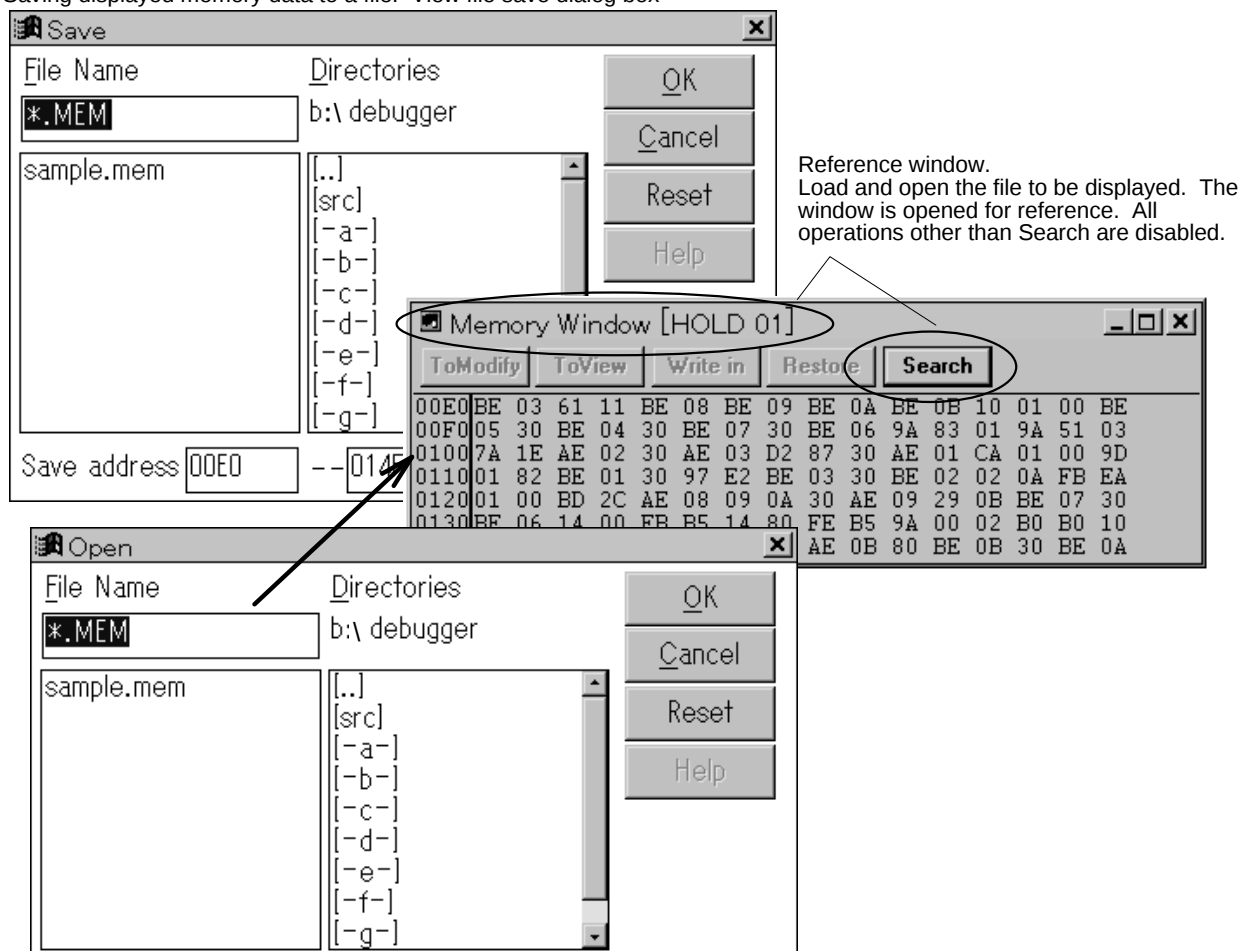
To save displayed memory data to a file:

1. Activate the Memory window.
2. Select File -> Save As... from the menu bar.
3. Save the displayed memory data by using the View file save dialog box.

To open and reference a file containing saved memory data:

1. Activate the Memory window.
2. Select File -> Open... from the menu bar.
3. Load the file to be referenced by using the View file load dialog box.

Saving displayed memory data to a file: View file save dialog box



Displaying memory data that has been saved to a file: View file load dialog box

2.4.4 Functions Available in the Memory Window

- The Memory window allows the user to perform a range of functions including modification in ASCII format, and data search.
- The available functions are listed below.

Function	Procedure
Character string search	1. Select a character string. 2. Click the Search button, or select V iew -> S earch... from the menu bar.
ASCII character display selection	Select V iew -> M emory -> A scii from the menu bar.
Type display selection	Select V iew -> M emory -> N ibble, B yte, W ord, or L ong from the menu bar.
Number system display selection	Select V iew -> B in, O ct, D ec, or H ex from the menu bar.

Data display.
ToModify button: Used to modify data.
The user can select binary, octal, decimal, or hexadecimal display.
The user can also select nibble, byte, word, and long for display.

Display in ASCII

Address display: An address displayed in reverse video acts as a pointer to be used with each function.

Search button: Find dialog box

Find

Find What: 00

Find Next

Cancel

Direction

☐ Up ☒ Down

Address: 60 --- cf

2.4.5 Jumping from the Memory Window

- This function enables a jump to the source line or disassembly start address corresponding to an address in the Memory window.
- A jump destination can be specified by selecting a desired address. The selected address serves as a jump pointer.
- When a jump is made to the Source window, a jump to the source line including the jump pointer occurs.

Jump destination	Procedure
Source window	1. Select an address. 2. Select <u>J</u> ump -> <u>S</u> ourceText... from the menu bar.
Assemble window	1. Select an address. 2. Select <u>J</u> ump -> <u>A</u> ssemble... from the menu bar.

A jump is made to the address displayed in reverse video.
In this case, address D0H is the jump pointer.

The screenshot illustrates the process of jumping between windows in a debugger. It shows three windows: Memory Window, Source Window (sample.c), and Assemble Window (00D0).

Memory Window: Displays a list of memory addresses and their corresponding data. Address 00D0B is highlighted in reverse video.

Source Window (sample.c): Displays the source code. Address 0024B is highlighted in reverse video.

Assemble Window (00D0): Displays the assembly code. Address 00D0 is highlighted in reverse video.

Arrows indicate the sequence of jumps: from Memory Window to Source Window, and from Source Window to Assemble Window.

Jump to the Source window. A jump to the source line including address D0H in the Source window occurs.

Jump to the Assemble window

2.5 Manipulating Registers

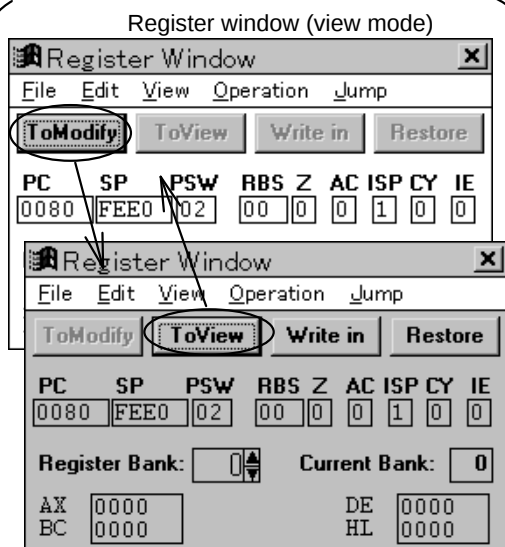
- Registers are classified into three major types: control registers, general-purpose registers, and special function registers (SFRs).
- The control registers and general-purpose registers can be displayed and modified in the Register window. The SFRs can be displayed and modified in the SFR window.

2.5.1 Displaying and Modifying Registers

- The user can display and modify control registers, general-purpose registers, and SFRs.
- The user can manipulate control registers and general-purpose registers in the Register window, and manipulate SFRs in the SFR window.

Memory		Procedure
Display	Control registers and general-purpose registers (Register window)	Select Browse -> Register... from the menu bar, or click the  button.
	SFRs (SFR window)	Select Browse -> Sfr... from the menu bar, or click the  button.
Modification (common to the Register window and SFR window)		1. Open a desired window. 2. Switch to modify mode by clicking the  button. 3. Position the cursor to a desired register, then modify the data. 4. After making the modification, execute the modification by clicking the  button. 5. Switch to view mode by clicking the  button.

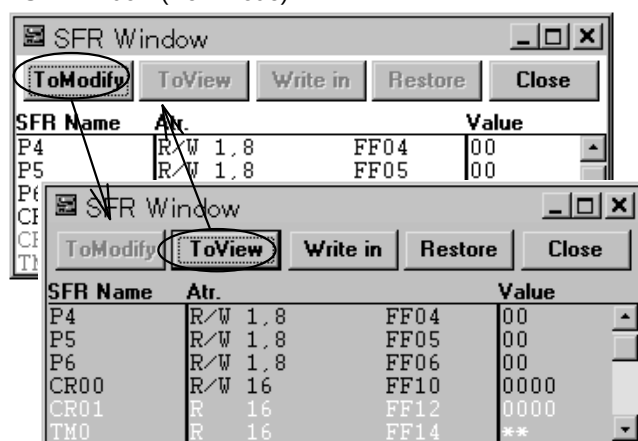
Displaying and modifying control registers and general-purpose registers



Register window (modify mode)

Displaying and modifying SFRs

SFR window (view mode)



SFR window (modify mode)

ToModify button: Used to switch to modify mode.
ToView: Used to switch to view mode.

2.5.2 Saving and Referencing Displayed Register Data

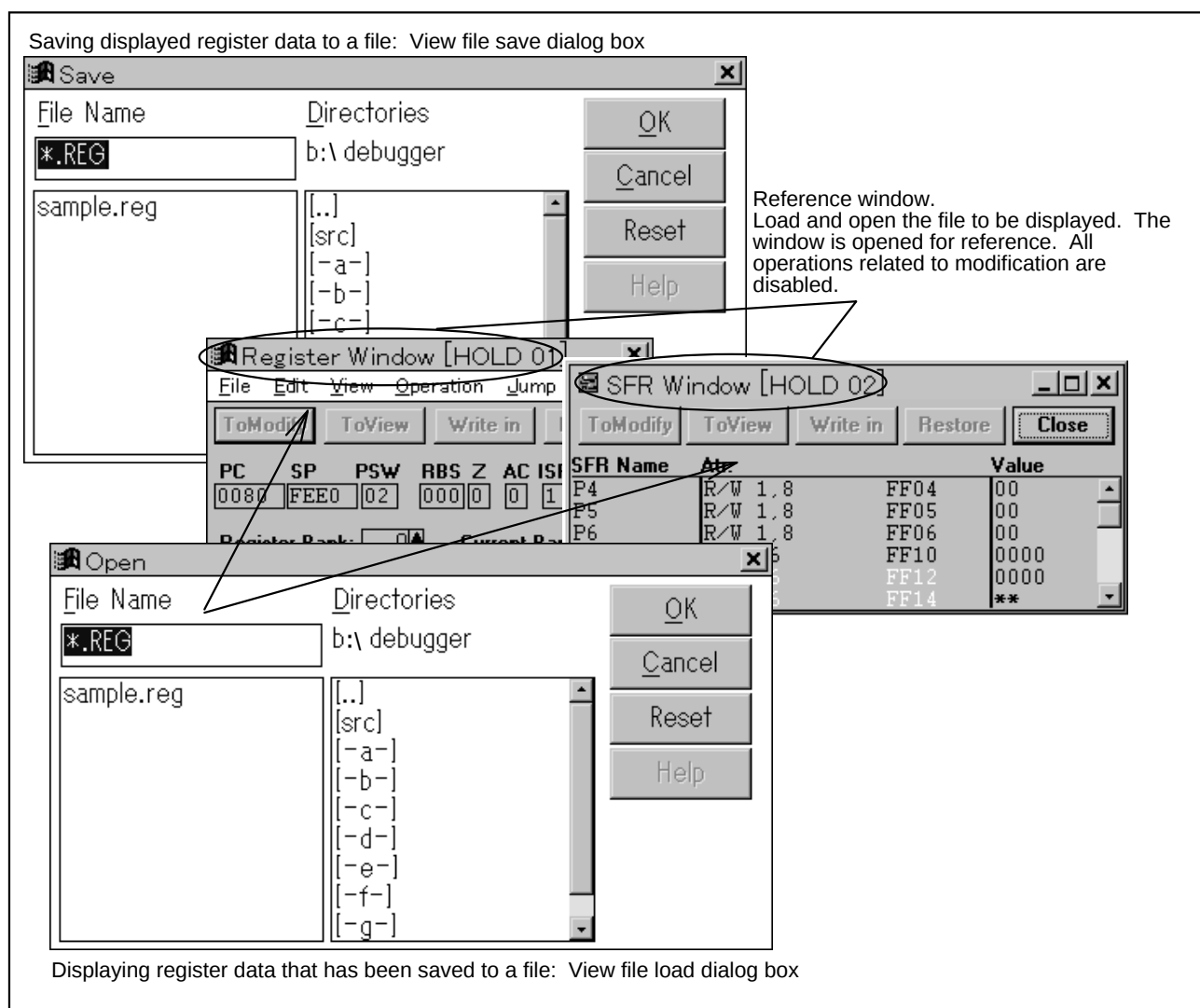
- Displayed register data can be saved to a file. A file containing saved register data can be referenced.
- Register data is saved in text format, allowing an editor to be used to reference saved register data.

To save displayed register data to a file:

1. Activate a desired window.
2. When the Register window has been selected, select **File -> Open/save Condition -> Save File as...** from the menu bar.
When the SFR window has been selected, select **File -> Save As...** from the menu bar.
3. Save the displayed register data by using the View file save dialog box.

To open and reference a file containing saved register data:

1. Activate a desired window.
2. When the Register window has been selected, select **File -> Open/save Condition -> Open Condition...** from the menu bar.
When the SFR window has been selected, select **File -> Open...** from the menu bar.
3. Load the file to be referenced by using the View file load dialog box.



2.5.3 Functions Available in the Register Window

- The Register window allows the user to choose between the function name display option and absolute name display option, choose between the register display option and pair register display option, and so forth.
- The available functions are listed below.

Function		Procedure
Display switching	Absolute name display/function name display	Absolute name: Select <u>V</u> iew -> <u>A</u> bsolute Name from the menu bar. Function name: Select <u>V</u> iew -> <u>F</u> unction Name from the menu bar.
	Register display/pair register display	Register display: Select <u>V</u> iew -> <u>R</u> egister from the menu bar. Pair register display: Select <u>V</u> iew -> Register <u>P</u> air from the menu bar.
Number system display selection		Select <u>V</u> iew -> <u>B</u> in, <u>O</u> ct, <u>D</u> ec, or <u>H</u> ex from the menu bar.

2.5.4 Functions Available in the SFR Window

- The SFR window allows the user to select the display order, specify whether attribute data is to be displayed, and so forth.
- The available functions are listed below.

Function	Procedure
Display order selection	The user can choose either address order or alphabetic order as the display order: Select <u>V</u> iew -> <u>S</u> fr -> <u>A</u> ddress Sort from the menu bar.
Attribute display selection	Select <u>V</u> iew -> <u>S</u> fr -> <u>A</u> tttribute -> <u>S</u> how or <u>H</u> ide from the menu bar.
Pickup display selection	Only those SFRs that have been modified but not yet written to a target in modify mode are displayed. Select <u>V</u> iew -> <u>S</u> fr -> <u>P</u> ick Up from the menu bar.

Address order, attribute display (default)

SFR Name	Atr.	Value
P0	R/W 1,8	FF00 00
P1	R/W 1,8	FF01 00
P2	R/W 1,8	FF02 00
P3	R/W 1,8	FF03 00
P4	R/W 1,8	FF04 00
P5	R/W 1,8	FF05 00

Pickup display

SFR Name	Atr.	Value
CR10	R/W 8	FF16 20
FFD0	R/W 1,8	FFD0 11
MK0H	R/W 1,8	FFE5 1F
PM1	R/W 1,8	FF21 00

Hiding attribute data

SFR Name	Value
P0	00
P1	00
P2	00
P3	00
P4	00
P5	00

Alphabetical order

SFR Name	Atr.	Value
ADCR	R 8	FF1F FF
ADIS	R/W 8	FF84 00
ADM	R/W 1,8	FF80 01
ADTC	R/W 1,8	FF69 00
ADTP	R/W 8	FF6A 00
CR00	R/W 16	FF10 0000

2.5.5 Jumping from the Register Window

- This function enables a jump to the source line, disassembly start address, or memory address corresponding to a register value in the Register window.
- A jump destination can be specified by selecting a desired register. The value of a selected register acts as a jump pointer.
- When a jump is made to the Source window, a jump to that source line including the jump pointer is performed.

Jump destination	Procedure
Source window	1. Select a register 2. Select <u>J</u> ump -> <u>S</u> ourceText... from the menu bar.
Assemble window	1. Select a register. 2. Select <u>J</u> ump -> <u>A</u> ssemble... from the menu bar.
Memory window	1. Select a register. 2. Select <u>J</u> ump -> <u>M</u> emory... from the menu bar.

A jump is made to the value of the selected register. In this case, address D0H, held in the BC register, is the jump pointer.

Jump to the Source window. A jump to that source line including address D0H in the Source window is performed.

Jump to the Assemble window

2.6 Creating Events

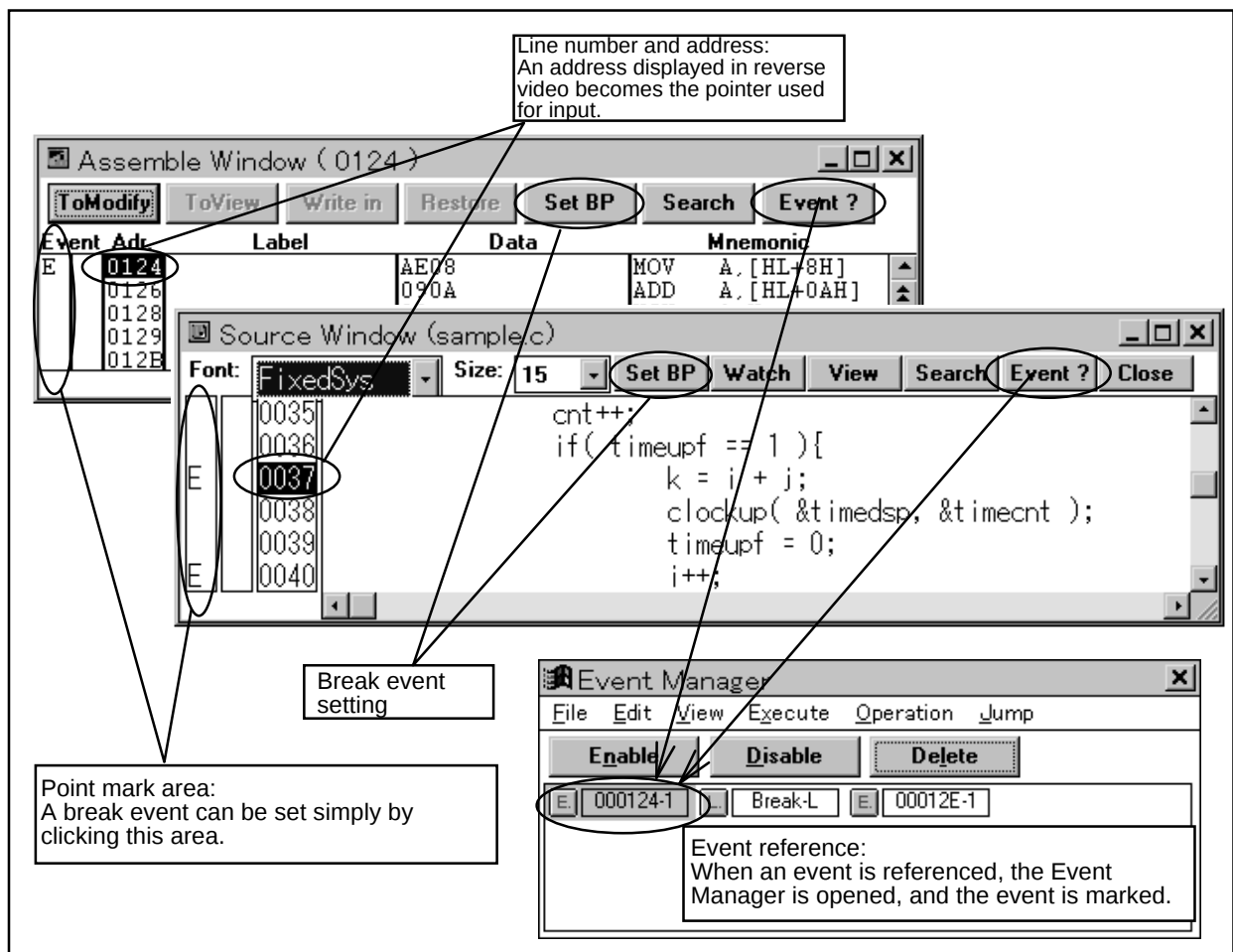
- An event, set beforehand in a program, specifies that an operation is to be performed when a specified condition is satisfied.
- Two types of conditions are used. One is an execution event, which is set for a program execution address. The other is an access event, which is set for memory data accessed by a programmed instruction.
- Four types of events are used to perform operations. These include break events for terminating the program or analyzer, and qualified events, section events, and snapshot events which are used to control the tracer.
- The event-related windows are listed below.

