

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

User's Manual



ID78K4-NS

Integrated Debugger

Reference (Windows™ Based)

Target Device 78K/IV Series

Document No. U12796EJ1V0UM00 (1st edition)
Date Published September 1997 N

© NEC Corporation 1997
Printed in Japan

[MEMO]

IBM-PC/AT is a trademark of IBM Corporation in the USA.

i80386 and i80486 are trademarks of Intel Corporation in the USA.

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

The official name of Windows is the Microsoft Windows™ Operating System.

The export of this product from Japan is prohibited without governmental license. To export or re-export this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

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: 800-366-9782
Fax: 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: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.

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

NEC do Brasil S.A.

Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

INTRODUCTION

Thank you for purchasing the ID78K4-NS Integrated Debugger.

Conventional debuggers debugged in command units. In contrast, ID78K4-NS runs on Windows and provides an easy-to-understand and easy-to-use graphical user interface (GUI). Since the debugger is operated primarily by using a mouse, it can be operated without referring to the manual. In addition, frequently used commands are assigned to button groups, so they can be started with just a mouse click.

Purpose

The purpose of this manual is to foster in-depth understanding of all of the integrated debugger's functions.

The target users should be familiar with the operations of in-circuit emulators and Windows.

Files Shipped with the Integrated Debugger

Integrated Debugger Files

File Name	File Description
ID78K4A.EXE	This is the debugger. This file is run to start the debugger.
DB78K4A.DLL	Stores the libraries for file processing and symbol processing.
AS78K4A.DLL	Stores the libraries for assembly and disassembly.
EX78K4A.DLL	Stores the libraries for communication processing with the in-circuit emulator.
EXSETUPA.DLL	Program for setting EXPC.INI.
EXPC.INI	Init file. This file specifies the settings and the interrupt address of the PC interface board.

Target Device

The device targeted for debugging by the integrated debugger is called the target device.

The target device is the NEC 16-bit Single-Chip Microcontroller in the 78K/IV Series.

The target device depends on the IE-78K4-NS in-circuit emulator.

In-Circuit Emulator

To use the integrated debugger, the IE-78K4-NS in-circuit emulator and a dedicated interface board are required.

The table below lists the interface boards for the host machines.

Interface Boards

Product	Description
IE-70000-98-IF-C	PC-9801 and 9821 series interface board
IE-70000-PC-IF-C	IBM-PC/AT series interface board
IF-70000-CD-IF	PCMCIA card for NOTE

Host Machines

The integrated debugger runs on Windows.

The table lists the requirements of the host machines.

Item	Requirement
Host machine	PC-9801, 9821 series or IBM-PC/AT series
CPU	i80386 or later (33-MHz i80486 or better is recommended.)
Main memory	4 Mbytes or more (8 Mbytes or more is recommended.)
OS	Windows 3.1, Windows 95, Windows NT Version 4.0
Screen size	640 x 400 dots or more (800 x 600 dots or more is recommended.)

Organization

Chapter 1 Overview

This chapter describes the input conventions related to the character set and file names that can be used in the integrated debugger.

Chapter 2 Starting and Exiting the Debugger

This chapter describes how to install, start, and exit the integrated debugger.

Chapter 3 Terminology Description

This chapter describes the terms required to explain how to use the integrated debugger.

Chapter 4 Debugging Window Functions

This chapter describes the basic operation of a debugging window.

Chapter 5 Debugging Windows

This chapter describes all of the debugging windows in the integrated debugger.

Chapter 6 Debugger Function Overview

This chapter provides an in-depth description of each function in the integrated debugger.

Notation

The meanings of the symbols commonly used in this manual are described.

<div></div>	: Indicates pressing the key shown in the frame.
+	: Indicates simultaneously pressing the keys written on the left and right.
" "	: Indicates the string enclosed by these symbols.
' '	: Indicates the character enclosed by these symbols.
[]	: Indicates optional parameters.
<div>GRPH</div> key	: This is the key expression in the PC-9801 and 9821 series. In the IBM-PC/AT series, this becomes the <div>Alt</div> key.

This document uses the following expressions.

Expression	Contents
Key	PC-9801 and 9821 Series
Menu	English

If the host machine is the IBM-PC/AT series, refer to Appendix B, "List of Key Functions."

Caution

- In source debugging, add the options to generate the debugging data when compiling, assembling, and linking. If arguments are not included, source debugging is no longer possible.
- If a C language startup routine is independently written, add the symbols listed below. If the symbols are not included, a portion of the step execution will not be correct.

Location for Addition	Added Symbol
Beginning of startup routine	_ <code>@cstart</code>
End of startup routine	_ <code>@cend</code>

Related Documents

The documents (user's manuals) related to this manual are presented.

Document Name
RA78K Series Assembler Package, Language
RA78K Series Assembler Package, Operation
RA78K Series Structured Assembler Preprocessor
CC78K Series C Compiler, Language
CC78K Series C Compiler, Operation
78K/IV Series, Instruction
μ PD784026 Series, Hardware

[MEMO]

CONTENTS

CHAPTER 1 OVERVIEW	15
1.1 Debugger Overview	15
1.2 Function Overview	15
1.3 Integrated Debugger Input Conventions	16
CHAPTER 2 STARTING AND EXITING THE DEBUGGER	23
2.1 Installation	23
2.1.1 Equipment connections	23
2.1.2 Installing the debugger	24
2.2 Starting and Exiting the Debugger	33
2.2.1 Starting	33
2.2.2 Exiting	35
CHAPTER 3 TERMINOLOGY DESCRIPTION	37
3.1 Debugging Modes	38
3.2 Files	38
3.3 Current File	38
3.4 Functions	39
3.5 Current Function	39
3.6 Structures	39
3.7 Stack Frame Number	39
3.8 Lines	40
3.9 Real-time RAM Sampling	40
CHAPTER 4 DEBUGGING WINDOW FUNCTIONS	41
4.1 Basic Window Operations	41
4.2 The Active State and the Hold State	43
4.3 View Mode and Modify Mode	44
4.4 Errors and Warnings	44
4.4.1 Errors and warnings during GUI operation	44
4.4.2 Errors and warnings output by the debugger	44
CHAPTER 5 DEBUGGING WINDOWS	45
5.1 Window Type and Layout	45
5.1.1 Windows	45
5.1.2 Dialogs	48
5.2 List of Debugging Windows	50
5.3 Descriptions of the Debugging Windows	52
CHAPTER 6 DEBUGGER FUNCTION OVERVIEW	203
6.1 System Operating Modes	203
6.1.1 Types of operating modes	203
6.1.2 System Operating Modes	203
6.1.3 System operating states	204
6.2 sing the Basic Functions	204
6.2.1 lock selection function	204

6.2.2	apping functions	205
6.2.3	Reset function.....	205
6.2.4	Load function	206
6.2.5	Emulation execution function.....	208
6.2.6	Break function.....	213
6.2.7	Trace functions	215
6.2.8	Event setting and detection function.....	220
6.2.9	Register manipulation functions	224
6.2.10	Memory manipulation functions	224
6.2.11	Save function.....	224
6.2.12	Time measurement function	224
6.2.13	Source debugging	225
APPENDIX A ERROR MESSAGES.....		227
APPENDIX B LIST OF KEY FUNCTIONS.....		237
B.1	Special Function Key Function List	237
B.2	Special Function Key Function List (CTRL + Key).....	238
APPENDIX C INDEX.....		239

LIST OF FIGURES (1/2)

Figure No.	Title	Page
2-1.	Configuration Dialog at Startup	34
2-2.	Startup Screen of the Debugger.....	35
5-1.	Main Window	53
5-2.	Configuration Dialog	68
5-3.	Extended Option Setting Dialog	73
5-4.	Project File Load Dialog	76
5-5.	Project File Save Dialog	79
5-6.	Load Module Selection Dialog.....	82
5-7.	Upload Dialog	85
5-8.	Source Path Specification Dialog	88
5-9.	Source File Selection Dialog	90
5-10.	Source Text Window	94
5-11.	Search Dialog	98
5-12.	Change Symbol Dialog.....	101
5-13.	Variable View Dialog	103
5-14.	Variable Window.....	105
5-15.	Add Variable Dialog.....	110
5-16.	Local Variable Window	113
5-17.	Address Specification Dialog.....	117
5-18.	Disassemble Window	120
5-19.	Memory Window.....	126
5-20.	Memory Fill Dialog.....	129
5-21.	Memory Copy Dialog	131
5-22.	Memory Compare Dialog.....	133
5-23.	Memory Comparison Result Dialog.....	135
5-24.	Stack Trace Window.....	137
5-25.	Event Dialog	140
5-26.	Event Manager	147
5-27.	Event Setting Example	153
5-28.	Event-related Images	154
5-29.	Event Link Dialog	155
5-30.	Setting Event Link Conditions	158
5-31.	Example Using Event Link Conditions.....	160
5-32.	Break Dialog	161
5-33.	Setting Break Event Conditions.....	164
5-34.	Trace Dialog	165
5-35.	Setting Trace Event Conditions.....	169
5-36.	Timer Dialog	170
5-37.	Trace View Window.....	172
5-38.	Trace Pick Up Dialog.....	177
5-39.	Register Window	181
5-40.	SFR Window.....	187

LIST OF FIGURES (2/2)

Figure No.	Title	Page
5-42.	View File Save Dialog.....	192
5-43.	Error/Warning Dialog	196
5-44.	Reset Confirmation Dialog.....	197
5-45.	Version Display Dialog.....	199
5-46.	Exit Confirmation Dialog	200
6-1.	Example of the System Operating States.....	204
6-2.	System Operation State (Go)	209
6-3.	Conceptual Diagram of Return Command Execution	209
6-4.	Example of the System Operating State (Return).....	210
6-5.	Example of the System Operating State (Go & Go)	210
6-6.	System Operating State (Come).....	211
6-7.	Example of the System Operation State (CPU Reset & Go)	211
6-8.	Example of the System Operation State (Step).....	212
6-9.	Conceptual Diagram of Next Step Execution	213
6-10.	Trace Memory Concept	216
6-11.	Concepts Behind Event Detection	223

LIST OF TABLES

Table No.	Title	Page
2-1.	Debugger File List	24
5-1.	List of Debugging Windows	50
6-1.	Description of the Trace Data Display	218
6-2.	Trace Search Items	219

[MEMO]

CHAPTER 1 OVERVIEW

1.1 Debugger Overview

The ID78K4-NS (later referred to as ID or the debugger) is operated by using a dedicated interface board to connect the host machine (host machine with the Windows operating system on a PC-9801, 9821 or an IBM-PC/AT) and the IE-78K4-NS in-circuit emulator.

1.2 Function Overview

This section describes the functions and features of ID.

(1) GUI function

Debugging can take place in the Windows environment and operated by the mouse.

Buttons and menus are arranged on each window. Related information can be easily viewed based on the displayed information.

(2) Source level debugging function

Operations such as referencing and setting variables and structures, displaying programs, and setting breakpoints can be efficiently performed at the source text level of function names and line numbers.

(3) Instruction level debugging function

Operations such as referencing and setting symbols and register values, displaying programs, and setting breakpoints can be efficiently performed at the instruction level of labels and addresses.

(4) Using the functions of the in-circuit emulator

The detailed event setting functions of the in-circuit emulator can be used to set breaks and to trace programs.

(5) Watch function (automatic display update function when the execution pauses)

When the execution of a user program pauses, the values in the displayed windows (display window, display/setting window) are automatically updated.

(6) Saving and restoring the debugging environment

The debugging state is saved, and the saved conditions are restored.

(7) Displaying the source text in a function

The source text in a function is displayed by selecting the function from a list of functions.

1.3 Integrated Debugger Input Conventions

Character Set

This character set can be used in the integrated debugger.

• English letters	Uppercase letters	A	B	C	D	E	F	G	H	I	J	K	L	M
		N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	Lowercase letters	a	b	c	d	e	f	g	h	i	j	k	l	m
		n	o	p	q	r	s	t	u	v	w	x	y	z
• Numbers		0	1	2	3	4	5	6	7	8	9			
• Equivalent English characters		@	?	_										
• Special characters		.	,	:	;	*	/	+	-	'	<	>	()
		\$	=	!	#	[]							

Character	Name	Main Use
.	Period	Bit position specifier
,	Comma	Delimiter between operands
:	Colon	Label delimiter
;	Semicolon	Comment start symbol
*	Asterisk	Multiplication operator
/	Slash	Division operator
+	Plus	Addition operator
-	Minus	Negative sign or subtraction operator
'	Quote	Character constant, character string start and end symbol
<	Inequality symbol	Comparison operator
>	Inequality symbol	Comparison operator
(Left parenthesis	Change in the operator precedence
)	Right parenthesis	Change in the operator precedence
\$	Dollar sign	Start symbol for relative addressing
=	Equal sign	Comparison operator
!	Exclamation mark	Start symbol for absolute addressing
#	Sharp	Symbol denoting an immediate value
[Left bracket	Indirect display symbol
]	Right bracket	Indirect display symbol
↵	Carriage return	Only one (↵) is allowed before a linefeed LF (0DH).

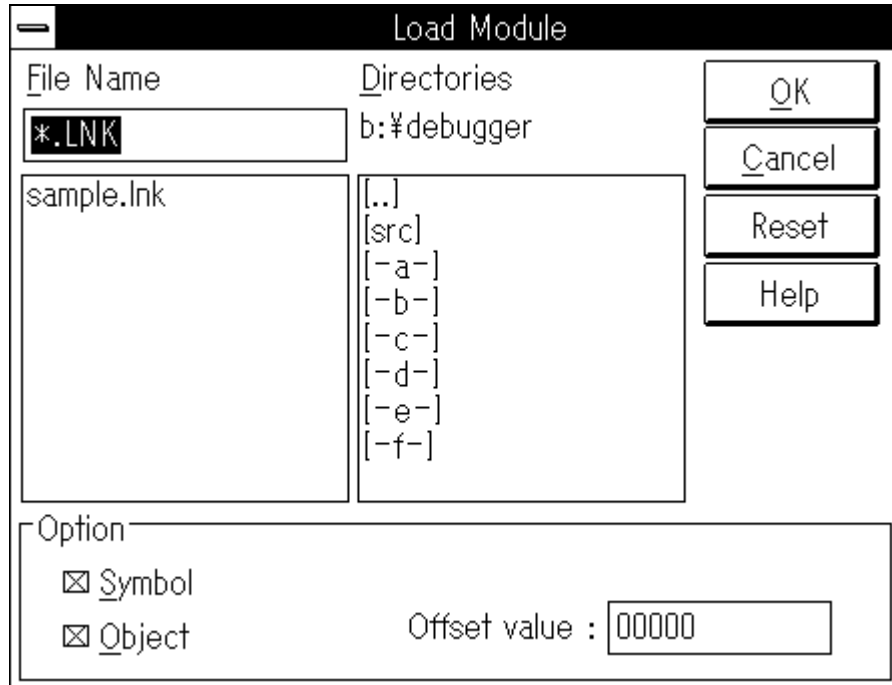
Specifying a File

A file is specified in the following format.