Operation		Window
Event management		Event Manager
Event condition creation	Event condition	Event Set dialog box
	Event link condition	Event Link dialog box
Event setting	Break condition	Break dialog box
	Trace condition	Trace dialog box
	Snapshot condition	Snap-Shot dialog box
	External sense clip condition	External Sense Clip dialog box

2.6.1 Setting and Referencing Events in the Source Window and Assemble Window

- In the Source window and Assemble window, break events can be set, and events can be referenced.
- If a break event is set in the Source window or Assemble window, a parallel-linked event link condition, named Break-L, is automatically created.
- All set break events become execution events (with the status set to Run).

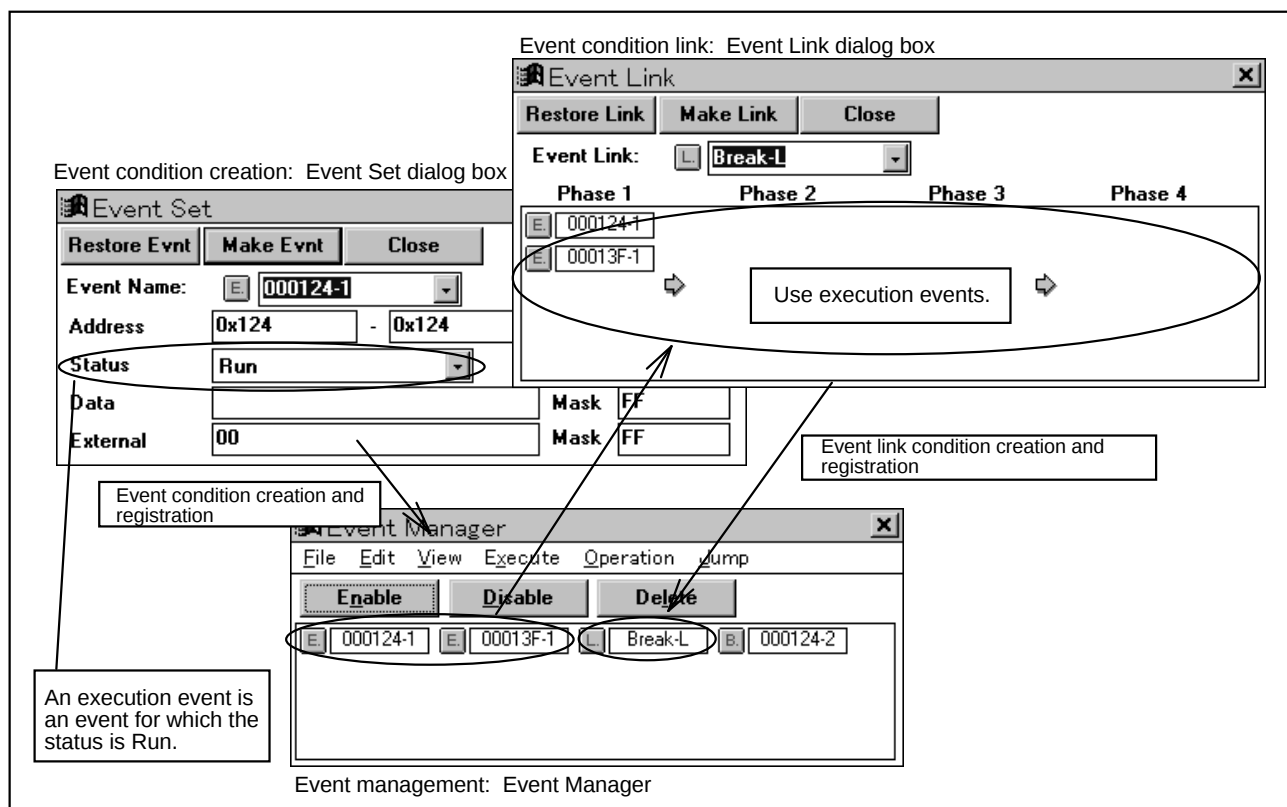
Function	Procedure
Break event setting	Use any of the five methods described below. 1. Click the point mark area. 2. Double-click a line number or address. 3. Select a line number or address, then click the  button. 4. Select a line number or address, then select <u>E</u>xecute -> Set <u>B</u>P from the menu bar. 5. Select a line number or address, then press +.
Event condition reference	Use either of the two methods described below. Select an address or line number indicated by E in the point mark area, then perform either of the following operations: 1. Click the  button. 2. Select View -> Event? from the menu bar.



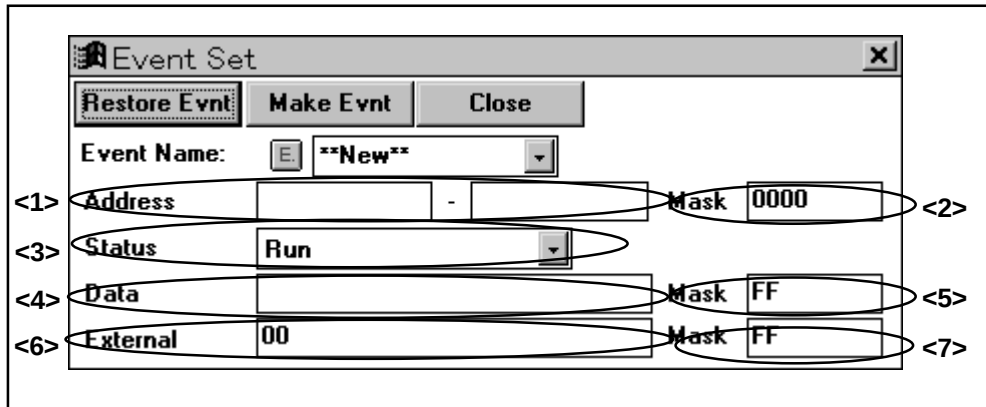
2.6.2 Creating Event Conditions

- Event conditions are divided into two main types: execution events for detecting an execution address, and access events for detecting access data.
- When an execution event is used, it can be combined with an event condition.

Function	Procedure
Event condition creation	The Event Set dialog box is used. Select Browse -> Event -> EventSet... from the menu bar.
Event link condition creation	1. Create an execution event in the Event Set dialog box. 2. Open the Event Manager by selecting Browse -> Event -> EventManager... from the menu bar. 3. Open the Event Link dialog box by selecting Browse -> Event -> EventLinkSet... from the menu bar. 4. Create an event link condition by dragging & dropping the execution event created in 1., above.



- Examples of event condition setting are given below.
When the Event Set dialog box is opened, the default screen, shown below, initially appears. Modify the screen settings as required.

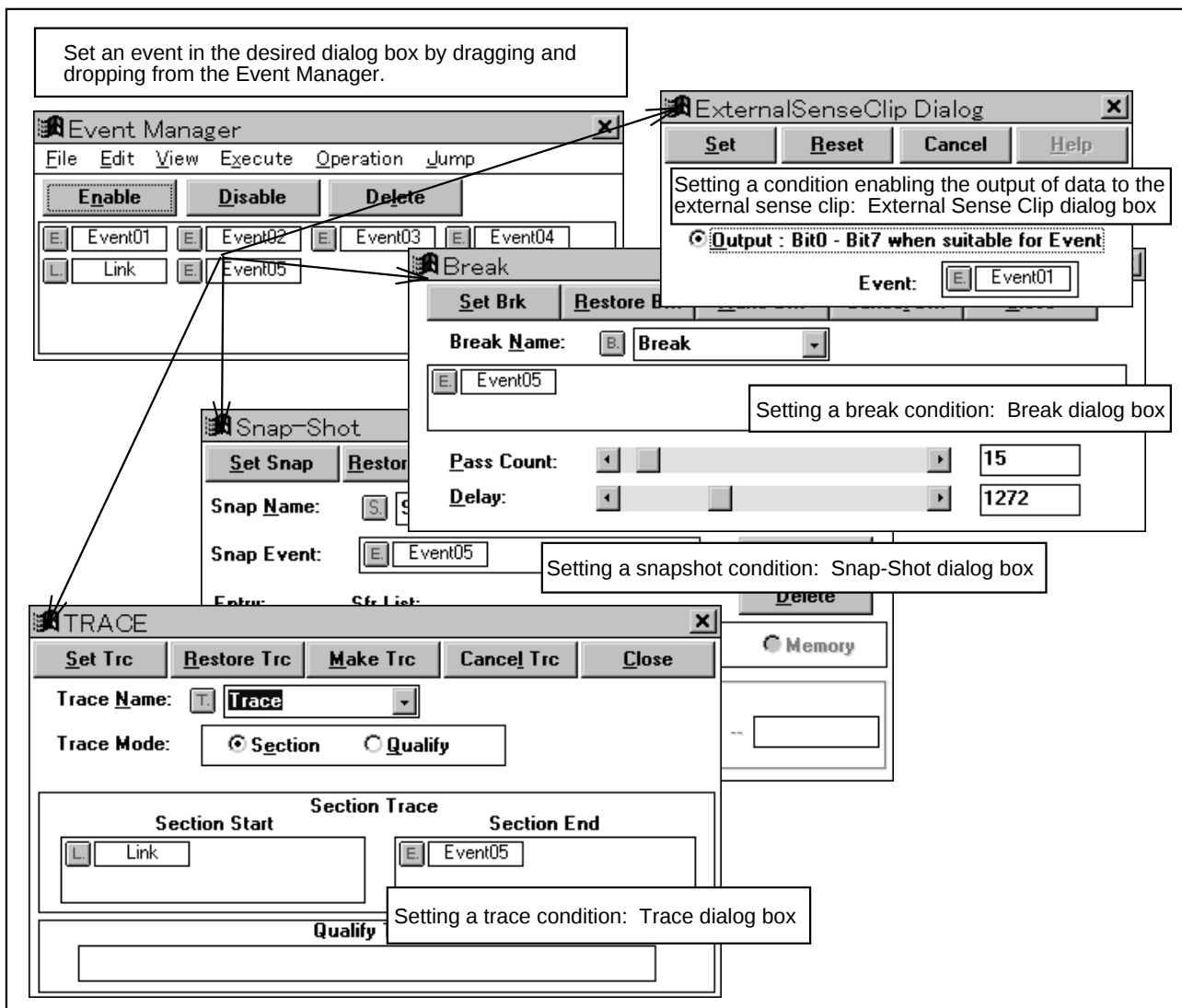


Condition	Setting			Remarks
When a program at address 0x100 is executed	<1>0x100	<2>0	<3>Run	The defaults are used for <4>, <5>, <6>, and <7>.
When memory access to address 0xfe00 is performed	<1>0xfe00 <4>0x00	<2>0 <5>ff	<3>Data R/W	The defaults are used for <6> and <7>.
When memory access is performed for addresses 0xfe00 to 0xfe7f	<1>0xfe00-0xfe7f <3>Data R/W <5>ff	<2>0	<4>0	The defaults are used for <6> and <7>. An event occurs when any address in the range is accessed.
When memory is read (with no address condition set)	<1>0 <4>0	<2>ffff <5>ff	<3>Data Read	The defaults are used for <6> and <7>.
If bit 0 is 1 when writing to address 0xfb01 is performed	<1>0xfb01 <4>1	<2>0 <5>fe	<3>Data Write	The defaults are used for <6> and <7>. For mask specification, set those bits to be monitored to 0, and set the other bits to 1.
When 0x10 is written to address 0xfb01	<1>0xfb01 <4>0x10	<2>0 <5>0	<3>Data Write	The defaults are used for <6> and <7>.
When an event is to be set at the start of function sub(), coded in C	<1>_sub	<2>0	<3>Run	The defaults are used for <4>, <5>, <6>, and <7>.
When the value of variable cnt, registered in C, becomes 0x46	<1>_cnt <4>0x46	<2>0 <5>0	<3>Data R/W	The defaults are used for <6> and <7>.
When an event is to be set with the START function of the assembler	<1>START	<2>0	<3>Run	The defaults are used for <4>, <5>, <6>, and <7>.
When the value of assembler variable DATA becomes 35H	<1>DATA <4>35H	<2>0 <5>0	<3>Data R/W	The defaults are used for <6> and <7>.

2.6.3 Setting Events

Event conditions registered in the Event Set dialog box or Event Link dialog box can be used as break conditions and trace conditions.

Condition	Procedure
When used as a break condition	Select Browse -> BreakSet... from the menu bar.
When used as a trace condition	Select Browse -> Trace -> TraceSet... from the menu bar.
When used as a snapshot condition	Select Browse -> Trace -> SnapShotTraceSet... from the menu bar.
When used as an external sense clip condition	Select Execute -> ExtSenseClip... from the menu bar.



2.6.4 Saving and Restoring Event Conditions

- Event conditions can be saved to a file. Saved event conditions can be referenced.
- Event conditions are saved in text format, allowing an editor to be used to reference saved event conditions.

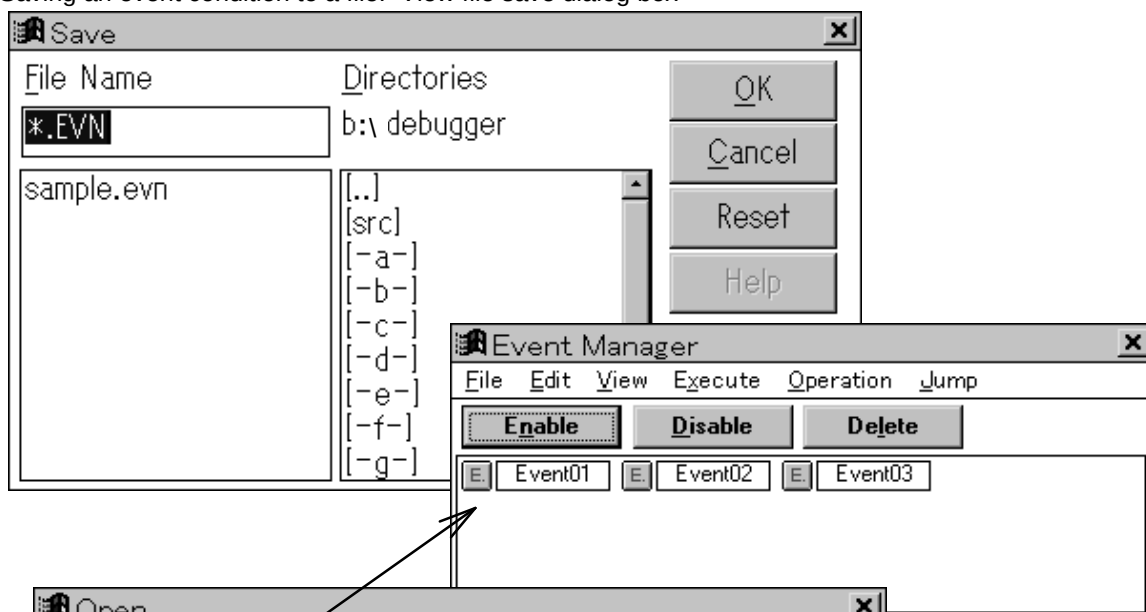
To save an event condition to a file:

1. Activate the Event Manager.
2. Select **File** -> **Open/save Condition** -> **Save File as...** from the menu bar of the Event Manager.
3. Save the event condition by using the View file save dialog box.

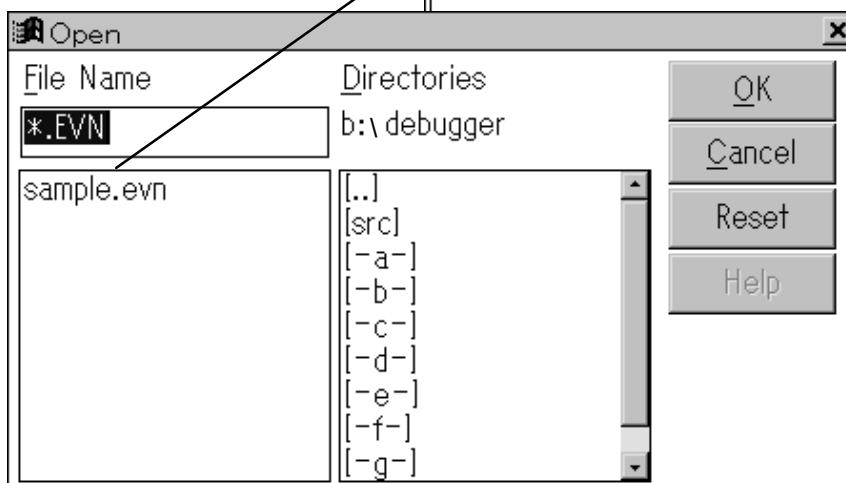
To restore a saved event condition:

1. Activate the Event Manager.
2. Select **File** -> **Open/save Condition** -> **Open Condition...** from the menu bar of the Event Manager.
3. Load the file containing the event condition to be restored with the View file load dialog box.

Saving an event condition to a file: View file save dialog box



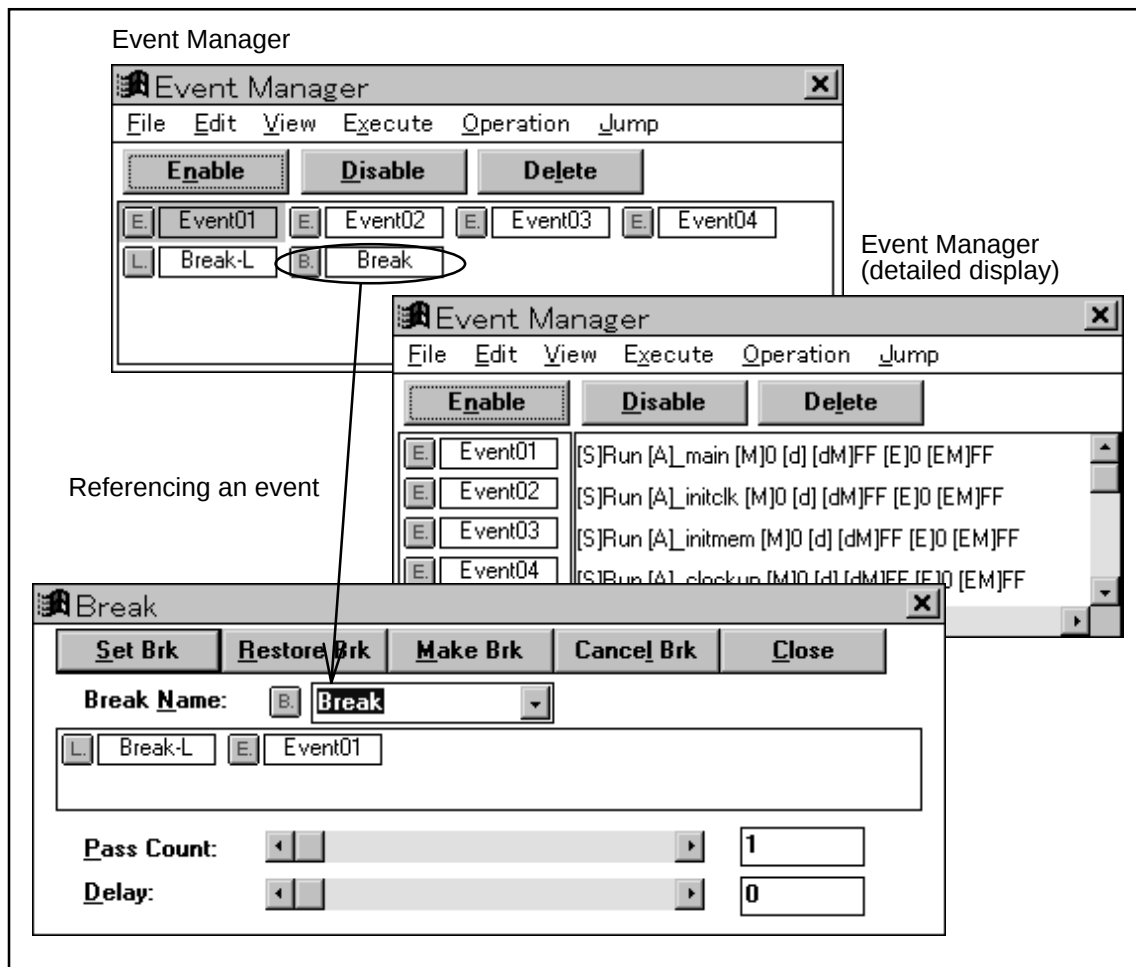
Restoring an event condition saved to a file: View file load dialog box



2.6.5 Functions Available in the Event Manager

- The Event Manager allows the user to use a variety of functions such as enabling/disabling an event, deleting an event, and referencing an event.
- The available functions are listed below.