File Name: primary-name[.file-type]

Directories: [drive-name:][\directory-name]...

primary-name : Character string up to 8 characters
 file-type : Character string up to 3 characters
 drive-name : Only one character
 directory-name : Same format as the file name



Wild Cards

- The * and ? in a path name or file name can be used as wild cards.
- * denotes any character string.
- ? denotes any one character. (White space is also one character.)
- If a wild card is specified, the corresponding directory names under the directory and all of the file names are displayed.
- If a file name is directly specified, an error occurs if a wild card is used.

Example : If the following eight files are saved in a directory, the file names corresponding to the wild cards are as follows.

AAAAA.HEX, ABC.C, ABC.HEX, ABC.SYM, ABCDEFGH.HEX, XYZ, BCDEFG.HEX, XYZ

Examples of Wild Card Specifications	Corresponding Files
A*.*	AAAAA.HEX, ABC.C, ABC.HEX, ABC.SYM, ABCDEFGH.HEX, XYZ
A*	XYZ
A*.HEX	AAAAA.HEX, ABC.HEX, ABCDEFGH.HEX
*.HEX	AAAAA.HEX, ABC.HEX, ABCDEFGH.HEX, BCDEFG.HEX
A???.HEX	ABC.HEX
A??.*	ABC.C ABC.HEX, ABC.SYM
???	XYZ
???.	XYZ
ABC.?	ABC.C
ABC.???	ABC.C ABC.HEX, ABC.SYM

Operands

The five types of operands are

- Numerical Value
- Address
- Register
- Symbol
- Expression and Operator

Numerical Values

The following four types of numerical values can be used.

- Binary number
Input format nY(*2)
n...nY(*2) (n = 0, 1)
- Octal number
Input format nO(*2)
n...nO(*2) (n = 0, 1, 2, 3, 4, 5, 6, 7)
- Decimal number
Input format n
n...n
nT(*2)
n...nT(2) (n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Hexadecimal number
Input format nH(*2)(*1)
n...nH(*2)(*1)
0xn(*2)
0xn...n(*2) (n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (*2))

Cautions 1. If the first character is A to F, a 0 must be added at the beginning.

Example: FFH → 0FFH

2. The suffixes (Y, O, T, H, 0x) and the hexadecimal letters may be uppercase or lowercase.

Addresses

- An address is specified by directly specifying a numerical value.
- A symbol or an expression can be used in the specification.
- If an address is specified by a numerical value, a hexadecimal, decimal, octal, or binary number can be used.

Registers

- A general-purpose register is specified by an absolute name or a function name.
- A name is assigned to each bit in the PSW.
- The registers have the following types.

Register Type	Register Name
Control Registers	PC
	SP
	PSW

Register Type	Register Name
PSW	UF
	RBS2
	RBS1
	RBS0
	S
	Z
	RSS
	AC
	IE
	P/V
	CY

Register Type	Register Name		
	Absolute Name	Function Name	
		When RSS = 0	When RSS = 1
General-purpose registers	R0	X	
	R1	A	
	R2	C	
	R3	B	
	R4		X
	R5		A
	R6		B
	R7		C
	R8		
	R9		
	R10		
	R11		
	R12	E	E
	R13	D	D
	R14	L	L
	R15	H	H
	RP0	AX	
	RP1	BC	
	RP2		AX
	RP3		BC
	RP4	VP	VP
	RP5	UP	UP
	RP6	DE	DE
	RP7	HL	HL
	RG4	VVP	VVP
	RG5	UUP	UUP
	RG6	TDE	TDE
	RG7	WHL	WHL

Symbols

- A symbol consists of any of the characters of A to Z, a to z, @, ?, _ (underline), and 0 to 9.
- The beginning of a symbol must be a character other than a number, 0 to 9.
- Uppercase letters (A to Z) are distinguished from lowercase letters (a to z).
- A symbol name has a maximum of 31 characters.
- If a symbol longer than 31 characters is defined, the first 31 characters are valid.
- A symbol is defined by loading a load module file.

- A symbol has following types in each valid range.
Public symbols (assembler, structured assembler, C)
Local symbols
 Local symbols in a module (assembler, structured assembler)
 Local symbols in a file (C)
 Local symbols in a function (C)
- The following are available in each language that is used.
Assembler and structured assembler
 Label names, constant names, bit symbol names
C
 Function names (including point variable names, enumeration variable names, array names, structure names, union names)
 Function names, label names
 Array elements, structure elements, union elements, bit fields (where the symbol is an array, structure, or union)
- If a C function name duplicates a register name, flag name, SFR name, or SFR bit name, an "_" must be added immediately before the symbol to explicitly distinguish it.
- A symbol can be described instead of an address or a numerical value.
- The valid range of a symbol is determined based on the source debugging data when assembled or compiled.
- A public symbol is only described by a symbol name.
- A local symbol is expressed for a file name or a module name.

Expressions and Operators

Expressions

- An expression uses operators to combine constants, register names, SFR names, and symbols.
- If an SFR name, label name, function name, or variable name is described as a symbol, the address is operated on as a symbol value.
- Elements other than the operators forming an expression are called terms (constant, label) and are the first term, second term,... from the left in the description.

Operators

The following types of operators are available.

List of Arithmetic Operators

Symbol	Meaning	Description
+	Addition	Returns the sum of the first and second terms.
–	Subtraction	Returns the difference between the first and second terms.
*	Multiplication	Returns the product of the first and second terms.
/	Division	Divides the first term by the second term and returns the integer part of the result.
MOD	Modulus	Divides the first term by the second term and returns the remainder of the result.
– sign	Unary operator (negative)	Returns the two's complement of the value of the term.
+ sign	Unary operator (positive)	Returns the two's complement of the value of the term.

List of Logical Operators

Symbol	Meaning	Description
NOT	Negation	Inverts each bit in the term and returns the value.
AND	Logical product	Returns the logical product of each bit in the first term and the second term.
OR	Logical sum	Returns the logical sum of each bit in the first term and the second term.
XOR	Logical exclusive or	Returns the exclusive or of each bit in the first term and the second term.

Other Cases

Symbol	Meaning	Description
(Left parenthesis	An operation enclosed by parentheses () has precedence over operations outside the parentheses.
)	Right parenthesis	

Cautions

- The left parenthesis and the right parenthesis are always used in pairs.
- A character string can be described in the term in a comparison operation.
- The operations have the following convention.

The order of the operations follows the precedence of the operators.

If operators have the same precedence, the operation is from left to right.

An operation enclosed by parentheses () has precedence over the operations outside of the parentheses.

Each term in an operation is treated as unsigned 32-bit data.

All of the results of the operations are handled as unsigned 32-bit data.

If an overflow occurs during an operation, the low-order 32 bits are valid and the overflow is not detected.

- The operator precedence is as follows.

High ↑ (,.)
 – sign, NOT
 *, /, MOD
 +, –
 AND
 Low ↓ OR, XOR

Terms

When a constant is described in a term, the following numerical values can be described.

- For binary numbers
 $0Y \leq \text{numerical value} \leq 111111111111111111111111111111Y$ (32 digits)
- For octal numbers
 $0O \leq \text{numerical value} \leq 3777777777O$
- For decimal numbers
 $-2147483648 \leq \text{numerical value} \leq 4294967295$
 A negative decimal number is converted internally into a two's complement number.
- For hexadecimal numbers
 $0H \leq \text{numerical value} \leq 0FFFFFFFH$

CHAPTER 2 STARTING AND EXITING THE DEBUGGER

This chapter describes how to install, start, and exit ID.

2.1 Installation

2.1.1 Equipment connections

This section describes how to connect the equipment.

(1) Verifying the environment

The following environments are required to start the debugger ID.

	PC-9801, 9821 Series	IBM-PC/AT Series
Host machine	CPU: i80386 or later, Memory: 4 Mbytes or more (The 33-MHz i80486 CPU and at least 8 Mbytes of memory are recommended.)	
OS	Windows Ver. 3.1, Windows95, Windows NT 4.0 or later (Enhanced mode)	
Interface board	For expansion slot (C bus) IF-70000-98-IF-C	(ISA bus) IF-70000-PC-IF-C
	For INOTE IF-70000-CD-IF	
In-circuit emulator	IE-78K4-NS	

(2) Setting up the equipment (setting EXPC.INI)

The environment of the interface is set by the initialization file (EXPC.INI).

Make the appropriate settings for the interface board based on the following description.

[IFPCD]

HOST=1 ;PC98:0 IBM:1

Set "0" when using the PC98 and "1" when using an IBM-PC/AT compatible computer.

TIME=1

This setting does not have to be changed. (Not used)

PC_BASE=0x220 ;I/O base address (0x220)

The I/O address is set on the interface board.

PCIF.EXE makes the setting when the interface board is used.

The I/O address is set.

IE_INT=0x0D ;interrupt vector (0x0D)

This setting does not have to be changed. (Not used)

HW_TYPE=0 ;I/F → Bord (0) or Card(1)

Set "0" when using the interface board and "1" when using the interface card.

TIME_INT1=20 ;Interval Timer Setting

This setting does not have to be changed.

2.1.2 Installing the debugger

This section describes how to install the debugger (for both the PC-9801, 9821 and the IBM-PC/AT).

(1) Checking the environment and the files

The files used by the debugger are listed below. Device files are not included in the debugger and must be separately purchased.

Table 2-1. Debugger File List

File Name	File Description
ID78K4A, EXE	This is the debugger. Executing this file starts the debugger.
DB78K4A, DLL	Stores the libraries for file processing and symbol processing.
AS78K4A, DLL	Stores the libraries for the assemble and disassemble processes
EX78K4A, DLL	Stores the libraries for communication with the in-circuit emulator.

Initial Setting Files (Install directory : Windows directory)

File Name	File Description
EXPC.INI ^{Note}	This is an init file. This file specifies the settings and interrupt address of the PC interface board.
NECDEV.INI	This is an init file. This file stores the installation information for NEC tools.

Note If EXPC.INI is already in the Windows directory or in the installation directory for executable files, it is not installed.

Sample Files (Install directory : C:\NECTOOLS\SMP78K4\ID78K4A)

File Name	File Description
SAMPLE.C	This is a sample C program.
SUB.C	This is a sample C program. This file stores the subroutines for SAMPLE.C.
SAMPLE.LNK	This is the load module file for the sample programs (SAMPLE.C, SUB.C). It is compiled by μ PD784026.

Device Files

The following device file is required to start ID78K4-NS.

File Name	File Description
D4xxx.78k	This is a device file.

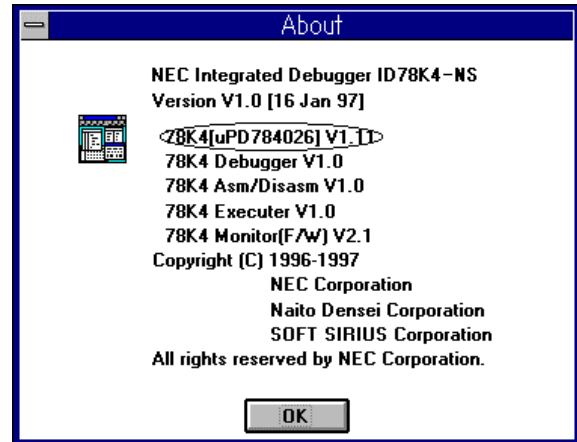
Note Use version V1.01 or later of the DF784026 device file. (DF784026 V1.00 is not compatible with the integrated debugger.)

There are several versions of the device file. When using the integrated debugger, always check the version.

(Help (H) → About (A)... in the menu bar)

The following is displayed by the integrated debugger.

D4xxx.78K → 78K4[μPD784xxx] Vx.xx




(2) Installing the debugger

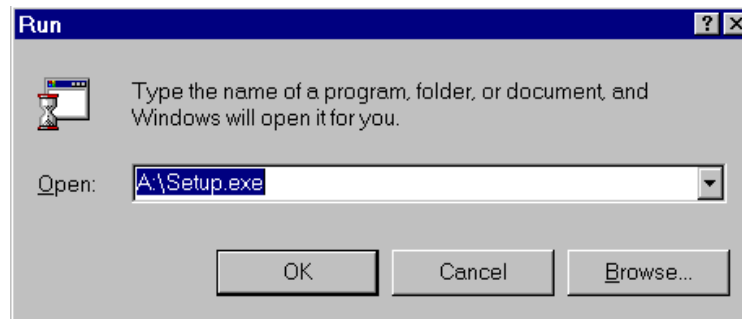
To use the debugger, the hardware must be set up and the debugger must be installed. This section describes how to install the debugger.

How to install the debugger will be explained. Here, the directory where Windows (Windows 3.1 or Windows95) is installed is described by "C:\Windows".

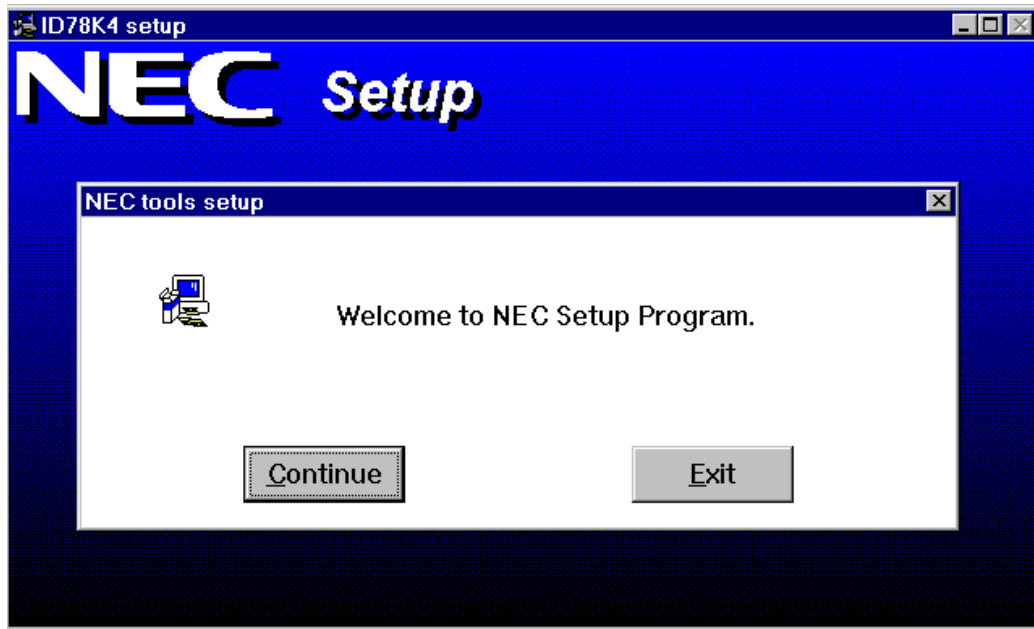
- (1) Turn on the voltage supply of the host machine (PC-9801, 9821 or IBM-PC/AT) and start Windows.
- (2) Insert the system disk of the debugger (the disk labeled Disk1 if there are multiple disks) into the floppy disk drive (F drive). Run "setup.exe" in the root directory.