Function	Procedure
Enabling/disabling an event	Select an event to be enabled or disabled, then perform the following: To enable the event: Click the  button. To disable the event: Click the  button.
Deleting an event	1. Select the event to be deleted. 2. Click the  button.
Detailed event condition display	Select <u>V</u> iew -> <u>D</u> etail from the menu bar of the Event Manager.
Changing the order of display	The order of display can be changed using the menu bar of the Event Manager. To enable display in event name order: Select <u>V</u> iew -> <u>N</u> ame. To enable display in type order: Select <u>V</u> iew -> <u>K</u> ind.
Referencing/modifying an event condition	1. Select the event to be referenced or modified. 2. Select an option from <u>O</u> peration in the menu bar of the Event Manager.



2.6.6 Jumping to an Event Setting Address

- This function enables a jump to the source line, disassembly start address, or memory address corresponding to the address of an event condition in the Event Manager.
- A jump destination can be specified by selecting an event condition. The start address of a selected event condition acts as a jump pointer.
- When a jump is made to the Source window, a jump to that source line including the jump pointer is performed.

Jump destination	Procedure
Source window	1. Select an event condition. 2. Select <u>J</u> ump -> <u>S</u> ourceText... from the menu bar.
Assemble window	1. Select an event condition. 2. Select <u>J</u> ump -> <u>A</u> ssemble... from the menu bar.
Memory window	1. Select an event condition. 2. Select <u>J</u> ump -> <u>M</u> emory... from the menu bar.

A jump is made to the address of the selected event, Event02.
In this case, the start address (D0H) of function main is the jump pointer.

The screenshot illustrates the process of jumping to a source line based on an event condition. The **Event Manager** window shows a list of events, with **Event02** selected. The **Event Set** window shows the details for **Event02**, including its name, address (**_main**), status (**Run**), and data. The **Memory Window** displays a list of memory addresses and their corresponding data. The **Source Window (sample.c)** shows the source code, with the address **0024** highlighted. The **Assemble Window (00D0)** shows the assembly code, with the address **00D0** highlighted. Arrows indicate the jump process from **Event02** in the Event Manager to its address (**00D0**) in the Assemble Window, and then to the corresponding source line (**0024**) in the Source Window.

Jump to the Source window. A jump to that source line including address D0H in the Source window is performed.

Jump to the Assemble window

2.7 Manipulating Symbols (Variables)

- The user can display and modify the values of variables.
- Before an operation such as symbol debugging can be performed, a load module file including debug information must be loaded.
- The user can enter symbols in the address and data input fields of each window.
- To enter symbols, observe the input formats indicated below.

Type of symbol	Input format
Variable defined in C	_fnc file#_fnc
Variable defined in assembler language	fnc file#fnc
Source line number	file:no
SFR	sfrname

fnc: Function name or variable name sfrname: SFR name file: File name
no: Line number

1. When specifying a variable defined in C, prefix the variable with an underbar (_).
2. Use a sharp (#) as the separator between a file name and variable name.
3. Use a colon (:) as the separator between a file name and line number.

- The windows related to symbol operations are listed below.

Operation	Window
Display of variables	Variable window
Registration of displayed variables	Add Variable dialog box
Temporary display of variables	Variable View dialog box
Display of local variables	Local Variable window

2.7.1 Displaying and Modifying Variables

- The user can display and modify the values of variables in the Variable window, Variable View dialog box, and Local Variable window.

Variable		Procedure
Display	Display at all times	Display the variables in the Variable window. Select <u>V</u> iew -> <u>W</u> atch Variable... from the menu bar.
	Temporary display	1. Select a source variable displayed in the Source window. 2. Select <u>V</u> iew -> <u>V</u> iew Variable... from the menu bar, or click the  button in the Source window.
	Display of local variables	Select <u>B</u> rowse -> <u>L</u> ocal Variable... from the menu bar.
Modifi- cation	Variable modification	Use the Variable window to modify a variable. Use the Local Variable window to modify a local variable. Both windows are modified as follows: 1. Open the desired window.
	Local variable modification	2. Switch to modify mode by clicking the  button. 3. Position the cursor to the variable to be modified, then modify the data. 4. After entering the new data, execute the modification by clicking the  button. Switch to view mode by clicking the  button.
Registration		The Variable window allows the user to register a displayed variable. To register a variable in the Source window: 1. Select a source variable displayed in the Source window. 2. Select <u>V</u> iew -> <u>W</u> atch Variable... from the menu bar, or click the  button in the Source window. To register a variable in the Add Variable dialog box for variable registration: 1. Select <u>V</u> iew -> <u>A</u> dd Variable... from the menu bar. 2. Register the variable in the Add Variable dialog box.
Deletion		The user can delete any variable displayed and registered in the Variable window. 1. Select the variable to be deleted. 2. Select <u>O</u> peration -> <u>D</u> elete from the menu bar.

After a variable has been selected in the Source window, that variable can be displayed or modified by clicking the Watch button, or can be temporarily displayed by clicking the View button.

Source Window (sample.c)

```

0035 cnt++;
0036 if( timeupf == 1 ){
0037     k = i + j;
0038     clockup( &timedsp, &timecnt
0039     timeupf = 0;
  
```

Variable View

Variable Name: timeupf

0

Temporary display of variables: Variable View dialog box

Variable Window

File Edit View Operation

ToModify ToView Write in Restore Delete

(File:Function:variable)	(Variable)	Value
+timedsp =		{...}
long cnt =		2
int timeupf =		0

Display and modification of variables: Variable window

Local Variable

ToModify ToView Write in Restore Close

```

int i = 0
int j = 0
int k = 1
int l = 1
long cnt = 2
  
```

Display and modification of local variables: Local Variable window

Add Variable Dialog

Name: timeupf

Type: ☒ C Language ☐ Other

Other:

Size: ☒ Byte ☐ Word ☐ Double Word

Number: 1

OK

To register a variable by keyboard input, or to display a variable registered by the assembler in the Variable window: Add Variable dialog box

2.7.2 Saving and Referencing Symbol Data

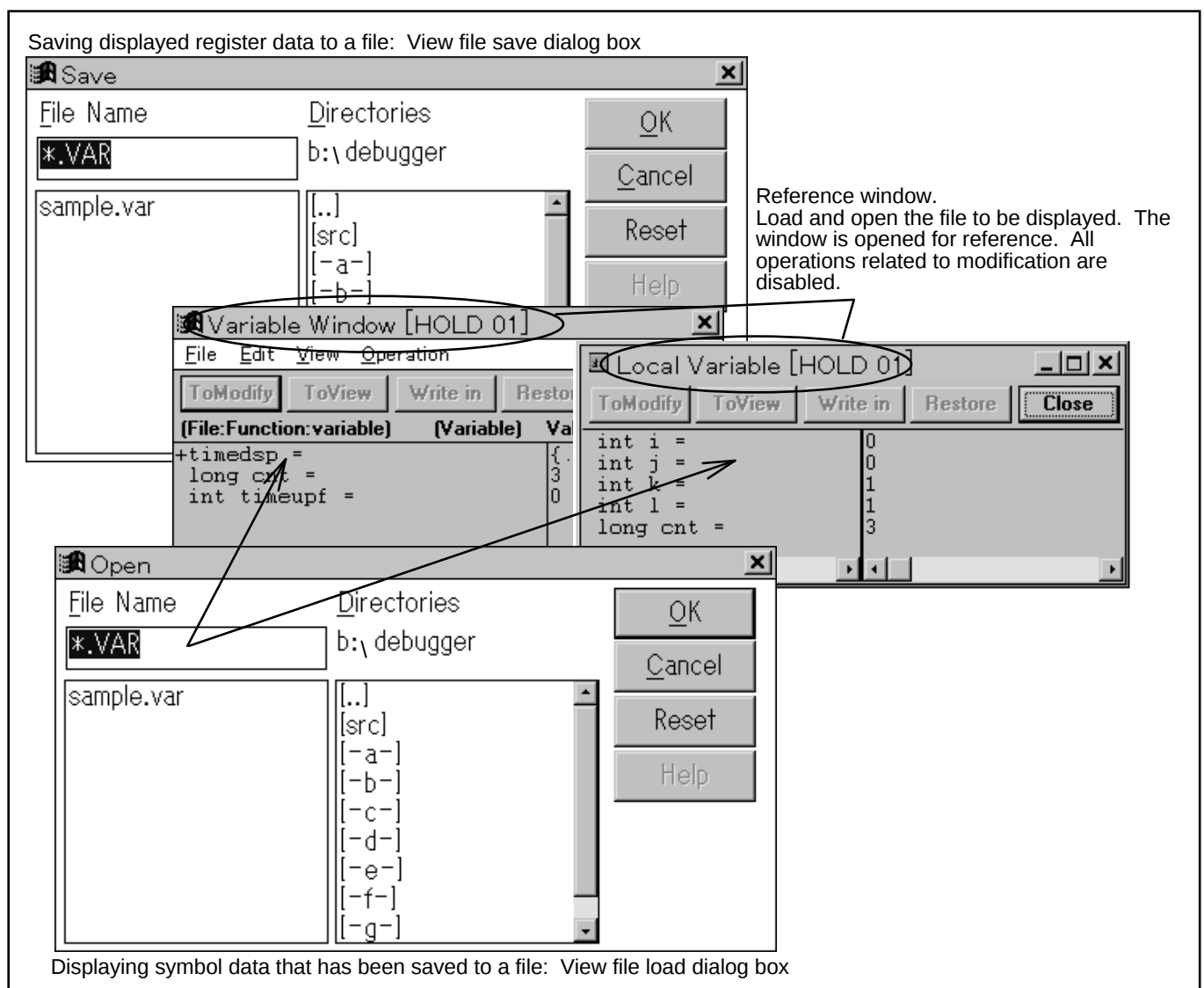
- Displayed symbol data can be saved to a file. A file containing saved symbol data can be referenced.
- Symbol data is saved in text format, allowing an editor to be used to reference saved symbol data.

To save displayed symbol data to a file:

1. Activate the desired window.
2. When the Variable window has been selected, select **File -> Open/save Condition -> Save File as...** from the menu bar.
When the Local Variable window has been selected, select **File -> Save As...** from the menu bar.
3. Save displayed symbol data by using the View file save dialog box.

To open and reference a file containing saved symbol data:

1. Activate the desired window.
2. When the Variable window has been selected, select **File -> Open/save Condition -> Open Condition...** from the menu bar.
When the Local Variable window has been selected, select **File -> Open...** from the menu bar.
3. Load the file to be referenced by using the View file load dialog box.

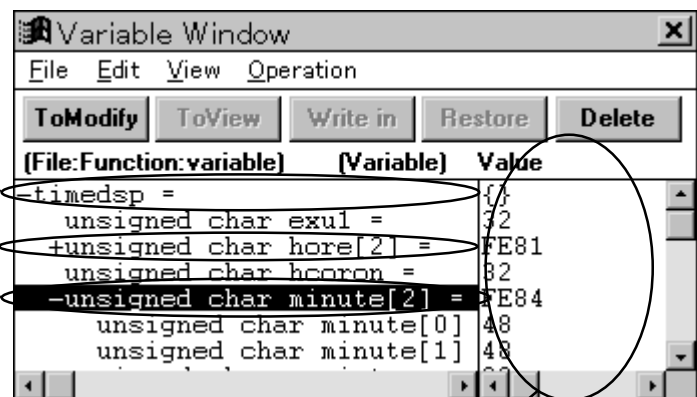


2.7.3 Functions Available in the Variable Window and Local Variable Window

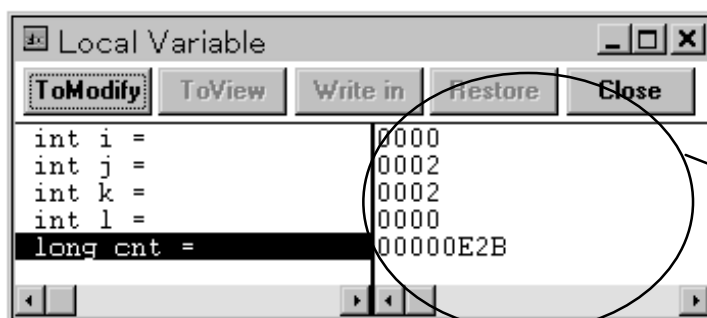
- The Variable window and Local Variable window give the user access to a variety of functions, such as the ability to modify the data number system.
- The available functions are listed below.

Function	Procedure
Display of variables of pointer type	A variable of pointer type is prefixed by + or -. Variable prefixed by +: The value of the variable indicated by the pointer is displayed by double-clicking. At this time, the prefix of the displayed variable changes to -. Variable prefixed by -: The display of the value of the variable indicated by the pointer is stopped by double-clicking. At this time, the prefix of the displayed variable changes to +.
Number system display selection	Select <u>V</u> iew -> <u>B</u> in, <u>O</u> ct, <u>D</u> ec, <u>H</u> ex, or <u>P</u> roper from the menu bar.

A variable prefixed by + or - is a variable of pointer type. Double-clicking a variable of pointer type prefixed by + displays the value of the variable indicated by the pointer. At this time, the prefix of the displayed variable changes from + to -.



Variable window



Data display.
ToModify button: Used to modify data. The user can choose binary, octal, decimal, hexadecimal, or automatic for display.

2.8 Using the Tracer Effectively

- The tracer records device operations in trace memory.
- The IE-78000-R-A has 32K frames of trace memory.
- Trace memory has a ring buffer structure.
- For combined events, four trace methods are supported:

Trace cycle	Trace mode		Remarks
Machine cycle trace	Total trace		Port trace operation is possible.
Event cycle trace	Total trace		Trace operation is performed only when the device performs a read, write, or fetch operation.
	Conditional trace	Sectional trace	The start and end of trace operation can be specified using an event condition.
		Qualified trace	Trace operation is performed only when an event condition match is detected.

- The trace-related windows are listed below.

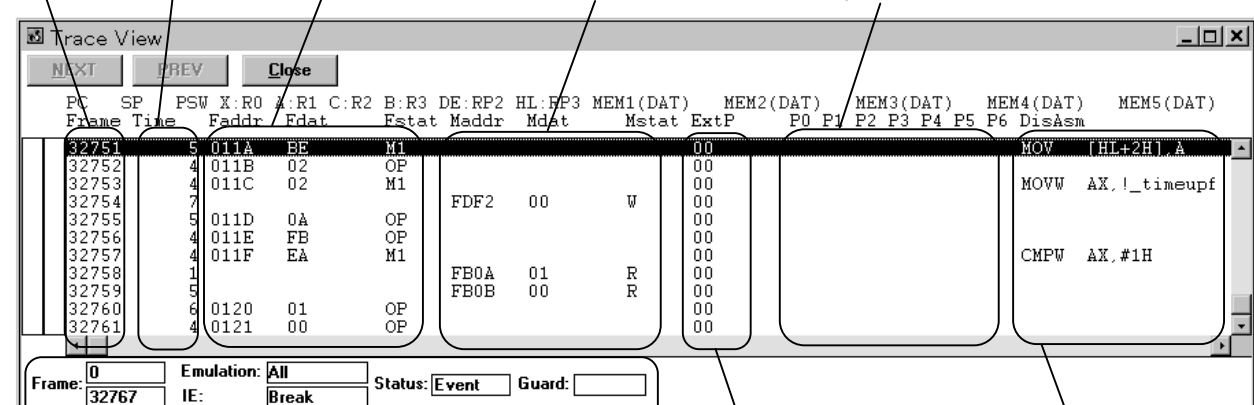
Operation		Window
Trace result display		Trace View window
Display item selection	Trace display	Show Trace dialog box
	Snapshot display	Snap Trace dialog box
Trace condition setting		Trace dialog box
Snapshot condition setting		Snap-Shot dialog box
Trace result search		Trace pick-up dialog box

2.8.1 Displaying Trace Results

- Trace results can be displayed in the Trace View window.

To display trace results:

Select **Browse -> Trace -> TraceView...** from the menu bar, or click the  button.



Frame number

The execution time between frames is counted.

Display of fetch-type access results. The results of program execution are displayed.

Display of data access results. The results of R/W accesses to memory are displayed.

Port trace results. In machine cycle trace, port trace operation is possible.

Trace status. Statuses such as the IE status at trace stop are displayed.

External sense clip input/output results are displayed.

Disassembly display. This display is provided when fetch-type access is performed and the status is M1.

Item	Description
Frame	Displays trace frame numbers. Valid range: $0 \leq \text{Trace frame number} \leq 32,767$
Time	Displays the number of clock pulses taken by the target chip between the start of execution of the immediately preceding trace address and the start of execution of the current trace address. For the clock signal, the CPU clock is not used. Instead, the 10-MHz clock signal of the in-circuit emulator is used. Measurement range: $1 < \text{Time tag} \leq 0\text{xfffff}$
Address Data Statu	Displays program fetch results. This field displays the following information depending on the fetch status displayed in the Status field: M1 : Fetch of the first byte of an instruction OP : Operation code fetch IF : Invalid fetch
Address Data Statu	Displays data access results. This field displays the following information depending on the access status display in the Status field: VECT : Vector read R : Data read W : Data write
ExtP	Displays the input level of the external sense clips when trace has been performed.
DisAsm	Displays the results of disassembly. This information is displayed only when the fetch status is M1 .

2.8.2 Saving and Referencing Trace Results

- Trace results can be saved to a file. A file containing saved trace results can be referenced.
- Trace results are saved in text format, allowing an editor to be used to reference saved trace results.