Example: In Windows95

Select "Run...(R)" in the start menu. After entering the file name, press the  button.

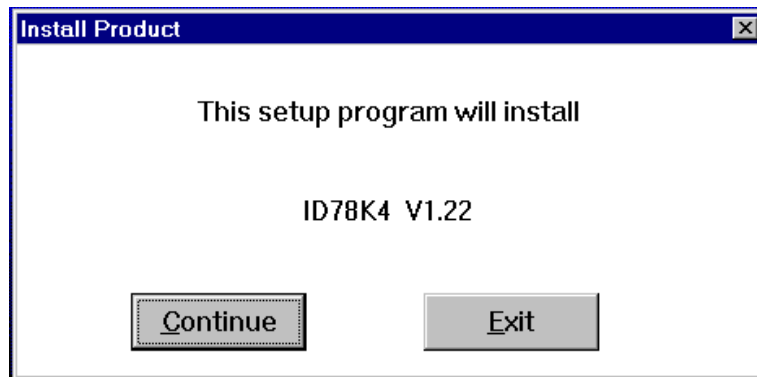


- (3) Start the setup program. The ID78K4-NS setup screen appears after initializing the setup. Select the **Continue(C)** button. Use the **Exit(E)** button to exit the installation.



- (4) Select the installation items

Since other NEC tools are not included in ID78K4-NS, press the **Continue(C)** button. Use the **Exit(E)** button to exit the installation.



(XX differs with the version.)

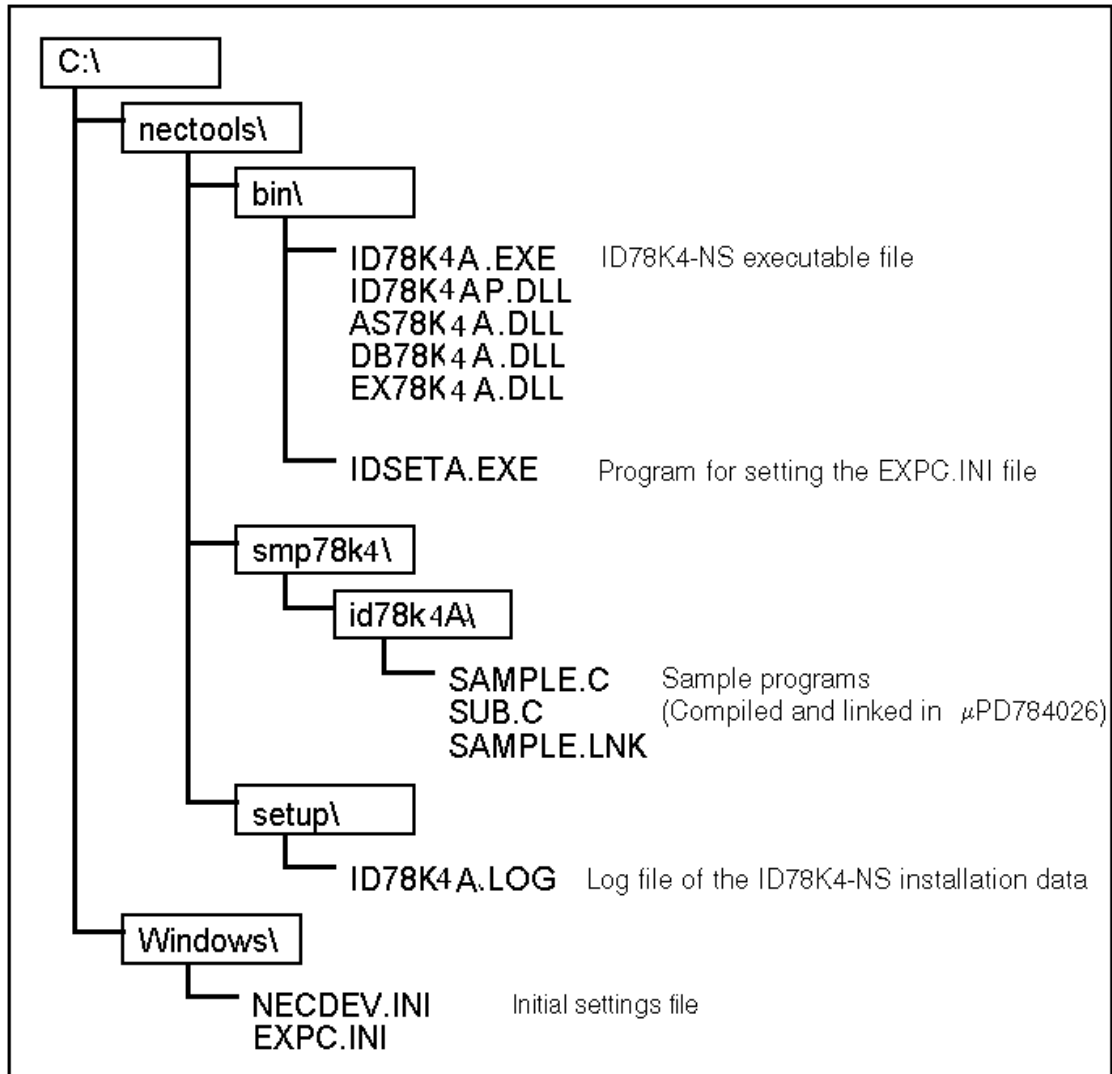
- (5) Select the drive and directory where the debugger will be installed.

Correct any problems with the drive and directory. After verifying, press **Continue(C)**. If there are any problems, press the **Back(B)** button to go back one step (to (4)) or press the **Exit(E)** button to exit the installation.

Caution If NEC tools will be installed, these settings will become valid. For other tools, it is recommended to install without changing the initial settings except for the drive name.

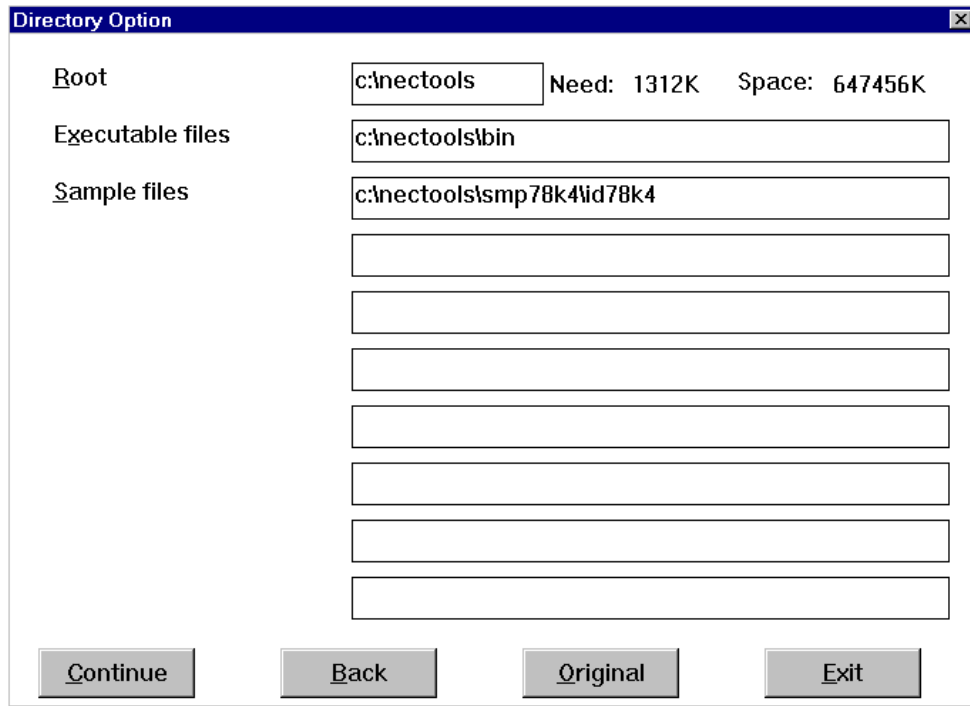
The defaults are given below.

1. For a new installation (if necdev.ini file is not in the Windows directory)
The installation is to the drive where Windows was installed.



2. When other NEC tools have already been installed

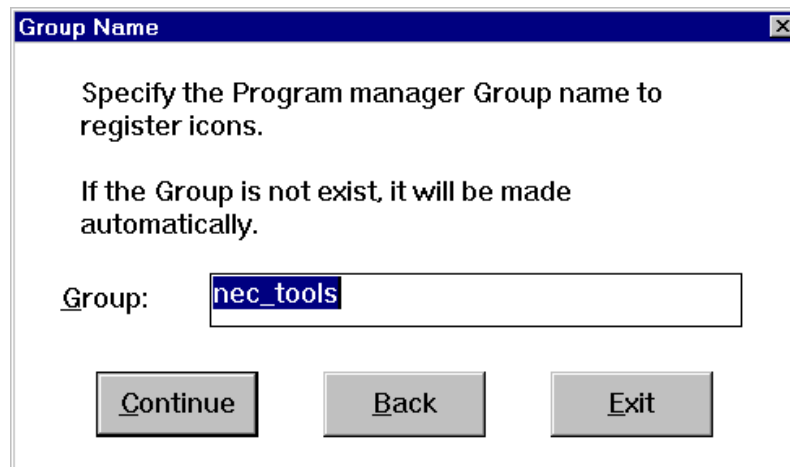
The directory in which NEC tools were previously installed is selected. (The display is in accordance with the necdev.ini data in the Windows directory.)



The 'Directory Option' dialog box has a title bar with a close button. It contains three labels on the left: 'Root', 'Executable files', and 'Sample files'. To the right of 'Root' is a text box containing 'c:\nec\tools' and a status label 'Need: 1312K Space: 647456K'. To the right of 'Executable files' is a text box containing 'c:\nec\tools\bin'. To the right of 'Sample files' is a text box containing 'c:\nec\tools\smp78k4\id78k4'. Below these text boxes are seven empty text boxes. At the bottom are four buttons: 'Continue', 'Back', 'Original', and 'Exit'.


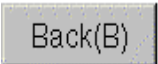
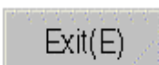
(6) Specify the group name registered to the icon.

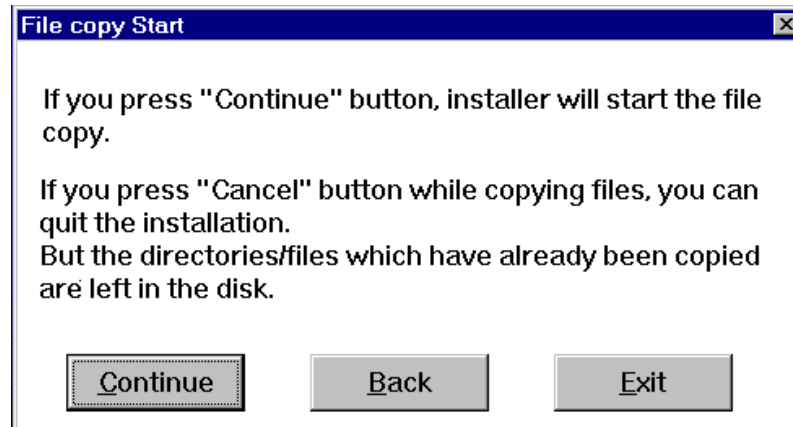
Specify the group name and then press the **Continue(C)** button. If there were any problems, press the **Back(B)** button to return to the previous step (to (5)) or press the **Exit(E)** button to exit the installation. The default group name is "NEC tools."



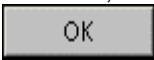
The 'Group Name' dialog box has a title bar with a close button. It contains the text 'Specify the Program manager Group name to register icons.' and 'If the Group is not exist, it will be made automatically.' Below this is a label 'Group:' followed by a text box containing 'nec_tools'. At the bottom are three buttons: 'Continue', 'Back', and 'Exit'.

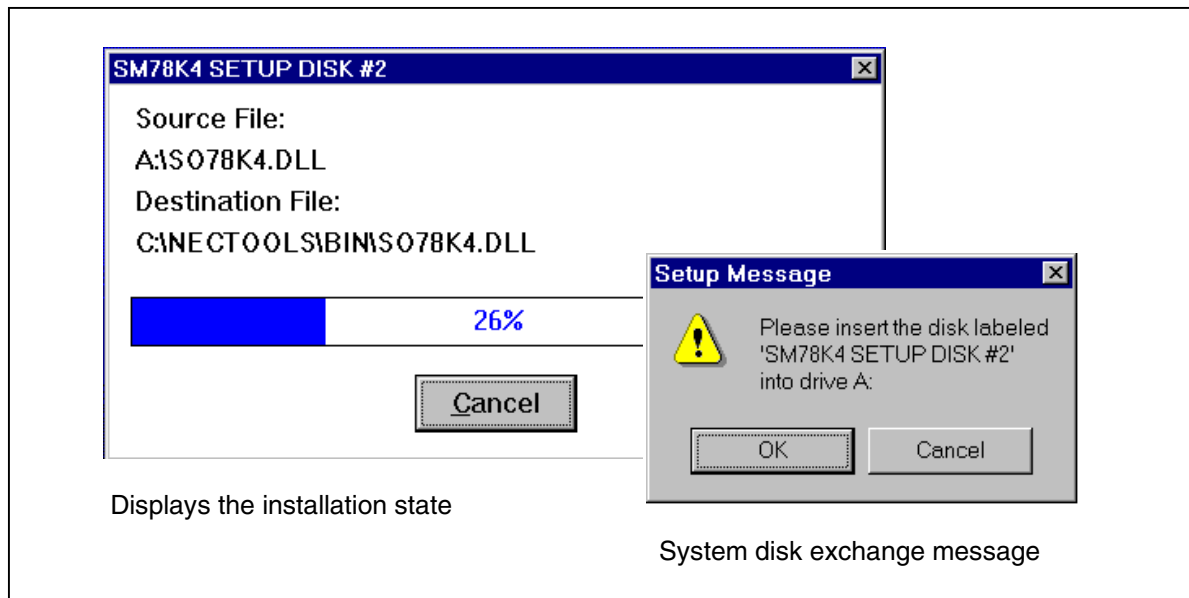
(7) Final confirmation when starting the installation

If the installation starts with the contents set in (3) to (6) and there are no problems, press the  button. If there are any problems, press the  button to return to the previous step (to (6)) or press the  button to exit the installation.