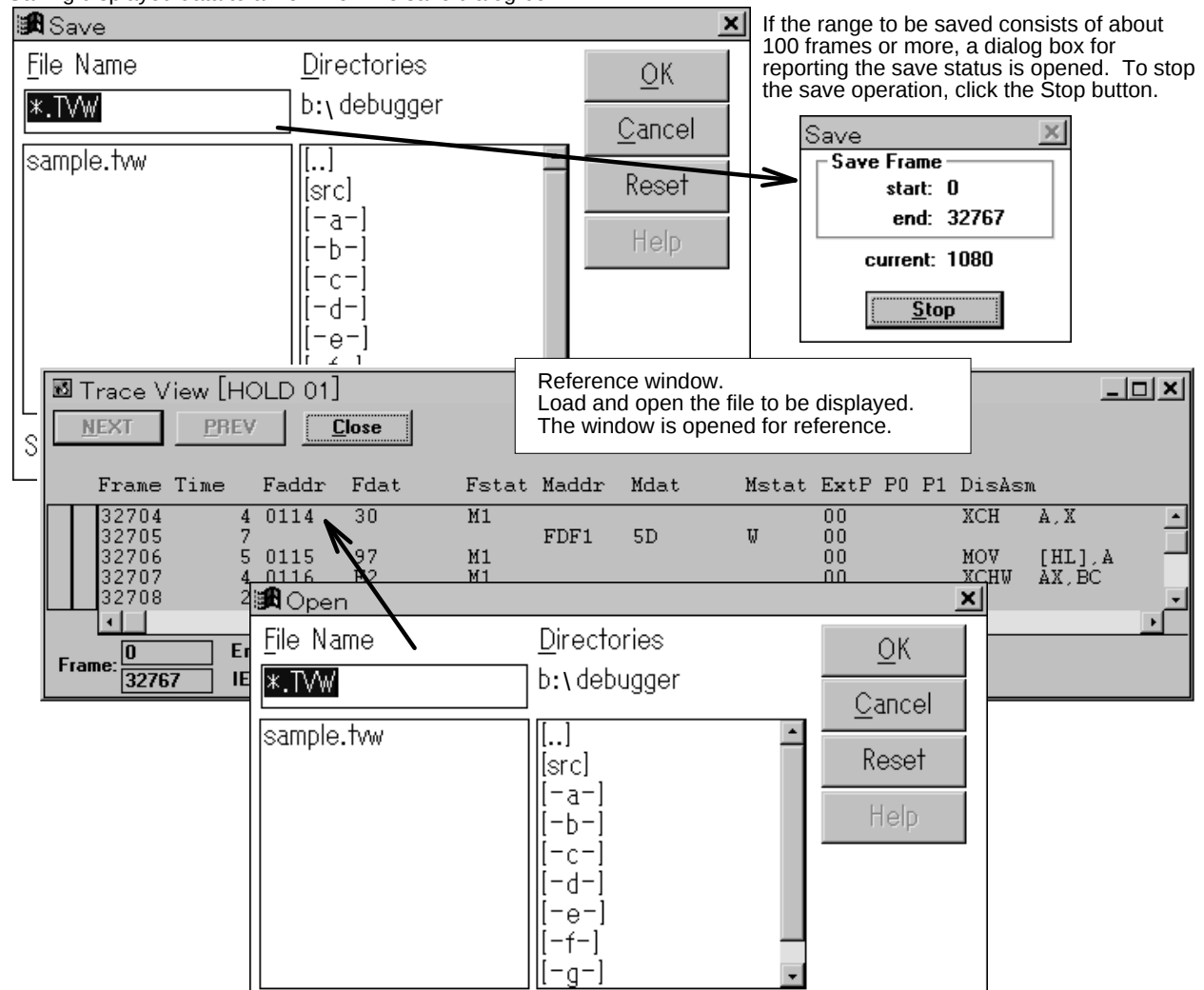
To save trace results to a file:

1. Activate the Trace View window.
2. Select File -> Save As... from the menu bar.
3. Save trace results with the View file save dialog box.

To open and reference a file containing saved trace results:

1. Activate the Trace View window.
2. Select File -> Open... from the menu bar.
3. Load the file to be referenced by using the View file load dialog box.

Saving displayed data to a file: View file save dialog box



Displaying data saved to a file: View file load dialog box

2.8.3 Effective Trace Memory Usage 1 (Trace Mode Setting)

- Trace memory can be used effectively by setting a trace condition and trace mode.
- Three major trace modes are supported:

Trace mode	Description
Total trace	All accesses are traced. The user can choose between machine cycle trace and event cycle trace.
Sectional trace	A section from one event to another is traced. This mode is useful, for example, for tracing one particular function.
Qualified trace	Only an event condition match point is traced. This mode is useful, for example, for tracing particular memory accesses.

- Each trace mode is described below.
 1. The data for each of the following modes indicates the results of executing test program 1 from address 80H to address 8EH.
 2. Test program 1 initializes, to zero, the four bytes of memory from address 0FE00H to address 0FE03H.

Test program 1: Clearing RAM

Addr	Data	Mnemonic	
0080	61D0	SEL	RB0
0082	16FFFC	MOVW	HL,#0FCFFH
0085	A100	MOV	A,#0H
0087	A304	MOV	B,#4H
0089	BB	MOV	[HL+B],A
008A	8BFD	DBNZ	B,\$89H
008C	00	NOP	
008D	00	NOP	
008E	FAFE	BR	\$8EH

1. Results of total trace (event cycle trace)

- ◆ The results of total trace are indicated below.
- ◆ All accesses are traced, so that all program operations can be identified.
- ◆ Program fetch operations, and data read and write operations are traced.

frame	Faddr	Fdat	Fstat	Maddr	Mdat	Mstat	DisAsm
32738	0080	61	M1				SEL RB0
32739	0081	D0	OP				
32740	0082	16	M1				MOVW HL, #0FCFFH
32741	0083	FF	OP				
32742	0084	FC	OP				
32743	0085	A1	M1				MOV A, #0H
32744	0086	00	OP				
32745	0087	A3	M1				MOV B, #4H
32746	0088	04	OP				
32747	0089	BB	M1				MOV [HL+B], A
32748	008A	8B	M1				DBNZ B, \$89H
32749				FD03	00	W	
32750	008B	FD	OP				
32751	0089	BB	M1				MOV [HL+B], A
32752	008A	8B	M1				DBNZ B, \$89H
32753				FD02	00	W	
32754	008B	FD	OP				
32755	0089	BB	M1				MOV [HL+B], A
32756	008A	8B	M1				DBNZ B, \$89H
32757				FD01	00	W	
32758	008B	FD	OP				
32759	0089	BB	M1				MOV [HL+B], A
32760	008A	8B	M1				DBNZ B, \$89H
32761				FD00	00	W	
32762	008B	FD	OP				
32763	008C	00	M1				NOP
32764	008D	00	M1				NOP
32765	008E	FA	M1				BR \$8EH
32766	008F	FE	OP				

- ◆ Total trace mode is set as follows:

1. Disable all trace event conditions.
2. Select a trace cycle.
Select Execute -> Trace -> Machine All. Trace or Event All. Trace from the menu bar.

2. Results of sectional trace

- ◆ The results of sectional trace from address 89H to address 8CH are shown below.
- ◆ The range to be traced can be specified, such that trace memory is used effectively.

frame	Faddr	Fdat	Fstat	Maddr	Mdat	Mstat	DisAsm
32750	0089	BB	M1				MOV [HL+B], A
32751	008A	8B	M1				DBNZ B, \$89H
32752				FD03	00	W	
32753	008B	FD	OP				
32754	0089	BB	M1				MOV [HL+B], A
32755	008A	8B	M1				DBNZ B, \$89H
32756				FD02	00	W	
32757	008B	FD	OP				
32758	0089	BB	M1				MOV [HL+B], A
32759	008A	8B	M1				DBNZ B, \$89H
32760				FD01	00	W	
32761	008B	FD	OP				
32762	0089	BB	M1				MOV [HL+B], A
32763	008A	8B	M1				DBNZ B, \$89H
32764				FD00	00	W	
32765	008B	FD	OP				
32766	008C	00	M1				NOP

- ◆ Sectional trace mode and the event conditions can be set as shown below.
- ◆ By enabling trace event conditions, a conditional trace operation can be performed.

The screenshot displays three overlapping windows from a software interface:

- Event Set (Top Left):** Shows 'Event01' with Address '89h', Status 'Run', and External '00'.
- Event Set (Top Right):** Shows 'Event02' with Address '8ch', Status 'Run', and Mask '0000'.
- Event Manager (Middle):** Contains buttons 'Enable', 'Disable', and 'Delete'. It lists 'Event01', 'Event02', and 'Trace01'.
- TRACE (Bottom):** Shows 'Trace01' with 'Section' mode selected. It has fields for 'Section Start' (Event01) and 'Section End' (Event02).

Annotations and Text:

- Arrows point from the 'Event01' and 'Event02' entries in the Event Manager to their respective 'Event Set' windows.
- An arrow points from the 'Section Start' field in the TRACE window to the 'Event01' entry in the Event Manager.
- Another arrow points from the 'Section End' field in the TRACE window to the 'Event02' entry in the Event Manager.
- A text box on the left states:

Set sectional trace conditions as follows:
 Start: Event01
 End: Event02

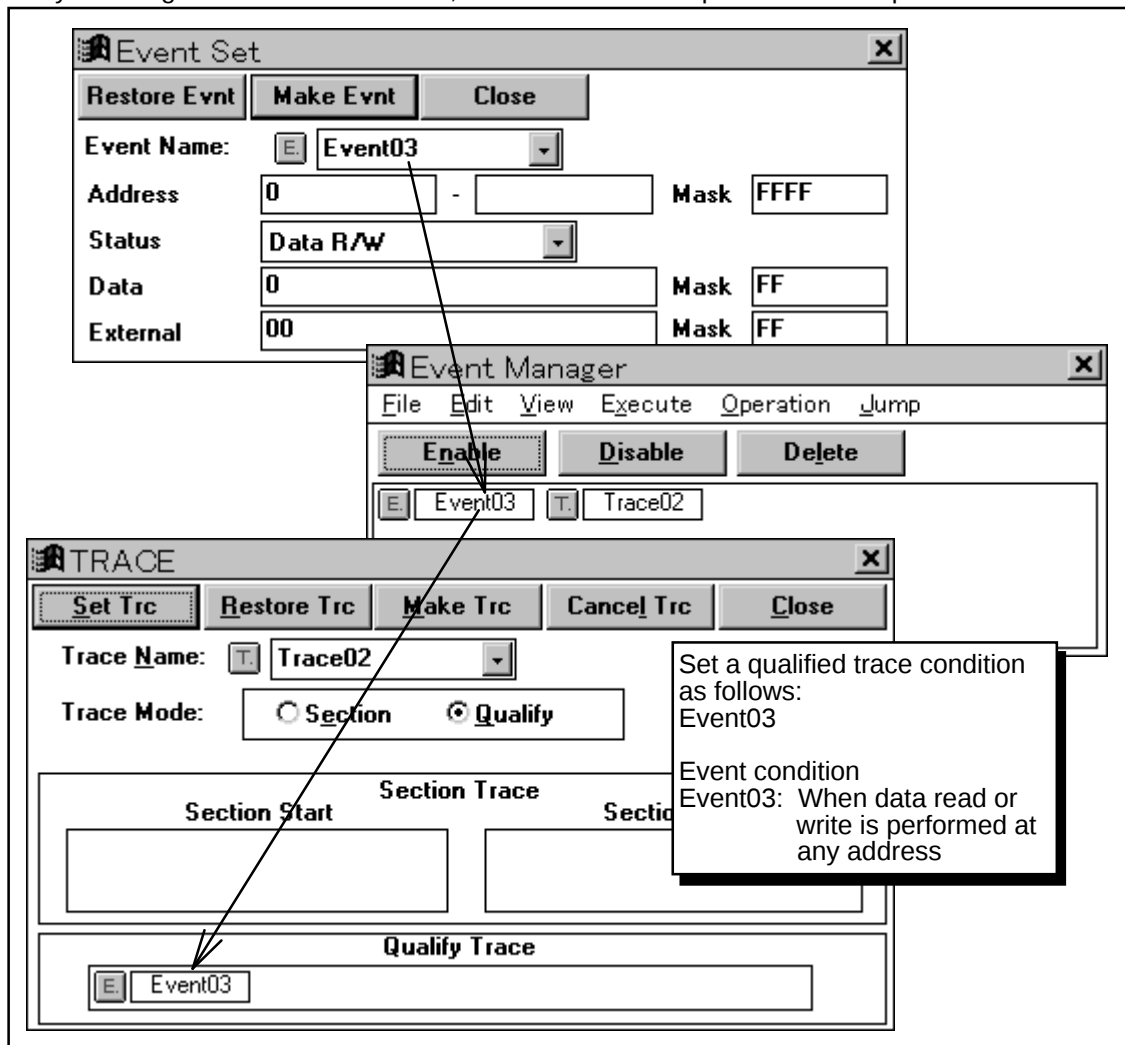
Event occurrence conditions
 Event01: Execution of address 89h
 Event02: Execution of 8ch

3. Results of qualified trace

- ◆ The results of qualified trace, for data read and write operations only, are indicated below.
- ◆ Only those points that are to be traced are traced, allowing large amounts of data to be traced.
Note, however, that since trace is performed only when an event condition match is detected, the context is difficult to grasp.

frame	Faddr	Fdat	Fstat	Maddr	Mdat	Mstat	DisAsm
32763				FD03	00	W	
32764				FD02	00	W	
32765				FD01	00	W	
32766				FD00	00	W	

- ◆ Qualified trace mode and an event condition can be set as shown below.
- ◆ By enabling a trace event condition, a conditional trace operation can be performed.



2.8.4 Effective Trace Memory Usage 2 (Trace Full Break, Snapshot Trace)

- Trace full break
 1. Trace memory has a ring buffer structure. This means that, once the trace memory is filled with trace data, the existing trace data is overwritten by the new data, starting from the oldest data.
 2. To preserve the trace results, trace operation can be stopped once the trace memory is full.

Trace full break setting:
Select **Execute** -> Trace **Full Break** from the menu bar.

- Snapshot trace
 1. Trace memory is used to store the execution history. In addition to the execution history, other data can be stored by specifying a snapshot event.
 2. The snapshot trace function writes specified data into trace memory when a condition is satisfied. The following data can be written:

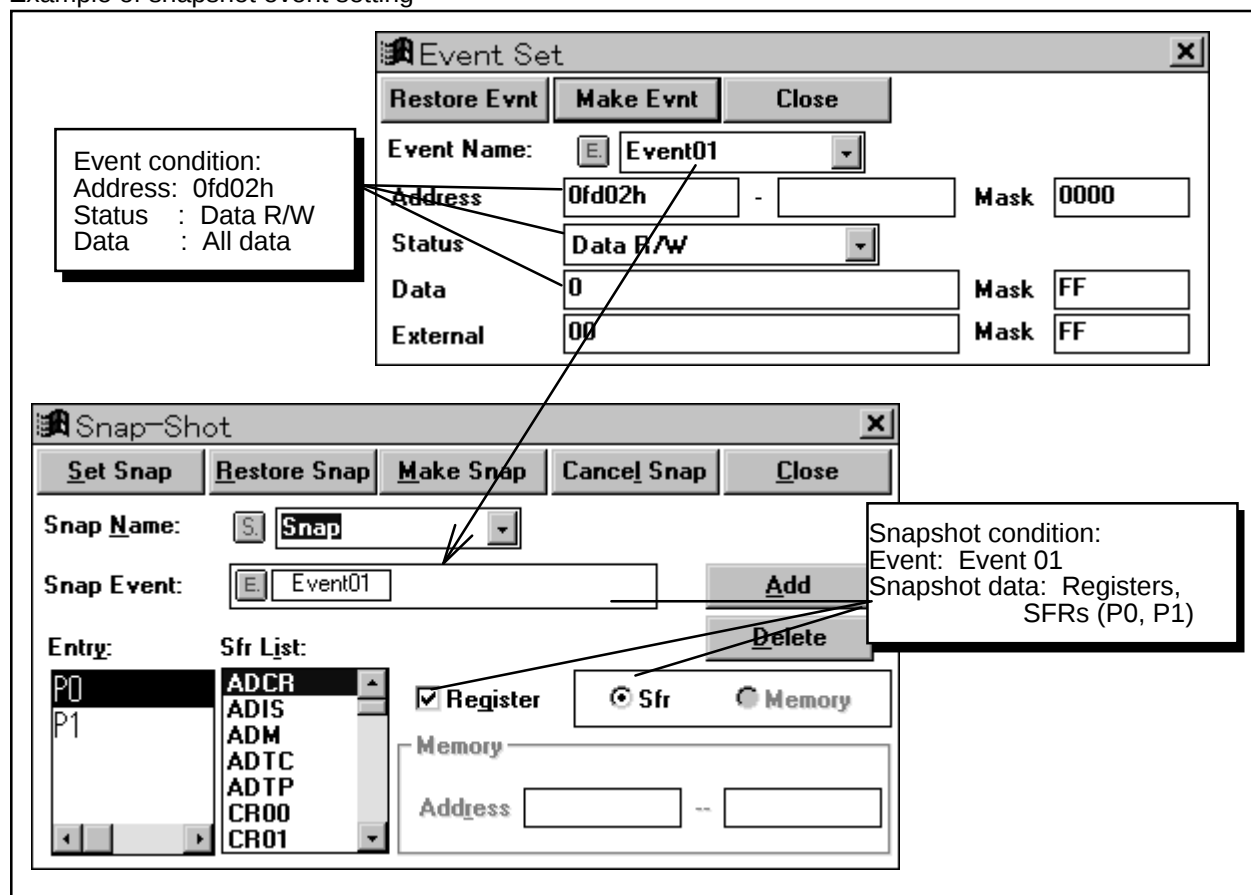
Data		Description
Register		All registers of the current bank (PC, SP, PSW, AX, BC, DE, HL)
Data	SFR	Up to five points in SFRs or memory can be traced.
	Memory	

3. Before data is written into trace memory, the execution of the user program is stopped.
4. For the program below, the method of writing the register and SFRs (P0, P1) when address 0fd02h is accessed is shown.

Test program 1: Clearing RAM

Addr	Data	Mnemonic	
0080	61D0	SEL RB0	Selects register bank 0.
0082	16FFFC	MOVW HL,#0FCFFH	Sets an initialization start address, minus 1.
0085	A100	MOV A,#0H	Sets initialization data.
0087	A304	MOV B,#4H	Sets the number of bytes to be initialized.
0089	BB	MOV [HL+B],A	Initializes memory.
008A	8BFD	DBNZ B,\$89H	Determines termination.
008C	00	NOP	
008D	00	NOP	
008E	FAFE	BR \$8EH	

Example of snapshot event setting



Trace data

PC	SP	PSW	X:R0	A:R1	C:R2	B:R3	DE:RP2	HL:RP3	MEM1(DAT)	MEM2(DAT)
frame	Faddr	Fdat	Fstat	Maddr	Mdat	Mstat	DisAsm			
32742	0089	BB	M1					MOV	[HL+B],A	
32743	008A	8B	M1					DBNZ	B,\$89H	
32744				FD03	00	W				
32745	008B	FD	OP							
32746	0089	BB	M1					MOV	[HL+B],A	
32747	008A	8B	M1					DBNZ	B,\$89H	
32748				FD02	00	W				
32749	008B	FD	OP							
0089	FEE0	02	00	00	02	0000	FCFF	P0(00)	P1(00)	
32755	0089	BB	M1					MOV	[HL+B],A	
32756	008A	8B	M1					DBNZ	B,\$89H	
32757				FD01	00	W				
32758	008B	FD	OP							
32759	0089	BB	M1					MOV	[HL+B],A	
32760	008A	8B	M1					DBNZ	B,\$89H	
32761				FD00	00	W				
32762	008B	FD	OP							

- ◆ In frame 32748, a match with event condition Event01 was detected, causing snapshot event Snap to occur.
- ◆ Between frame 32749 and frame 32755, the debugger stopped once to write snapshot data into the tracer.