(8) File installation

The files are installed in the directory specified in (6). If there are two or more system disks, the exchange message dialog for the system disk will appear. After exchanging the disk, press the  button.

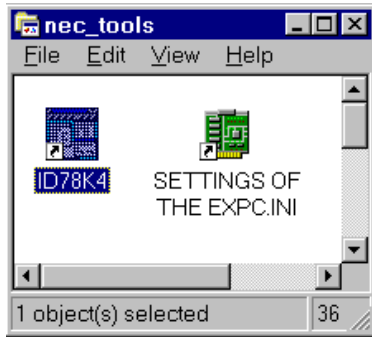


(9) Exiting the file installation

If the file installation ends without error, the group with the group name specified in (6) is registered in the following locations.

For Windows 3.1 : Registered in the Program Manager

For Windows95 : Registered in Program (P) of the start menu



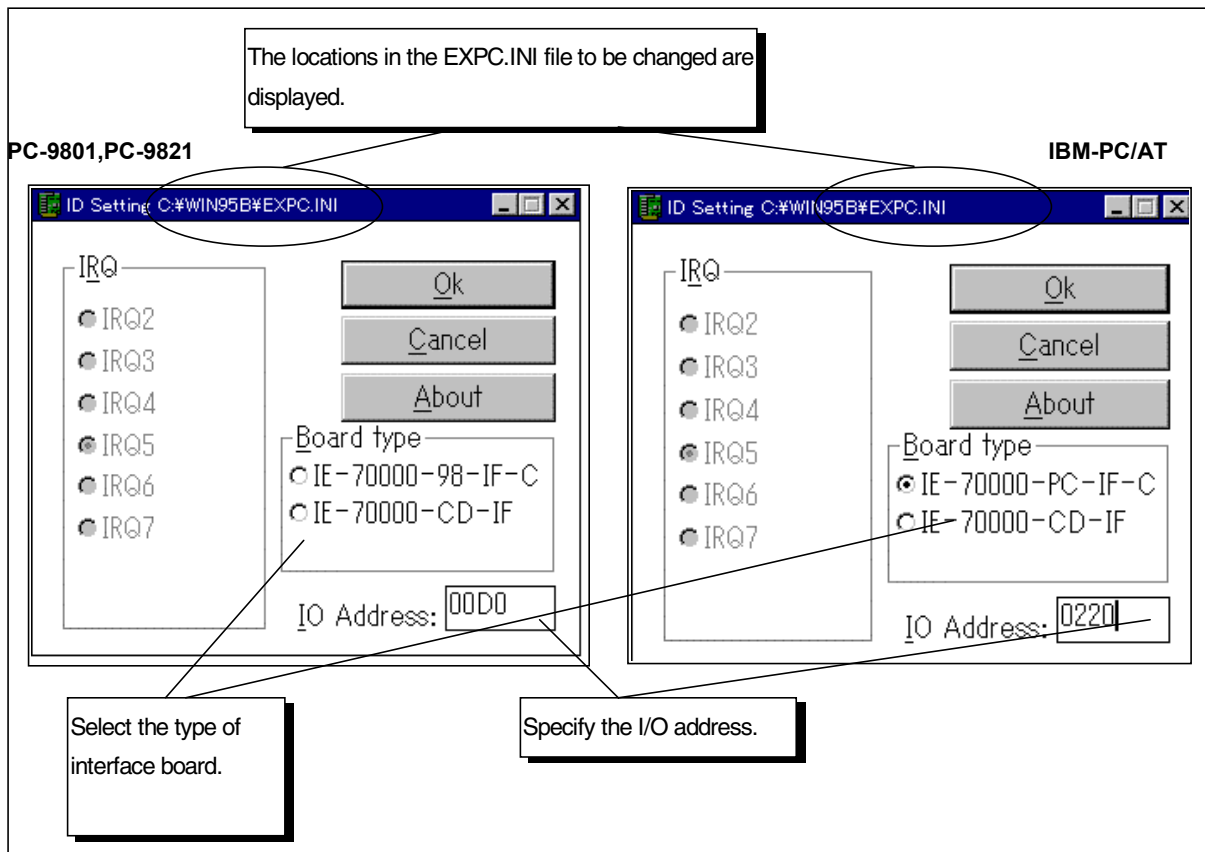
(10) Next, set the EXPC.INI file.

(11) Starting the utility for setting the EXPC.INI file

Since the icon shown below is in the group registered in (9), start the utility by double clicking.



(12) Input the settings for the PC interface board set in section 1.1, "Hardware Settings."



1. Board type

The PC interface board is selected. (Does not apply to the IBM-PC/AT.)

Select one of these three types.

Type	Remarks
IE-70000-98-IF-C	C bus interface board
IE-70000-PC-IF-C	ISA bus interface board
IE-70000-CD-IF	PCMCIA interface board

2. IRQ

The interrupt is selected.

If IE-78K4-NS is used, this does not have to be changed.

3. I/O address

The I/O address is input as a hexadecimal number.

The low-order 4 bits are fixed at 0.

Caution Make the settings of each item of the interface board, interrupt, and I/O address identical to the settings of the interface board in the host machine.

- (13) Next, install the device file.

The installation method differs depending on whether device file to be installed is compatible or not compatible with the installer. If compatible with the installer, go to (14). If not compatible, go to (15).

- (14) Install by following the device file installer. Go to (17).

- (15) Edit "necdev.ini" in the Windows directory.

As illustrated below, when ID78K4-NS is installed, necdev.ini is created in the Windows directory. Add the underlined sections.

<pre>[Installer] RootDir=c:\nectools Level=100 Group-NEC Tools [ProjectManager] Series1=78K4.PM [78K4.PM] Name=78K4 Series Debugger=c:\nectools\bin\id78K4ap.dll <u>Dev=78K4.DEV</u> <u>[78K4.DEV]</u> Dir=c:\nectools\dev <u>uPD784026=D4026.78K</u> <u>uPD784038=D4038.78K</u></pre>	<p>Add Dev=<u>78K4</u>.DEV to [<u>78K4</u>.PM].</p> <p>Create [<u>78K4</u>.DEV].</p> <p>Make Dir=xxx. xxx is the directory where the device file will be installed and is specified by its full path.</p> <p>Add the device name =Dxxx.78K (device file name).</p> <p>In this example, the device file is installed in c:\nectools\dev, and PD784026 and PD784038 are registered.</p>
---	---

- (16) Copy "Dxxx.78K" in the device file to the directory specified in (15). In the example in (15), copy the file in c:\nectools\dev.