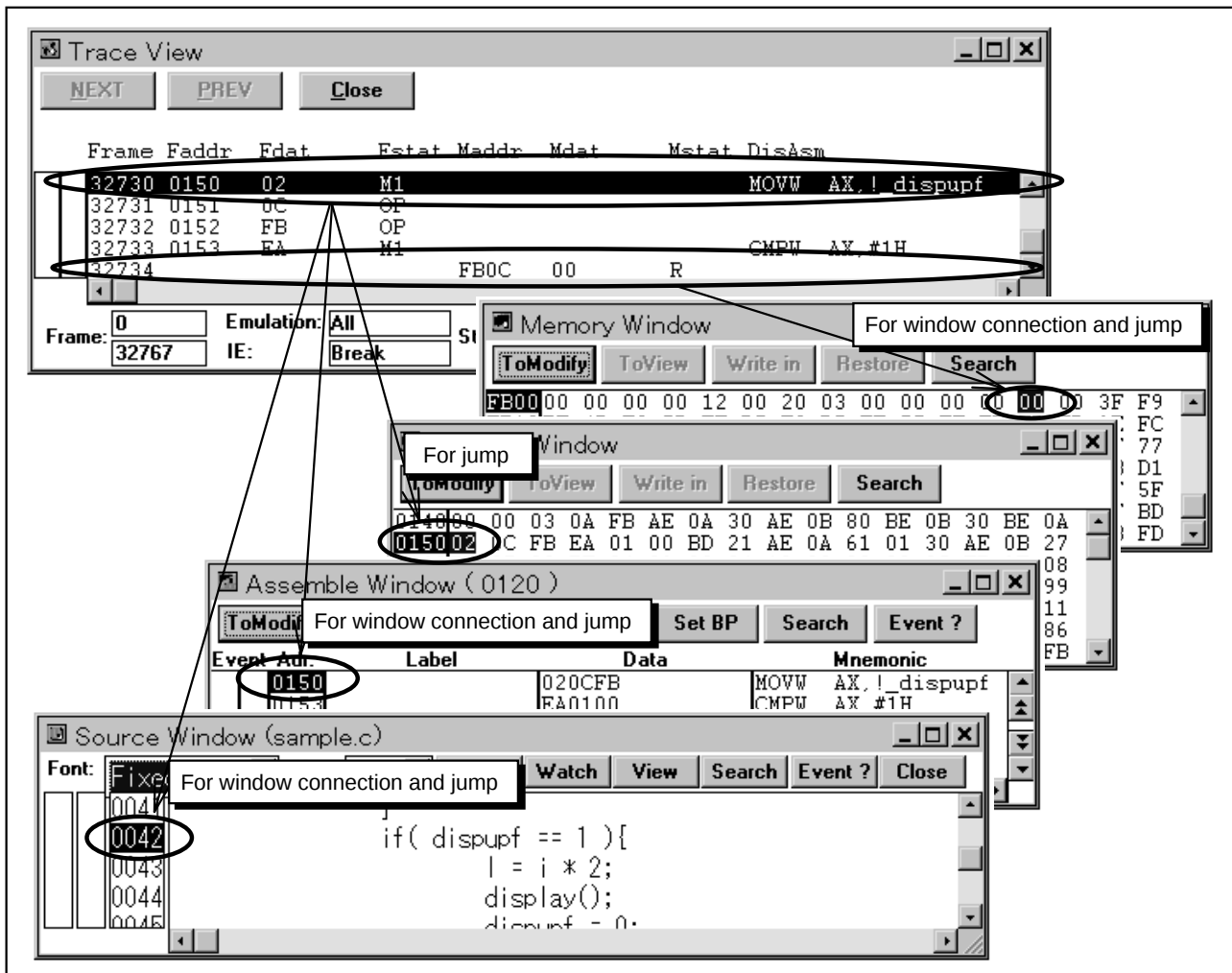
2.8.5 Inter-Window Connection Functions (Window Connection Function, Jump Function)

- Window connection function: This function displays trace results in each window. When the user positions the cursor to the Trace View window, each of the windows (Source window, Assemble window, and Memory window) can be manipulated interactively, thus allowing trace operation in each window.
- Jump function: A jump can be made to a position in the Source window, Assemble window, and Memory window corresponding to the address value of a frame line specified in the Trace View window.
- When the window connection function is used, the Trace View window remains active. When the jump function is used, however, the jump destination window becomes the active window.

	Function	Operation
Window connection function	Connection to the Source window	Activate the Trace View window. Select <u>W</u> indow Connect -> <u>S</u> ourceText from the menu bar.
	Connection to the Assemble window	Activate the Trace View window. Select <u>W</u> indow Connect -> <u>A</u> ssemble from the menu bar.
	Connection to the Memory window	Activate the Trace View window. Select <u>W</u> indow Connect -> <u>M</u> emory from the menu bar.
Jump function	Jump to the Source window	Select a frame in the Trace View window. Select <u>J</u> ump -> <u>S</u> ourceText... from the menu bar, or press CTRL+U .
	Jump to the Assemble window	Select a frame in the Trace View window. Select <u>J</u> ump -> <u>A</u> ssemble... from the menu bar, or press CTRL+A .
	Jump to the Memory window	Select a frame in the Trace View window. Select <u>J</u> ump -> <u>M</u> emory... from the menu bar, or press CTRL+M .

- With the window connection function and jump function, connection is made to the data in each window as follows:

Function	Window connection	Jump function
Source window	Fetch address	Fetch address
Assemble window		
Memory window	Data read address and data write address	Fetch address, data read address, and data write address



2.9 Measuring the Execution Time

- The IE-78000-R-A has two timers. One timer measures the time from the start of execution to the end of trace operation. The other timer measures the time from the start of the previous trace operation to the start of the current trace operation.
- The specifications of the two timers are as follows:

Timer	Maximum measurement time	Minimum measurement time
For execution time measurement	Approx. 14 minutes and 18 seconds	Approx. 500 nanoseconds
For trace interval measurement (time tag)	Approx. 1.677 seconds	Approx. 100 nanoseconds

2.9.1 Measuring Program Execution Time

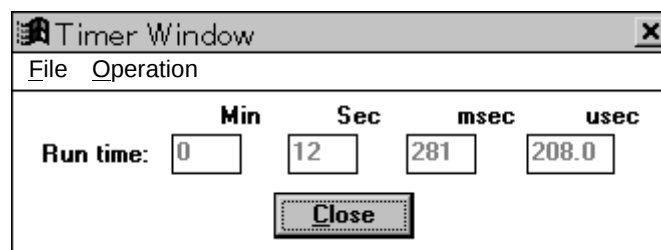
- The time from the start of program execution to the end of program execution is displayed in the Timer window.
- The measurement time depends on the execution mode, as indicated below.

Execution mode	Measurement section
Step execution	Last instruction
Real-time execution	From the start of execution to a break
Non-break real-time execution	From the start of execution to termination of the tracer

- The Timer window can be opened as follows:

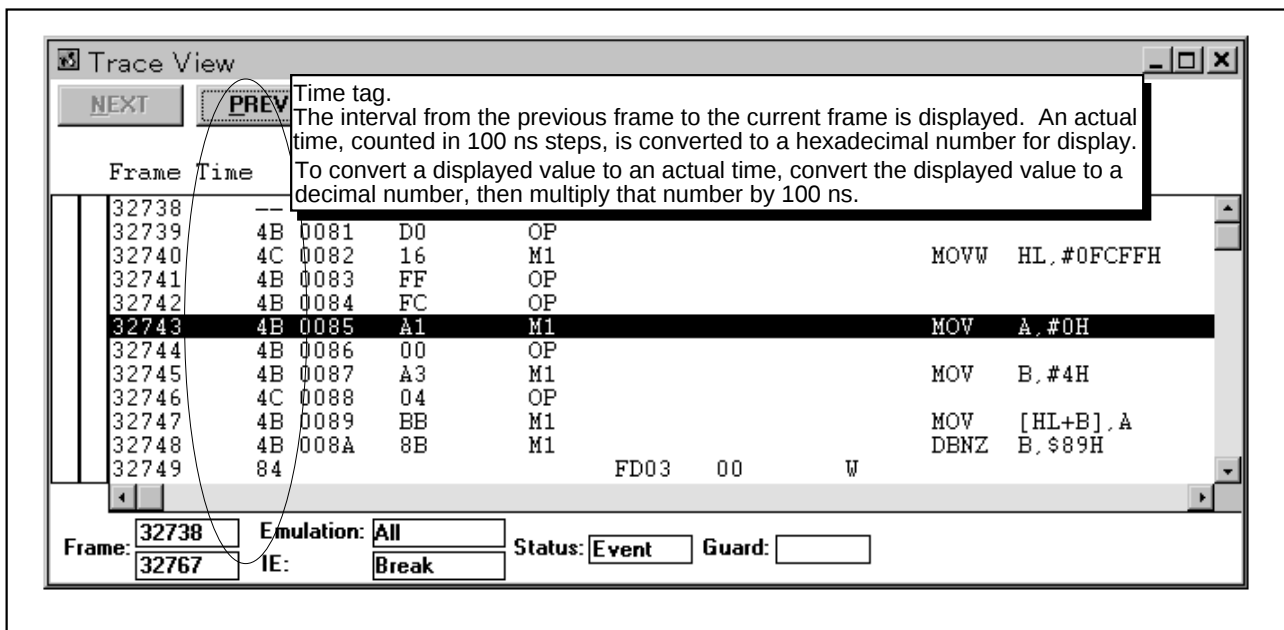
Execution time display:

Select **Browse -> Timer...** from the menu bar, or press the  button.



2.9.2 Time Measurement Using the Tracer

- For measurement of a short section, regularly executed processing, and so forth, the time tag is useful.
- When compared with the execution times displayed in the Timer window, shorter times are obtained with the time tag. However, the time tag stores multiple data items in trace memory, so that information such as time distribution data can be checked using a separate tool.
- With the time tag, the time from the start of the previous trace operation to the start of the current trace operation is measured. This measurement is conducted not only while the program is being executed but also while the program is stopped. This means that the time tag data for the first program execution frame is meaningless.



Chapter 3 Advanced Use of ID78K0

This chapter describes several advanced uses of the ID78K0. Note that these uses are usually not essential to normal operation.

3.1 Verifying the Validity of Evaluation

Evaluation is essential to the development of a program. If the evaluation of a particularly important item is omitted for some reason, bugs may remain in a program that is offered for retail sale. This section describes the use of the coverage functions to verify the validity of evaluation.

3.2 Using External Sense Clips

The in-circuit emulator status or the contents of memory can be output in real time, by using external sense clips together with event conditions. This section describes the use of the external sense clips.

3.3 Measuring Time by Setting Conditions

Basically, the Timer window of the ID78K0 supports only the measurement of the time that elapses between the start and end of program execution. Shorter periods can, however, be measured by using the tracer in combination with events.

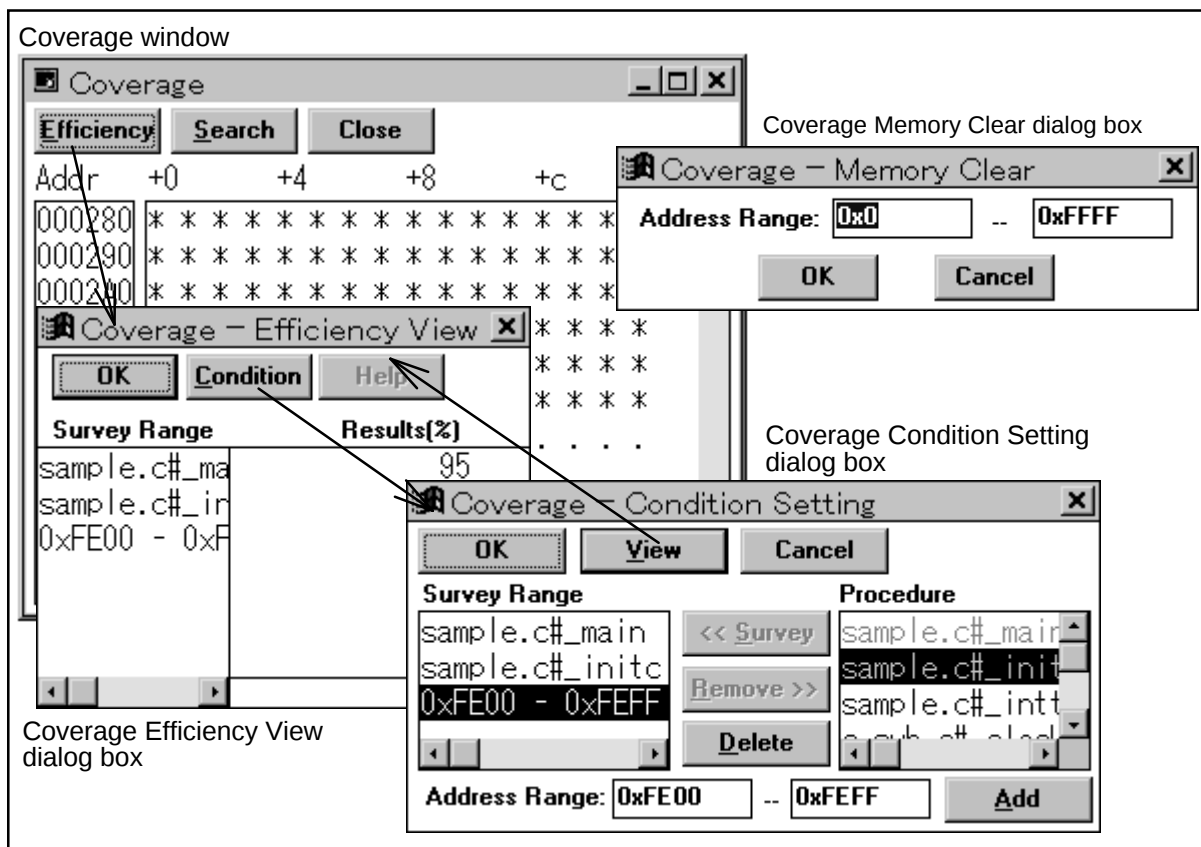
3.1 Verifying the Validity of Evaluation

- Evaluation is essential to the development of a program. If the evaluation of a particularly important item is omitted for some reason, bugs may remain in a program that is offered for retail sale.
- This section describes the use of the coverage functions to verify the validity of evaluation. Note, however, that the validity of evaluation cannot be completely verified based on only the results of coverage.

3.1.1 Coverage

- Coverage is a record of the flow of the execution of a program. While the tracer can trace program execution backwards, coverage merely indicates whether specified instructions within a program have actually been executed.
- The debugger supports coverage for the read, write, and fetch operations.
- The results of coverage can be displayed in the Coverage window.
- The following window and dialog boxes are used for coverage:

Window	Description
Coverage window	Displays the results of coverage.
Coverage Efficiency View dialog box	Displays the coverage results, as a percentage, for each function or specified address range.
Coverage Condition Setting dialog box	Used to add items to be displayed in the Coverage Efficiency View dialog box.
Coverage Memory Clear dialog box	Initializes the coverage memory.



3.1.2 Verifying the Validity of Evaluation Based on Coverage

- Ideally, all possible patterns of program execution should be evaluated. Due to time or other restrictions, however, evaluation may have to be restricted by, for example, sampling and combining several patterns. Evaluation based on sampled patterns must, however, be checked for validity.
- One method of verifying the validity of evaluation is the use of the coverage results to check whether all instructions have been executed.
- The above check can easily be performed by using the Coverage window and the memory map in the link list file (.MAP), output upon linkage of the program.

Verification based on coverage

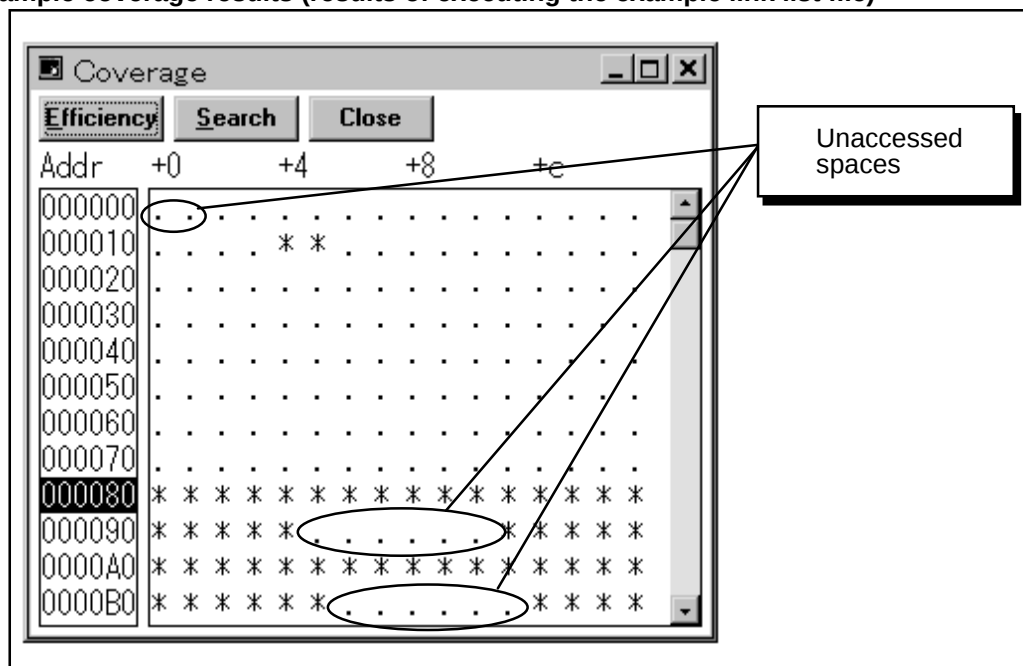
1. Refer to the memory map in the link list file to identify any free spaces (* gap *) in the program.
2. Refer to the contents of the Coverage window to check whether all memory spaces other than the free spaces, identified in step 1, have been accessed (read, written, or fetched).
3. If any unaccessed space is revealed by step 2, check the program and review the evaluation items. If any free space in the program has been accessed, check that space by, for example, setting event conditions.

Example link list file

*** Memory map ***					
SPACE=REGULAR					
MEMORY=ROM					
BASE ADDRESS=0000H SIZE=8000H					
OUTPUT	INPUT	INPUT	BASE	SIZE	
SEGMENT	SEGMENT	MODULE	ADDRESS		
@@VECT00			0000H	0002H	CSEG AT
	@@VECT00	@cstart	0000H	0002H	
* gap *			0002H	0012H	
	@@VECT14		0014H	0002H	CSEG AT
	@@VECT14	SAMPLE	0014H	0002H	
* gap *			0016H	002AH	
	@@CALT		0040H	0000H	CSEG CALLT0
		@@CALT @cstart	0040H	0000H	
		@@CALT SAMPLE	0040H	0000H	
		@@CALT C_SUB	0040H	0000H	
* gap *			0040H	0040H	
Intermediate lines omitted					
@@CNST			0080H	0000H	CSEG UNITP
	@@CNST @cstart		0080H	0000H	
	@@CNST SAMPLE		0080H	0000H	
	@@CNST C_SUB		0080H	0000H	
@@CODE			0080H	02D1H	CSEG
	@@CODE @cstart		0080H	0050H	
	@@CODE SAMPLE		00D0H	0130H	

In this example, 0002H to 0013H, 0016H to 003FH, and 0040H to 007FH are free spaces.

Example coverage results (results of executing the example link list file)



In this example, the reset vector at addresses 0 and 1 has not been accessed. The operation performed upon a reset must, therefore, be evaluated.

3.1.3 Notes on Coverage Results

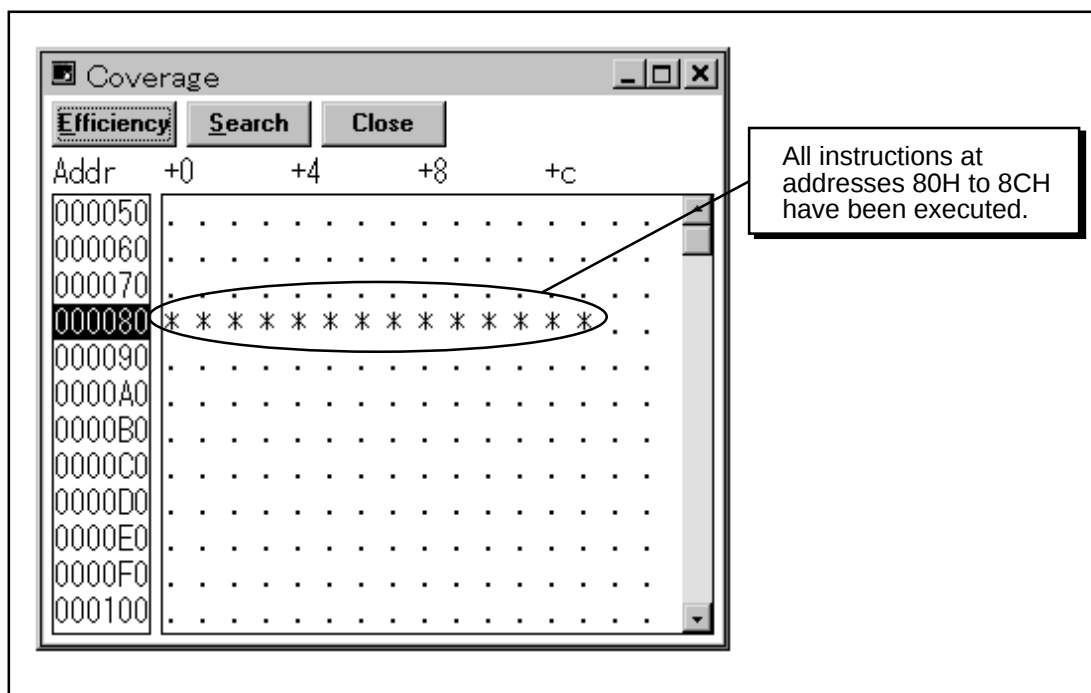
- When checking the coverage results, note the results of conditional branches.
- The IE-78000-R-A supports C0 coverage, which cannot be used to check how processing has branched at a conditional branch instruction.

Example of execution of a conditional branch instruction

1. When the following program is executed from address 80H to 8CH, a conditional branch instruction is executed at address 86H. Execution jumps to address 88H because the condition is false.

Addr	Data	Mnemonic	
0080	A101	MOV A, H	; Assume that the H register contains 1.
0082	A302	MOV B, #2H	
0084	4D01	CMP A, #1H	
0086	BD02	BNZ \$8AH	
0088	610B	ADD A, B	
008A	A200	MOV C, #0H	
008C	00	NOP	

2. The coverage results are as follows, indicating that all instructions have been executed.



3. Actually, however, the condition may be true, depending on the stored data, thus causing address 88H to be skipped. In such a case, the coverage results do not cover all evaluation items.

3.2 Using External Sense Clips

- External sense clips have various functions. They can be used to post notification of the in-circuit emulator status or output 1-byte RAM data in real-time.
- The use of external sense clips may enable essential processing which has not been possible conventionally.
- External sense clips No. 01 to 08 are provided. The debugger handles them as bits 0 to 7, respectively.

External sense clip number	Debugger handles as:
No.08	Bit 7
No.07	Bit 6
No.06	Bit 5
No.05	Bit 4
No.04	Bit 3
No.03	Bit 2
No.02	Bit 1
No.01	Bit 0

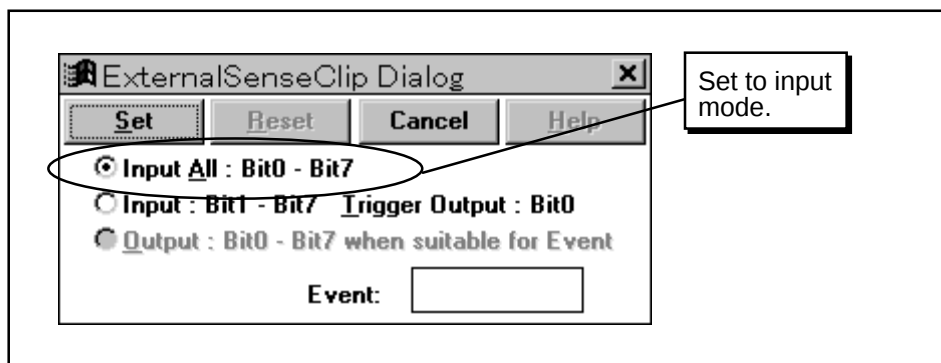
- When external sense clips are set to output mode, they must be pulled up using resistors. In such a case, a voltage exceeding +15 V cannot be applied to the sense clips.
- The tracer traces the potential difference between each external sense clip and GND, regardless of whether the sense clips are set to input or output mode. The HC4050B (used as an input buffer) determines whether the trace data for each external sense clip is 1 or 0.
- The trace data for external sense clips can be used for event conditions, thus enabling the setting of a wide range of event conditions.

3.2.1 Tracing External Data

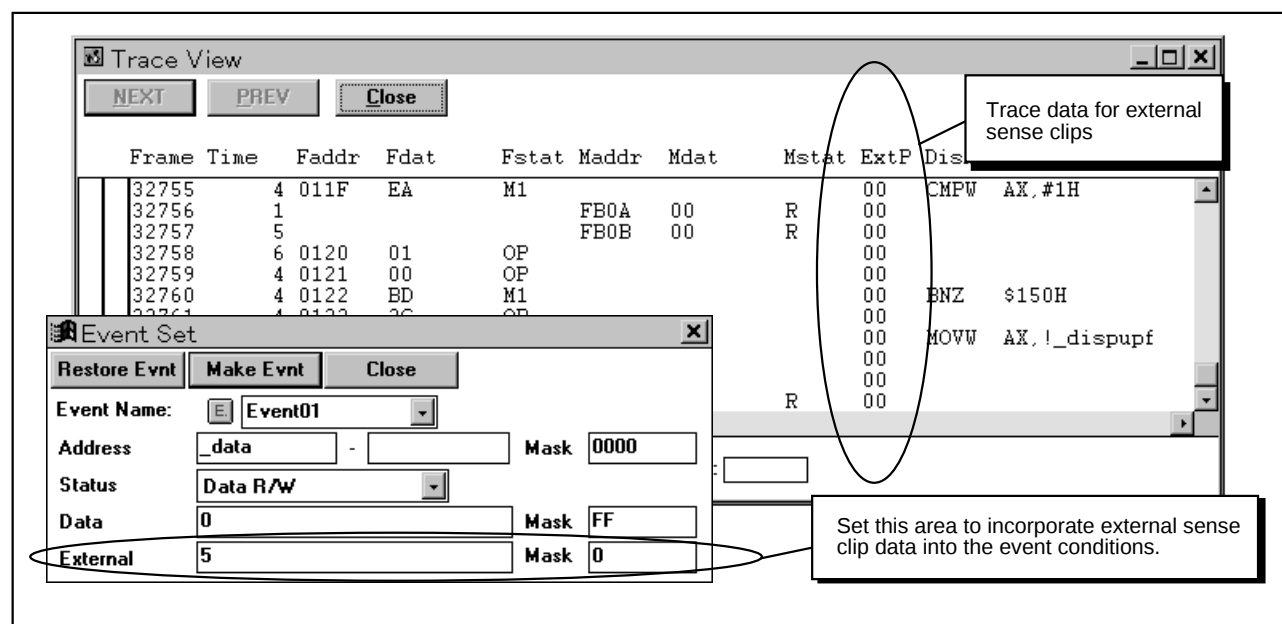
- To trace the state of each pin of the target device, set the external sense clips to input mode (default). Input data can be incorporated into event conditions, such that an event can be triggered by an external source.

Setting procedure

- Set the external sense clips to input mode.

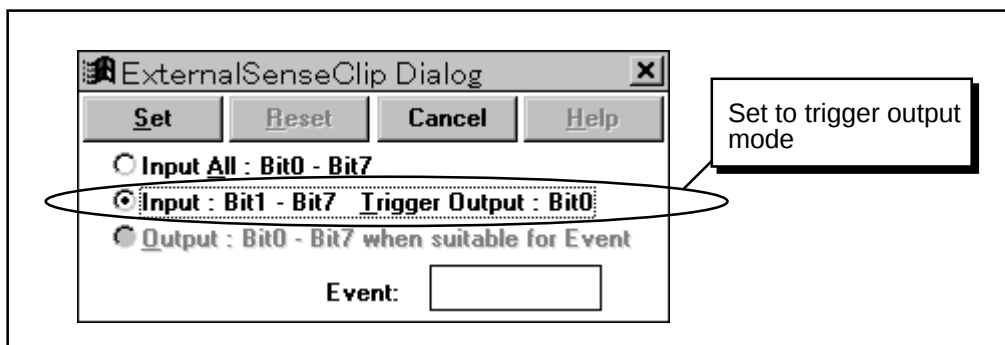


- Connect external sense clips to the pins to be traced.
- To set an event, set event conditions using the Event Set dialog box. The results of trace can be checked using the Trace View window.



3.2.2 Trigger Output

- To output the in-circuit emulator status or other data, set the external sense clips to trigger output mode.



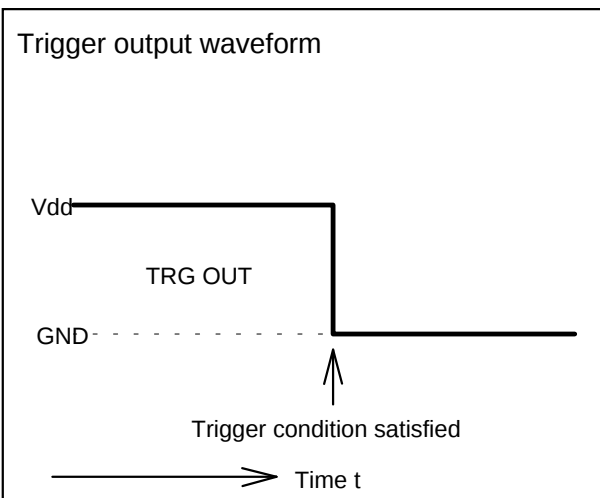
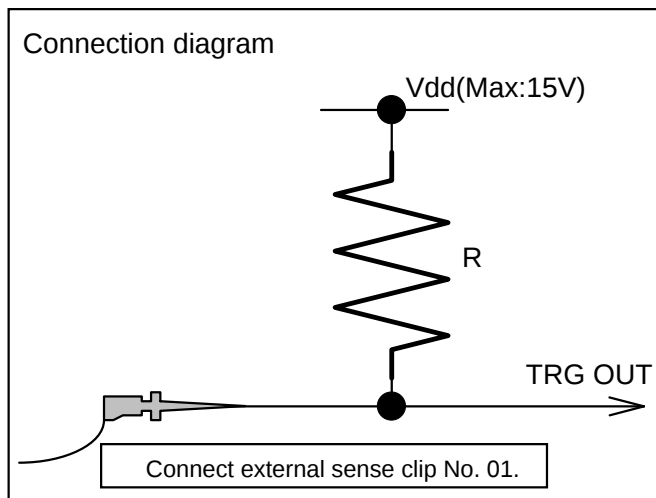
- Trigger output data is output under the following condition:

Trigger output condition

When the pass count becomes 0 upon the occurrence of a break event

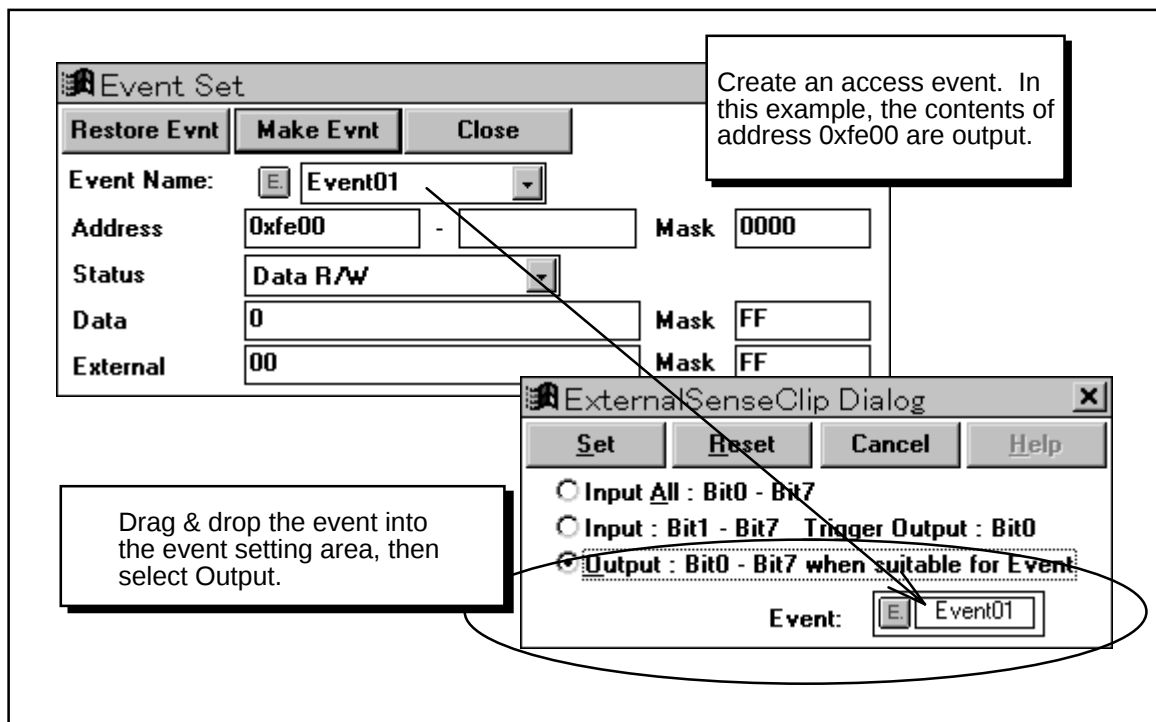
Trigger data is not output upon the occurrence of a fail-safe or manual break.

- Trigger output data is output from external sense clip No. 01.
- When external sense clips are set to output mode, they must be pulled up using resistors.

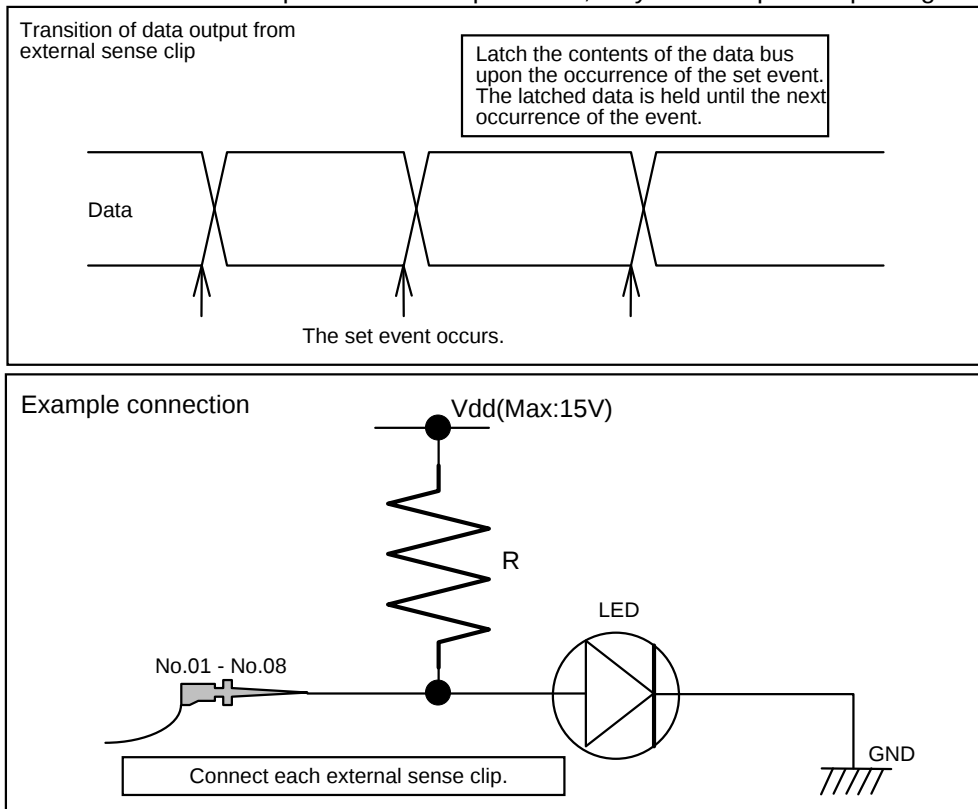


3.2.3 Real-Time RAM Output

- The IE-78000-R-A does not support real-time RAM sampling. Only 1-byte data in memory can be output in real-time, by using event conditions in combination with external sense clips and simple external jigs.



- Because data is being output in real time, it can also be used as a trigger source for other targets.
- When external sense clips are set to output mode, they must be pulled up using resistors.



3.2.4 Creating an Event by ANDing a Data Condition

- Advanced events can be created by using external sense clips.
- An event condition can be created by ANDing a data condition, as follows:

Procedure for creating an event condition by ANDing a data condition

1. Create an event having a data condition.
2. Set the event created in step 1 as an output condition for the external sense clips.
3. Pull up the eight external sense clips using resistors.
4. Create an event having an address condition or execution condition.
5. Specify an external sense data condition for the event created in step 4, thus creating an event having a condition ANDing those specified in steps 1 and 4.

Example

Causing an event to occur when 15h is written into address 0fe12h, provided address 0fe00h contains 20h

The image shows two screenshots of the 'Event Set' dialog box and an 'ExternalSenseClip Dialog' box, illustrating the steps to create an event by ANDing data conditions.

Top Screenshot (Event Set Dialog):

- Event Name:** Event01
- Address:** 0fe00h
- Status:** Data R/W
- Data:** 00
- External:** 00
- Masks:** Mask 0000, Mask FF, Mask FF

Annotations:

- A box points to the 'Event Name' field: "Set Address, Status, and Data. Set External as the default or Mask in ff."
- A box points to the 'Status' dropdown: "Set an event condition, and set the external sense clips to output mode."

Bottom Screenshot (Event Set Dialog):

- Event Name:** Event02
- Address:** 0fe15h
- Status:** Data Write
- Data:** 15h
- External:** 20
- Masks:** Mask 0000, Mask 00, Mask 00

Annotations:

- A box points to the 'Event' field in the 'ExternalSenseClip Dialog' (which is set to Event01): "Finally, set Address, Status, and Data for another event. Specify the value at address 0fe00h for External."

The 'ExternalSenseClip Dialog' also shows options for output: "Output All : Bit0 - Bit7", "Output : Bit1 - Bit7", and "Output : Bit0 - Bit7 when suitable for Event".

In this example, one data condition is ANDed with another data condition. A data condition can also be ANDed with an execution condition. An event condition like that shown above can be set for various events, thus enabling the creation of advanced events. First try specifying "on."

3.3 Measuring Time by Setting Conditions

- The timer measurement function of the IE-78000-R-A does not support the setting of event conditions. The user may, however, require information such as the intervals that elapse between a function being called, or whether timer interrupts are generated correctly and on time.
- Time measurement using event conditions in combination with the tracer is described below.

Setting procedure

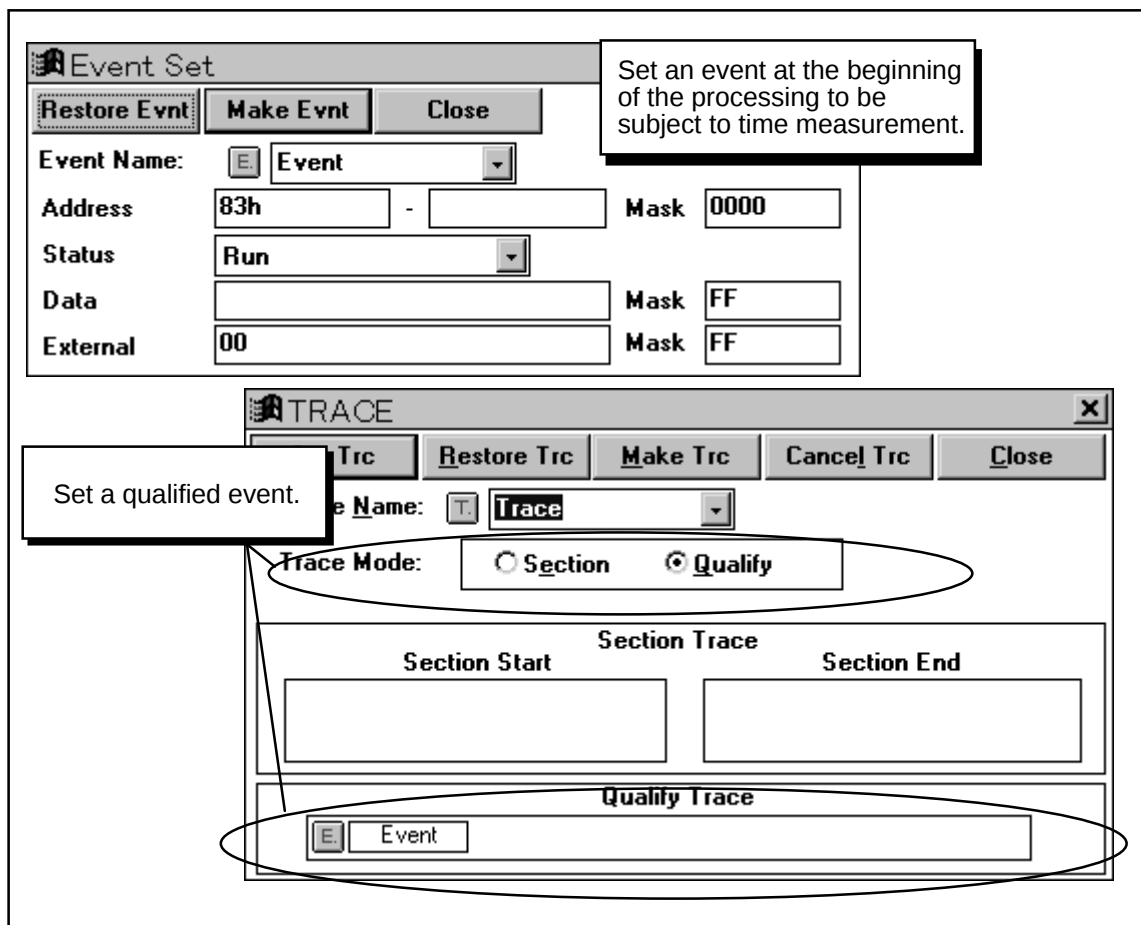
1. Set an event at the beginning of the function for which time will be measured. The following program is used as an example:

Example program to be subject to time measurement

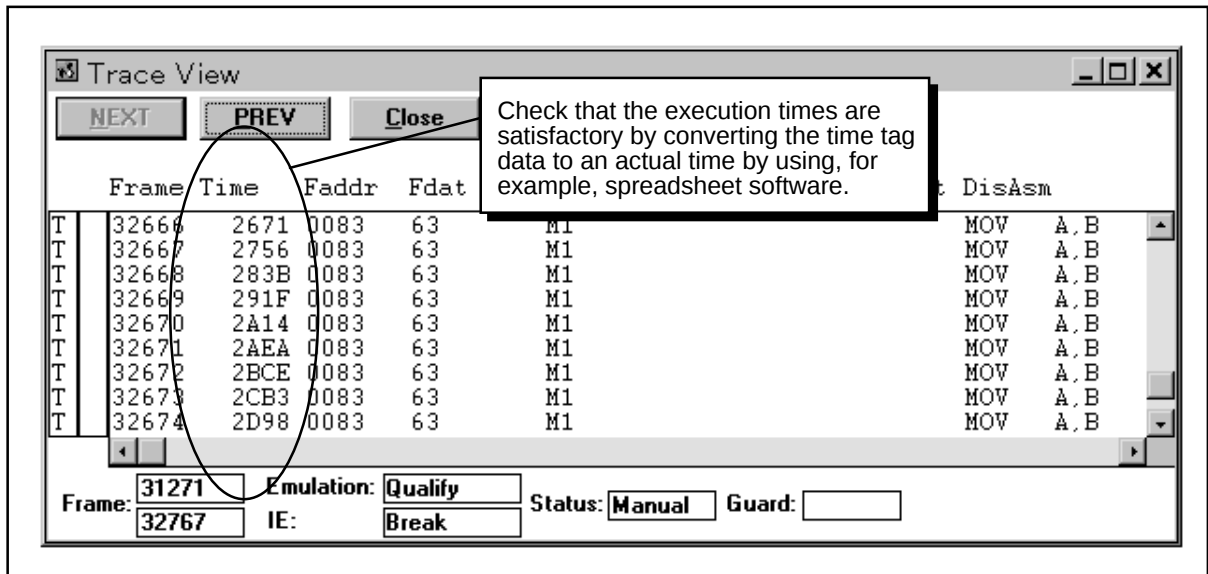
	Addr	Data	Mnemonic	
>	0080	A300	MOV	B, #0H
	0082	43	INC	B
T	0083	63	MOV	A, B ; Set an event.
	0084	8BFE	DBNZ	B, \$84H
	0086	73	MOV	B, A
	0087	FAF9	BR	\$82H

This program executes an infinite loop between addresses 82H and 87H. The intervals (μ s) between the executions of the instruction at address 83H are measured.

2. Set event and trace conditions as follows:



3. After the event and trace conditions have been set, the execution of the program is traced as follows:



4. Save the trace results to a file. The execution times can be obtained by converting the time tag data to actual times by using, for example, spreadsheet software.

Conversion results

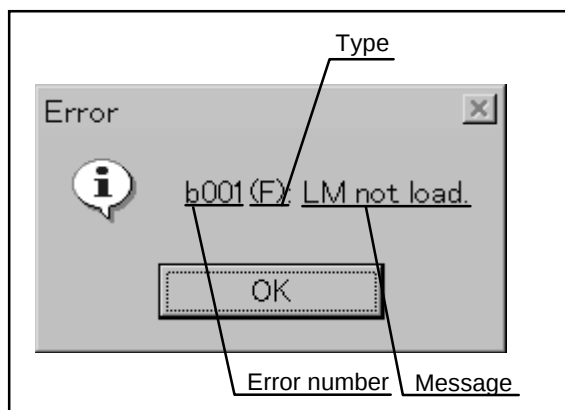
Frame number	Time tag data [hex]	Actual time [μ s]
32666	0x2671	984.1
32667	0x2756	1007.0
32668	0x283B	1029.9
32669	0x291F	1052.7
32670	0x2A14	1077.2
32671	0x2AEA	1098.6
32672	0x2BCE	1121.4
32673	0x2CB3	1144.3
32674	0x2D98	1167.2

Time tag data is counted every 100 ns. To obtain an actual time, convert hexadecimal to decimal, then convert the radix.

Appendix A Error Messages

This appendix lists the error and warning messages output by ID78K0.

An error message consists of [error number] + [type] + [message].



A type is represented by an alphabetic character. There are three types:

Type	Explanation
A	<u>A</u> bort error. Processing is interrupted and the debugger ends. If this error occurs, debugging cannot be continued.
F	<u>F</u> ormat (syntax) error. Processing is interrupted. The currently open windows and dialog boxes are closed.
W	<u>W</u> arning. Processing is interrupted. The currently open windows and dialog boxes remain as is.

A message contains the names of the file, variable, and device related to the error, as follows:

Representation in message	Explanation
xxx	Low-order three digits of device name
yyy	File name
zzz	Function name

Error messages (1/9)

Error No.	Type	Message	Explanation
---	--	Can't open this file. please make sure, now Active Window.	The project file format is incorrect, or the file content has collapsed. Loading the project file was discontinued.
---	--	Cannot find "character string".	The search character was not found. The search was discontinued. Alternatively, opening the specified file was discontinued because no data was in the file.
---	--	Event Name is not set.	There is no event name. Specify the name of the event when adding it.
---	--	Event number already exist.	It is impossible to add an event having the same number as an existing event. Change the number of the event to be added or of the existing event.
---	--	Not enough memory.	Because of insufficient memory, a window cannot be displayed, its content cannot be changed, or changes to it cannot be retained. Assign sufficient memory, and retry.
---	--	Other view mode window exist.	Two or more active windows of the same type cannot be opened simultaneously. An active window that was already open was closed.
---	--	Sorry, Too large view file.	The specified view file (.MEM, .TVW, or .DIS) contains more than 1000 lines. Its display was discontinued.
---	--	"event name" is already exist.	It is impossible to add an event having the same name as an existing event. Change the name of the event to be added or of the existing event.
0001	A	Communication open error	Communication with the in-circuit emulator (IE) is not possible.
0003	A	Hardware error	A hardware error is detected.
0004	A	Monitor time out	Data was not transferred to and from the monitor program. Clock pulses may not be being supplied to the target CPU or power may not be supplied. Check the above and restart the debugger.
0005	A	Not found monitor file	The monitor file is not found.
0006	A	Monitor file error	A monitor file error is detected.
0009	A	Communication failed	Communication with the IE failed.
000a	A	Verify error	A verify error is detected.
000e	A	User program Cannot run	The user program cannot be executed.
000f	A	Illegal receive data	An illegal response is received.
0012	A	Emulation-Board conflicts with Device-file	The EM board ID does not match the value in the device file.
0014	W	Target power off	The power of the target device is off.
0015	W	Program is running	The user program is running.
0016	W	Already break	The user program is already in the break status.
0017	W	Tracer is running	The tracer is running.

Error messages (2/9)

Error No.	Type	Message	Explanation
0018	W	Timer is running	Timer measurement is in progress.
001d	W	Measure is off	Timer measurement is not performed.
0020	W	Execution mode error	An execution mode error is detected.
0021	W	Mapping error	A mapping error is detected.
0022	W	Trace block not found	The specified trace block does not exist.
0023	W	There is no trace data	There is no trace data.
0024	W	Trace range over	The trace range has been exceeded.
0026	W	Bus hold mode	The bus hold mode is active.
0077	F	Search data not found	The search data does not exist.
0078	F	Measure overflow	The timer measurement result overflowed.
007a	F	Not specified coverage range	The coverage range has not been specified.
007e	W	Event No.3 is using	Event condition No. 3 is in use.
00c8	W	User program is stepping	The user program step is being executed.
01a1	A	Invalid EX78Kx.OM0	The executor file (EX78K0.OM0) was not read correctly. The executor file may not exist or may have been destroyed. Install the executor file again and restart the debugger.
01a3	A	Unconnected Emulation-board	The emulation board (IE-780xx-R-EM) is not correctly connected. Connect the IE-780xx-R-EM to the IE-78000-R-A correctly.
01a5	A	Unconnected I/O emulation-board	Emulation board 1 (IE-78xxx-R-EM1) is not correctly connected. Connect the IE-78xxx-R-EM1 to the IE-78000-R-A correctly.
01a6	A	Executor is running	The executor is running.
01a8	A	Invalid EXPC.INI	The initialize file (EXPC.INI) was not read correctly. The initialize file may not exist or may have been destroyed. Install the initialize file again and restart the debugger.
0600	A	Communication buffer error	The area for the buffer used for exchanging data with the IE was not reserved. End other MS-Windows applications, change the setting of the MS-Windows swap file, or install additional main memory in the host machine.
0f13	A	Send timed out	Data transmission to the IE failed. Possible causes include an invalid interface board setting and IE power off condition. Install the initialize file again, then restart the debugger.
0f14	A	Receive timed out	No response was received from the IE. The IE may be abnormal. Check the IE and restart the debugger.
0f15	A	Invalid D0xxx.78K	The device file (D0xxx.78K) cannot be read correctly. The device file may not be located in the specified directory or it may have been destroyed. Install the device file again, then restart the debugger.

Error messages (3/9)

Error No.	Type	Message	Explanation
1000	A	failure in initialization	An attempt to initialize the IE failed. Check whether the IE is abnormal.
1003	F	Illegal relocation address	It is impossible to relocate to a specified address.
1004	F	Illegal parameter	The parameter is illegal.
1006	F	Illegal address	The address is illegal.
1007	A	Not enough substitute memory	An attempt was made to map IE alternate memory in an area of 64K bytes or more.
100b	F	Program Is Running	A user program is running. This command cannot be executed.
100c	F	Different Bussize	An attempt was made to make duplicate specification in areas having different bus sizes.
100d	F	Total Maximum Over	An attempt was made to specify a bus larger than the maximum size (8).
100e	F	Enable Maximum Over	The bus size is larger than 8.
100f	W	Wrong Target Status(Power Off)	The target state is unstable.
10ff	A	Communication Error	It is impossible to communicate with the IE. Check whether the IE is abnormal.
2000	F	Illegal sfr name	The SFR name is illegal.
2002	F	User program is running	A user program is running. This command cannot be executed.
2003	F	Illegal SFR number	An attempt was made to access a nonexistent SFR.
2004	F	Illegal bit number	There is no bit SFR at the specified bit position.
2005	W	Redraw sfr name	The SFR has been disabled from redrawing.
2006	F	This SFR is hidden SFR	This SFR is not open to general use. It is impossible to display or change data for the SFR.
2007	F	Can't Read/Write	An attempt was made to write to a write-protected SFR or read from a read-protected SFR.
2008	F	Too big number	The specified SFR was not found.
200a	F	Illegal Bit Pattern	An attempt was made to specify an illegal value for an SFR.
20ff	A	Communication Error	Communication with the IE is impossible. Check whether the IE is abnormal.
3000	F	Illegal address	The address is illegal.
3001	F	Different data	There is a memory content mismatch.
3002	F	Illegal source address	The specified source address range does not fall within the mapping range (during a memory search, comparison, or copy).
3003	F	Illegal destination address	The specified destination address range does not fall within the mapping range (during a memory search, comparison, or copy).
3004	F	Illegal address (source & destination)	The specified address range does not fall within the mapping range (during a memory search, comparison, or copy).
3005	F	Illegal parameter	The parameter is illegal.
3006	F	User program is running	A user program is running. This command cannot be executed.
3008	F	No Parameter	There is no parameter.

Error messages (4/9)

Error No.	Type	Message	Explanation
3009	F	Parameter Size Alignment Error	The parameter size is illegal. Change the parameter according to the memory access size.
300a	F	Memory Alignment Error	The address value is illegal. Change the address value according to the memory access size.
300b	F	Source Start Address Alignment Error	The source address is illegal. Change the source address according to the memory access size.
300c	F	Error, Destination Start Address Alignment Error	In the destination address range, a memory range with a conflicting access memory size was specified.
300d	F	End Address Alignment Error	The end address is illegal. Change the end address according to the memory access size.
300e	F	Different Access Size in This Area	In the address range, a memory range with a conflicting access memory size was specified.
300f	F	Different Access Size in Source Area	In the source address range, a memory range with a conflicting access memory size was specified.
3010	F	Different Access Size in Destination Area	In the destination address range, a memory range with a conflicting access memory size was specified.
3011	F	Different Access Size, Source & Destination	The access size conflicts between the source and destination address ranges.
30ff	A	Communication Error	Communication with the IE is impossible. Check whether the IE is abnormal.
4000	F	Number is referenced now	The specified event condition cannot be deleted.
4001	F	Illegal table number	The specified table number is illegal.
4002	F	Illegal start address	The start address is illegal.
4003	F	Illegal end address	The end address is illegal.
4004	F	Illegal status	The status is illegal.
4005	F	Illegal data	The data is illegal.
4006	F	Can't action number	An attempt was made to use an event number that was already in use.
4007	F	Can't empty number	An attempt was made to register more than 32,767 events of the same type.
4008	F	Table not found	The specified event has not been registered.
4009	F	Illegal data size	The data size is illegal.
400a	F	Illegal type mode	The mode is illegal.
400b	F	Illegal parameter	The parameter is illegal.
400c	F	Illegal type number	The type is illegal.
400d	F	Table overflow	An attempt was made to register more than 32,767 events of the same type.
400e	F	No entry event number	The specified event does not exist.
400f	F	Illegal Elink data	An event condition specified with a range condition or pass condition was used as an event link condition. Alternatively, only one event condition was specified.
4010	F	Function not found	The specified function was not found.
4011	A	No free memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
4012	F	Timer not enabled	The timer is disabled. Enable it if timer measurement must be made.

Error messages (5/9)

Error No.	Type	Message	Explanation
4013	W	Data access size mismatch at the bus size	The access size in an event condition does not match the bus size for mapping.
4014	F	Can't use software break	At present, no software break can be used. Specify that a software break be usable, using the Extended Option dialog box.
4015	F	Not point-address	It is impossible to use, as an address condition, an event condition specifying a range.
4016	F	Not renew event condition.	This event condition is being used for another event. It is impossible to change the address range condition or pass count condition.
4017	F	Specified odd-address by word-access.	The data value was not detected in word data beginning at an odd address. Do not include that data value in the setting.
5000	A	Illegal type number	The type is illegal.
5002	A	Illegal file name	The device file cannot be opened.
5003	A	Cannot file seek	An attempt to seek the file failed.
5004	A	Cannot file close	An attempt to close the file failed.
5005	A	Illegal device format	The format of the device file is illegal.
5006	A	Cannot device initialize	An attempt to initialize the IE failed.
5007	A	Illegal device information	There is no device information.
5008	F	Cannot open device file	The specified device file cannot be opened.
5009	F	Cannot open EX78KX.OM0 file	The EX78K0.OM0 cannot be opened.
500a	F	No match device file of version	The version of the device file is illegal.
500b	W	Device has no relocatable iram.	The currently selected device does not support relocation in internal RAM.
6001	F	Illegal entry symbol name	The symbol name is illegal.
6002	F	Illegal parameter	The parameter is illegal.
6003	F	Illegal entry function name	The function name is illegal.
6004	F	Out of Buffer flow	Function display in the Stack window is incomplete. The maximum allowable line size is 512 characters.
6005	F	Illegal expression	The expression is illegal.
7001	F	User program is running	A user program is running. This command cannot be executed.
7002	F	User program is stopped	A user program is at a break. This command cannot be executed.
7003	F	Trace function is active	The tracer is running. This command cannot be executed.
7004	F	Trace memory is OFF	The tracer is off.
7005	F	No Return Address, Can't Execute	The return address of the current function was not found. Step execution based on the Return command is not carried out.
7010	W	Warning, No Source Line Information	Instruction-level step execution was carried out because there was no source information.
7012	A	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.

Error messages (6/9)

Error No.	Type	Message	Explanation
70fe	A	Bus Hold Error	The bus is on hold. The user program cannot be executed.
70ff	A	Communication Error	Communication with the IE is impossible. Check whether the IE is abnormal.
7801	F	Step wait canceled	Step execution was discontinued. So, communication with the IE may become impossible.
7802	F	Step aborted	An illegal access break occurred during step execution. Check the user program.
7f00	F	Interrupted step	Step execution was forced to end.
7f02	F	Suspended step	Step execution was suspended.
7f03	A	Run/Step cancel failed. CPU resetted	An attempt to break the user program failed. The IE is unstable because the evaluation chip was reset. Make sure that the IE is normal, then restart it.
7f04	F	Illegal address	An attempt was made to execute in a non-mapped area.
8000	F	File not found	The file was not found.
8001	F	Illegal line number	The line number is illegal.
8002	F	Current data is not set	The current information has not been set.
8003	F	Illegal address	The address is illegal.
9002	F	Illegal set value	The specified value cannot be set in a register. Specify a value that can be set.
a001	F	Illegal expression	The expression is illegal.
a002	F	Start address bigger than end address	The start address is greater than the end address (start address > end address). Check the addresses.
a003	F	Source path not found	The specified source path information is illegal. Specify the correct source path information.
a004	F	Expression is too big	The size of the expression is greater than 127 characters.
a005	A	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
a006	F	Illegal argument	The argument is illegal.
a008	F	Source path not set	The source path has not been specified.
a009	F	File not found	The file was not found.
a00a	F	File not open	The file cannot be opened.
a00b	A	File not close	An attempt to close the file failed.
a00c	A	File not read	An attempt to read the file failed. It is likely that the file has collapsed.
a00d	F	Not source file of LM	The specified source file has not been registered for the load module file. A file not registered for the load module file cannot be displayed in the Source window.
a00e	F	Illegal line number	The line number is illegal.
a00f	F	Illegal variable	The variable does not exist.
a010	A	Communication failed	Communication with the IE is impossible. Check whether the IE is abnormal.

Error messages (7/9)

Error No.	Type	Message	Explanation
a011	F	Can't access register	The register cannot be accessed. Check the IE.
a012	F	Can't access memory	The specified memory (variable) cannot be accessed. Check the IE or map setting.
b000	F	Command line error	The parameter is illegal.
b001	F	Task type not found	The load module file does not contain program information.
b002	F	File not found	The file was not found.
b003	F	Function not found	The specified function was not found.
b004	F	Illegal magic number	The magic number for the load module file is illegal.
b005	F	Symbol not found	The symbol was not found.
b008	F	Illegal value	The expression is illegal.
b009	A	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
b00a	F	Illegal symbol entry	There is an illegal symbol in the load module file. It is likely that there is a bug related to the programming language.
b00b	F	Current type noting	There is no debug information. Load the load module file.
b00c	F	Current file noting	There is no current source file. Alternatively the source file cannot be opened because the load module file has not been loaded.
b012	F	Line number too large	The line number is illegal.
b015	A	Read error	An attempt to read the file failed. It is likely that the file has collapsed.
b016	A	Open error	The file cannot be opened.
b017	A	Write error	An attempt to write to the file failed.
b019	A	Seek error	An attempt to seek the file failed.
b01a	A	Close error	An attempt to close the file failed.
b01d	F	Address not found	There is no source line that corresponds to the current PC value.
b01e	F	No line information(not compile with -g)	There is no source line information in the load module file. Attach the debug option, and carry out recompilation, assembly, and linkage.
b01f	F	Cannot find member	No member was found in the specified structure.
b020	F	Cannot find value	The specified enumeration constant is illegal.
b021	F	Striped LM	There is no symbol information in the load module file.
b022	F	Null statement line	The line number is illegal.
b026	F	Max dimension array over	A four-dimensional or greater-scale array cannot be displayed.
b027	F	End of file	The file is not complete.
b029	F	Illegal address	The address is illegal.
b02a	A	Communication failed	Communication with the IE is impossible. Check whether the IE is abnormal.
b02b	F	No stack frame point	Stack tracing is impossible with the current PC value.

Error messages (8/9)

Error No.	Type	Message	Explanation
b02c	F	Max block overflow	The maximum number of blocks in one function is exceeded. The function cannot be displayed. (The maximum number of blocks per function is 256.)
b02d	F	Illegal argument	The argument is illegal.
c001	F	Cannot open file	The file cannot be opened.
c002	A	Cannot close file	An attempt to close the file failed.
c003	A	Cannot read file	An attempt to read the file failed. It is likely that the file has collapsed.
c004	A	Cannot seek file	An attempt to seek the file failed.
c005	F	Illegal file type	The format of the file is illegal. This file cannot be handled.
c006	F	Illegal magic number	The magic number for the load module file is illegal.
c007	F	This file is not load-module file	The specified file is not a load module file.
c008	F	Old coff version	The version of the load module file is illegal.
c009	A	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
c00a	F	Illegal address	The address is illegal.
c00b	F	LM not load	The load module file has not been loaded.
c00c	F	Illegal argument	This is an internal error.
c00d	F	User program is emulating	A user program is running. This command cannot be executed.
c00e	F	User program is tracing	The tracer is running. This command cannot be executed.
c010	A	Communication failed	Communication with the IE is impossible. Check whether the IE is abnormal.
c011	F	Illegal file format	The format of the load module file (LNK) is illegal.
c012	F	Check sum error	A checksum error occurred in reading the load module file. Check the load module file.
c013	F	Too big size	The address range for uploading has exceeded 1M byte.
c014	F	Cannot write file	An attempt to write to the file failed.
c100	F	Not support	The Tektronix format is not supported.
d001	F	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
e000	F	Illegal argument	This is an internal error.
e001	F	Illegal start address	The start address is illegal.
e002	F	Illegal end address	The end address is illegal.
e003	F	Size too long	The address value is illegal.
e004	F	Can't open file	The specified file cannot be opened.

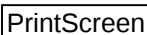
Error messages (9/9)

Error No.	Type	Message	Explanation
e005	F	Can't read file	An attempt to read the file failed. It is likely that the file has collapsed.
e006	F	Can't seek file	An attempt to seek the file failed.
e007	F	Can't write file	An attempt to write to the file failed.
e008	F	Not enough memory	There is no sufficient memory. End unnecessary applications, or close the Debugger window.
e009	F	Illegal file format	The format of the file is illegal.
XXXX	F	Internal error	An internal error occurred.

Appendix B Key Functions

Debugging can be carried out more effectively when ID78K0 is operated using the special function keys. In the following explanation of the special function keys, general key representations (generic key representations) are used. For the IBM-PC/AT Series, the key representations may differ slightly depending on the keyboard type.

B.1 Functions of Special Function Keys

Key		Function
PC-9801 and 9821 Series	IBM-PC/AT Series	
		Deletes the character immediately before the cursor and moves the cursor to the position of the deleted character. The character string following the cursor is moved back.
		Captures the entire screen into the clipboard as a bit image. (Windows function)
		<1> Closes the pulldown menu. <2> Closes the modal dialog box.
		Moves the cursor to the menu bar.
		Displays the last line. Also, the cursor is positioned to the last line.
		Displays the first line. Also, the cursor is positioned to the first line.
		Scrolls the display up by one screen. Also, the cursor is positioned to the top of the screen.
		Scrolls the display down by one screen. Also, the cursor is positioned to the top of the screen.
		Inserts one blank.
		Positions the cursor to the next item.
		Moves the cursor up. Scrolls the screen down by one line when the cursor is positioned to the top of the screen.
		Moves the cursor down. Scrolls the screen up by one line when the cursor is at the bottom of the screen.
		Moves the cursor to the left. Scrolls the screen to the right by one item when the cursor is in the leftmost column.
		Moves the cursor to the right. Scrolls the screen to the left by one item when the cursor is in the rightmost column.
		Confirms input data.

B.2 Functions of Special Function Keys (**CTRL**+ Key)

Key (Common to the PC-9801, 9821, and IBM-PC/AT Series)	Function
A	Using the data value in the current window as an address to jump to, disassembles and displays the program starting from that address. Opens the Assemble window.
B	Sets a breakpoint in a selected line.
C	Copies a selected character string to the clipboard buffer.
D	PC setting and window view: The Call dialog box is opened.
E	PC setting.
F	Switches a window to modify mode. This has the same effect as clicking the ToModify button.
G	Executes a program. This has the same effect as clicking the  button.
H	Switches a window to the Hold state.
I	Switches a window to the Active state.
M	Using the data value in the current window as an address to jump to, displays the contents of memory starting from that address. Opens the Memory window.
O	If the Source window is current: Allows the user to select a source view file. Opens the source file select dialog box. Otherwise: Displays an appropriate view file in the current window. Opens the view file save dialog box.
P	Stops the execution of a program. This has the same effect as clicking the  button.
R	Performs step execution until control returns to the calling function. This has the same effect as clicking the  button.
S	Saves the contents of the current window to a view file.
T	Performs step execution. This has the same effect as clicking the  button.
U	Using the data value in the current window as an address to jump to, displays an appropriate source text and source line. Opens the Source window.
V	Pastes the contents of the clipboard buffer to the text cursor position.
W	Switches a window to view mode. This has the same effect as clicking the ToView button.
X	Performs Next step execution. This has the same effect as clicking the  button.
Z	Cancels the previous editing operation.

Appendix C Menus

This Appendix lists the menus supported by ID78K0.

Symbols used in the menu lists

Symbol	Meaning
[Item]	Item on a menu bar
No symbol	Item in a pull-down menu
→ (arrow)	Item in a cascaded menu The number of arrows corresponds to the nesting level.

Table C-1 Main Window (1/4)

Menu	Mnemonic	Explanation
[File]		
<u>O</u> pen...	CTRL+O	Opens a file.
<u>S</u> ave	CTRL+S	Saves the contents of the current window into the view file.
Save <u>A</u> s...		Saves the contents of the current window into a view file having a different name.
<u>C</u> lose		Closes the current window.
<u>P</u> rint		Prints the contents of the current window.
<u>D</u> own load...		Downloads a program.
<u>U</u> p load...		Uploads a program.
Open/Save Project ▶		
→ <u>O</u> pen Project...		Opens a project file.
→ <u>S</u> ave		Overwrites the project file with the current debugging environment.
→Save <u>A</u> s...		Saves the current debugging environment into a project file.
Open/Save <u>L</u> og		Records the history of execution.
<u>E</u> xit		Exits from the debugger.
[Edit]		
<u>U</u> ndo	CTRL+Z	Cancels the most recent editing.
<u>C</u> opy	CTRL+C	Copies a selected character string into the clipboard buffer.
<u>P</u> aste	CTRL+V	Pastes the contents of the clipboard buffer at the point to which the text cursor is positioned.
<u>W</u> rite in		Writes the modified contents into the target device.
<u>R</u> estore		Cancels the modified contents.
<u>M</u> emory ▶		
→Memory <u>E</u> ill...		Initializes memory.
→Memory <u>C</u> opy...		Copies the contents of memory.
→Memory <u>C</u> ompare...		Compares the contents of memory.
→ <u>F</u> ile Compare...		Compares the view file with the contents of memory.

Table C-1 Main Window (2/4)

Menu	Mnemonic	Explanation
[View]		
<u>S</u> earch...		Searches for a character string or numerical value.
<u>A</u> ddress...		Displays the contents of memory at a specified address.
<u>V</u> iew Variable...		Displays the value of a specified variable temporarily.
<u>W</u> atch Variable...		Displays the value of a specified variable continuously.
Add Variable...		Adds a variable to the Variable window.
<u>S</u> ym To Adr...		Converts symbols.
D <u>e</u> lete		Deletes a specified value.
<u>B</u> in		Selects binary display format.
<u>O</u> ct		Selects octal display format.
<u>D</u> ec		Selects decimal display format.
<u>H</u> ex		Selects hexadecimal display format.
Prope <u>r</u>		Selects a default display format for each variable.
<u>E</u> vent ?		Displays event information.
<u>M</u> emory ▶		
→ <u>N</u> ibble		Displays data in nibble format.
→ <u>B</u> yte		Displays data in byte format.
→ <u>W</u> ord		Displays data in word format.
→ <u>L</u> ong		Displays data in long format.
→ <u>A</u> scii		Switches on or off ASCII view mode.
<u>S</u> fr ▶		
→ <u>A</u> ddress Sort		Selects alphabetic display order or display in order of addresses.
→ <u>P</u> ick Up		Displays only modified SFRs.
→ <u>A</u> tttribute ▶		
→→ <u>S</u> how		Displays the attribute view area.
→→ <u>H</u> ide		Hides the attribute view area.
→ <u>C</u> ompulsion Read		Performs forced reading of a read-protected SFR.
→ <u>S</u> ynchronize		Writes the modified SFRs to the target device.
<u>T</u> race View ▶		
→ <u>T</u> race View...		Selects the trace view contents.
→ <u>S</u> nap View...		Selects the snapshot trace view contents.
→ <u>N</u> ormal Title		Displays the trace frame titles.
→ <u>S</u> nap Title		Displays the snapshot frame titles.
→ <u>A</u> ll Title		Displays all titles.
→ <u>O</u> pen Frame...		Specifies a view frame number.
→ <u>P</u> ick Up...		Selects a view frame.
<u>C</u> overage ▶		
→ 1 <u>B</u> yte		Displays data in 1-byte units.
→64 <u>B</u> yte		Displays data in 64-byte units.

Table C-1 Main Window (3/4)

Menu	Mnemonic	Explanation
[Option]		
<u>T</u> ool Bar		Displays or hides the tool bar.
<u>S</u> tatus Bar		Displays or hides the status bar.
<u>B</u> utton		Displays or hides the buttons in the window.
<u>S</u> ource Mode		Selects the source mode.
<u>I</u> nstruction Mode		Selects the instruction mode.
<u>C</u> onfiguration...		Sets the environment.
Source <u>P</u> ath...		Sets source path information.
<u>E</u> xtended Option...		Sets extended options.
<u>M</u> ask Option...		Sets mask options.
[Execute]		
<u>S</u> top	CTRL+P	Stops the execution of a program.
<u>G</u> o	CTRL+G	Executes a program.
<u>R</u> eturn	CTRL+R	Executes a program, step by step, until control is returned to the calling function.
<u>S</u> tep	CTRL+T	Executes a program step by step.
<u>N</u> ext	CTRL+X	Performs Next step execution of a program.
<u>G</u> o & <u>G</u> o		Repeatedly executes a program.
<u>G</u> o & <u>C</u> ome		Executes a program up to a specified address.
<u>S</u> lowmotion		Continues step-by-step execution.
<u>C</u> PU Reset & <u>G</u> o		Resets the CPU before starting execution.
<u>C</u> PU Reset...		Resets the CPU.
Set <u>B</u> P	CTRL+B	Sets a breakpoint.
Set <u>P</u> C	CTRL+E	Sets the address in the program counter.
<u>C</u> all...	CTRL+D	Sets PC in the specified address and moves.
<u>E</u> xtSenceClip...		Sets external sense clip mode.
<u>T</u> race ▶		
<u>C</u> ond. Trace		Sets conditional tracing mode.
<u>M</u> achine All. Trace		Sets machine cycle, all-tracing mode.
<u>E</u> vent All. Trace		Sets event cycle, all-tracing mode.
Trace <u>F</u> ull Break		Breaks after full tracing.
[Operation]		
<u>A</u> ctive	CTRL+I	Puts the window in the active state.
<u>H</u> old	CTRL+H	Puts the window in the hold state.
<u>T</u> o <u>M</u> odify	CTRL+F	Puts the window in modify mode.
<u>T</u> o <u>V</u> iew	CTRL+W	Puts the window in view mode.
<u>W</u> indow Connect ▶		
→ <u>S</u> ourceText		Links to the Source window.
→ <u>A</u> ssemble		Links to the Assemble window.
→ <u>M</u> emory		Links to the Memory window.

Table C-1 Main Window (4/4)

Menu	Mnemonic	Explanation
[Browse]		
<u>S</u> ourceText...		Opens the Source window.
<u>V</u> ariable...		Opens the Variable window.
<u>A</u> ssemble...		Opens the Assemble window.
<u>M</u> emory...		Opens the Memory window.
<u>R</u> egister...		Opens the Register window.
Stack <u>T</u> race...		Opens the Stack window.
S <u>f</u> r...		Opens the SFR window.
<u>L</u> ocal Variable...		Opens the Local Variable window.
<u>B</u> reakSet...		Opens the Break dialog box.
<u>T</u> imer...		Opens the Timer window.
<u>T</u> race ▶		
→ <u>T</u> raceSet...		Opens the Trace dialog box.
→Trace <u>V</u> iew...		Opens the Trace View dialog box.
→ <u>S</u> napShotTraceSet...		Opens the Snap-Shot dialog box.
<u>E</u> vent ▶		
→ <u>E</u> ventSet...		Opens the Event Set dialog box.
→Event <u>M</u> anager...		Opens the Event Manager.
→Event <u>L</u> inkSet...		Opens the Event Link dialog box.
<u>C</u> overage ▶		
→ <u>V</u> iew...		Opens the Coverage window.
→C <u>l</u> ear...		Opens the Coverage Memory Clear dialog box.
→C <u>o</u> ndition...		Opens the Coverage Condition Setting dialog box.
→ <u>E</u> fficiency...		Opens the Coverage Efficiency View dialog box.
[Jump]		
<u>S</u> ourceText...	CTRL+U	Jumps to the Source window.
<u>A</u> ssemble...	CTRL+A	Jumps to the Assemble window.
<u>M</u> emory...	CTRL+M	Jumps to the Memory window.
[Window]		
<u>C</u> ascade		Displays the window in cascade style.
<u>T</u> ile		Displays the window in tile style.
Arrange <u>I</u> cons		Re-arranges the icons.
Close <u>A</u> ll		Closes all windows except the main window.
[Help]		
<u>A</u> bout...		Displays the information about the version.

Table C-2 Event Manager

Menu	Mnemonic	Explanation
[File]		
<u>O</u> pen...		Opens an event setting file.
<u>S</u> ave		Saves the current event settings into the event setting file, overwriting the previously saved setting.
Save <u>A</u> s...		Saves the current event settings into a specified event setting file.
<u>P</u> rint		Prints the event registration/setting information.
<u>C</u> lose		Closes the Event Manager.
[Edit]		
<u>U</u> ndo		Cancels the most recent editing.
<u>C</u> opy		Copies a specified icon using a different name.
<u>A</u> ll Select		Selects all icons.
<u>D</u> elete		Deletes a specified icon.
[View]		
<u>N</u> ame		Sorts the icons into event name order.
<u>K</u> ind		Sorts the icons into event type order.
<u>D</u> etail		Switches between normal view and detail view.
[Execute]		
Set <u>B</u> reak		Enables a break condition.
Cancel <u>B</u> reak		Disables a break condition.
Set <u>T</u> race		Enables a trace condition.
Cancel <u>T</u> race		Disables a trace condition.
Set <u>S</u> napShotTrace		Enables a snapshot condition.
Cancel <u>S</u> napShotTrace		Disables a snapshot condition.
[Operation]		
<u>B</u> reakSet...		Opens the Break dialog box.
<u>T</u> raceSet...		Opens the Trace dialog box.
<u>S</u> napShotTraceSet...		Opens the Snap-Shot dialog box.
<u>E</u> ventSet...		Opens the Event Set dialog box.
Event <u>L</u> inkSet...		Opens the Event Link dialog box.
[Jump]		
<u>S</u> ourceText...		Jumps to the Source window.
<u>A</u> ssemble...		Jumps to the Assemble window.
<u>M</u> emory...		Jumps to the Memory window.

Table C-3 Register Window

Menu	Mnemonic	Explanation
[File]		
Open/save Condition ►		
→ <u>O</u> pen Condition...		Opens the selected file for reference.
→ <u>S</u> ave Condition		Saves the contents of the window into a view file.
→ <u>S</u> ave File as...		Saves the current event settings into a specified view file.
<u>C</u> lose		Closes the Register window.
[Edit]		
<u>U</u> ndo		Cancels the most recent editing.
<u>C</u> opy		Copies a selected character string into the clipboard buffer.
<u>P</u> aste		Pastes the contents of the clipboard buffer at the point to which the text cursor is positioned.
<u>W</u> rite in		Writes the modified contents into the target device.
<u>R</u> estore		Cancels the modified contents.
[View]		
<u>A</u> bsolute Name		Displays absolute register names.
<u>F</u> unctional Name		Displays functional register names.
<u>R</u> egister		Displays registers individually.
Register <u>P</u> air		Displays register pairs.
<u>B</u> in		Displays data in binary format.
<u>O</u> ct		Displays data in octal format.
<u>D</u> ec		Displays data in decimal format.
<u>H</u> ex		Displays data in hexadecimal format.
[Operation]		
<u>A</u> ctive		Puts the Register window in the active state.
<u>H</u> old		Puts the Register window in the hold state.
To <u>M</u> odify		Puts the Register window in modify mode.
To <u>V</u> iew		Puts the Register window in view mode.
[Jump]		
<u>S</u> ourceText...		Jumps to the Source window.
<u>A</u> ssemble...		Jumps to the Assemble window.
<u>M</u> emory...		Jumps to the Memory window.

Table C-4 Variable Window

Menu	Mnemonic	Explanation
[File]		
Open/save Condition ►		
→ <u>O</u> pen Condition...		Opens the selected file for reference.
→ <u>S</u> ave Condition		Saves the contents of the window into a view file.
→ <u>S</u> ave File as...		Saves the contents of the window into a specified view file.
<u>C</u> lose		Closes the Variable window.
[Edit]		
<u>U</u> ndo		Cancels the most recent editing.
<u>C</u> opy		Copies a selected character string into the clipboard buffer.
<u>P</u> aste		Pastes the contents of the clipboard buffer at the point to which the text cursor is positioned.
<u>W</u> rite in		Writes the modified contents into the target device.
<u>R</u> estore		Cancels the modified contents.
[View]		
<u>B</u> in		Displays variable values in binary format.
<u>O</u> ct		Displays variable values in octal format.
<u>D</u> ec		Displays variable values in decimal format.
<u>H</u> ex		Displays variable values in hexadecimal format.
<u>P</u> roper		Displays variable values in default format for each variable.
[Operation]		
<u>A</u> ctive		Puts the Variable window in the active state.
<u>H</u> old		Puts the Variable window in the hold state.
<u>T</u> o <u>M</u> odify		Puts the Variable window in modify mode.
<u>T</u> o <u>V</u> iew		Puts the Variable window in view mode.
<u>D</u> elete		Removes a specified variable from the Variable window.

Table C-5 Timer Window

Menu	Mnemonic	Explanation
[File]		
Open/save Condition ►		
→ <u>O</u> pen Condition...		Opens a file.
→ <u>S</u> ave Condition		Saves the contents of the window into the original file.
→ <u>S</u> ave File as...		Saves the contents of the window into a specified file.
<u>C</u> lose		Closes the Timer window.
[Operation]		
<u>A</u> ctive		Places the Timer window in the active state.
<u>H</u> old		Places the Timer window in the hold state.

Facsimile Message

From:

Name

Company

Tel.

FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America

NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Hong Kong, Philippines, Oceania

NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Asian Nations except Philippines

NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Europe

NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

Korea

NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Japan

NEC Semiconductor Technical Hotline
Fax: 044-548-7900

South America

NEC do Brasil S.A.
Fax: +55-11-6465-6829

Taiwan

NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐
Organization	☐	☐	☐	☐