- (17) This completes the installation of the debugger.

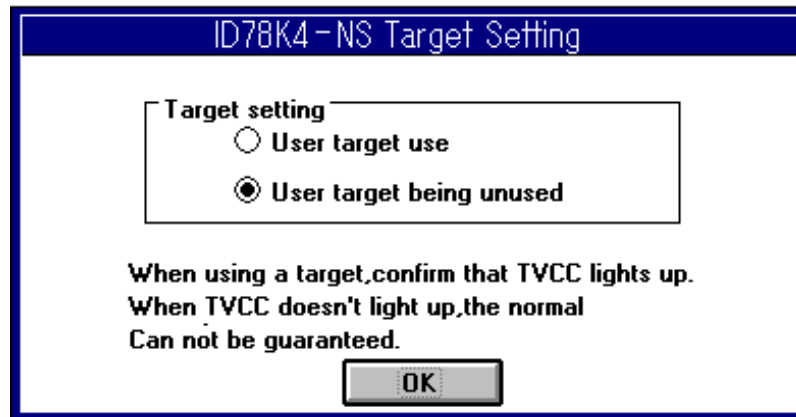
2.2 Starting and Exiting the Debugger

2.2.1 Starting

- (1) Start Windows.
- (2) Turn on the power supply to the in-circuit emulator.
- (3) If a target will be used, turn on the power supply to the target.
- (4) Use the mouse to double click the icon registered when the debugger was loaded.



- (5) When the debugger starts, the target setting dialog opens first.



If a target will be used, select "User target use."

If a target will not be used, select "User target being unused."

Note If a target will be used, verify that the TVCC is lit. If TVCC is not lit, normal operation cannot be guaranteed.

- (6) After checking the settings, click the  button to exit that target setting dialog.

- (7) When the debugger starts, the configuration dialog opens first.

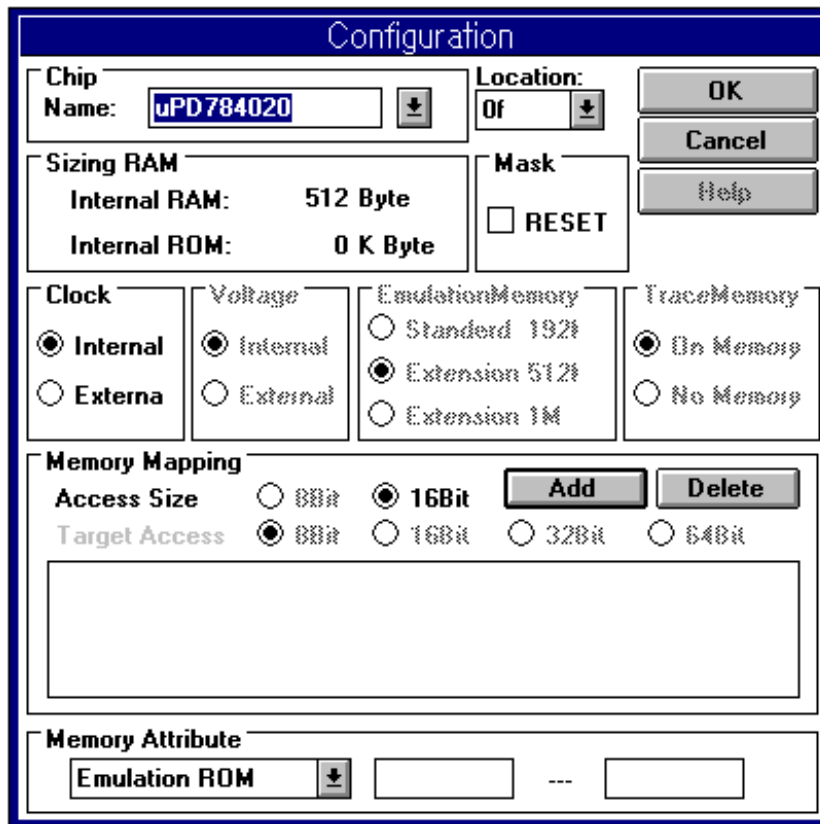



Figure 2-1. Configuration Dialog at Startup

- (8) Select the target device for debugging.
(The target device for debugging can only be selected at startup.)



- (9) Set the clock source, pin mask, location, and memory mapping.
(The location setting can only be selected at startup.)

- (10) After completing all of the settings, click the  button to complete the initial settings for the device and download the needed data to the in-circuit emulator.

- (11) When the download ends, the main window of the debugger opens. Debugging takes place centered around the main window.

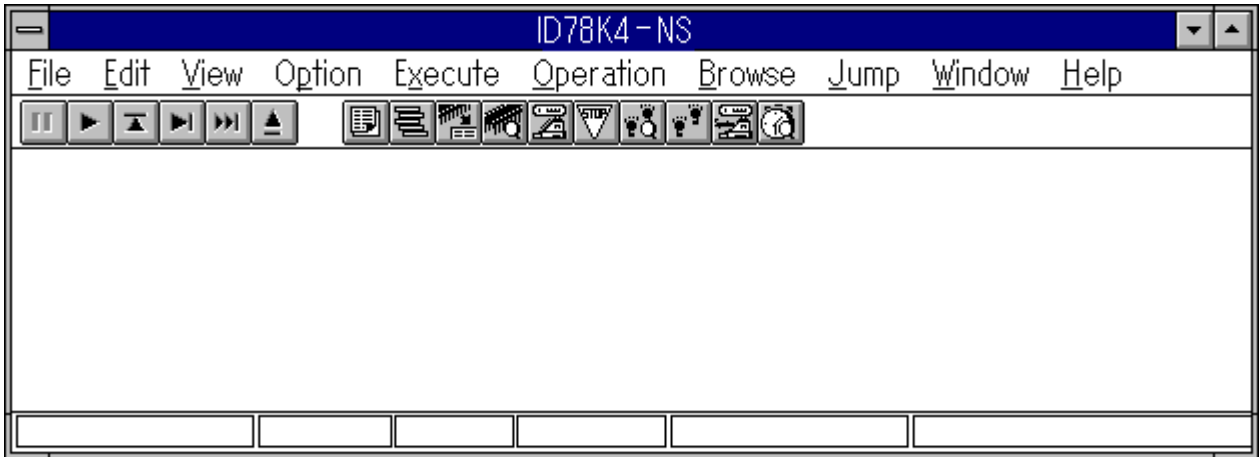



Figure 2-2. Startup Screen of the Debugger

2.2.2 Exiting

- (1) Select **F**ile in the **menu bar** in the main window.
- (2) Select **E**xit in the **F**ile pull-down menu.
- (3) Open the exit dialog.



- (4) The debugger can be exited by clicking the  button.

[MEMO]

CHAPTER 3 TERMINOLOGY DESCRIPTION

This chapter describes the terminology needed to explain how to use ID.

- (1) Debugging Modes**
- (2) Files**
- (3) Current File**
- (4) Functions**
- (5) Current Function**
- (6) Structures**
- (7) Stack Frame Number**
- (8) Line**
- (9) Real-time RAM Sampling**

3.1 Debugging Modes

When the window interface is used, the two debugging modes are the source mode and the instruction mode.

- In the source mode
Step execution is performed in one line units of the source text.
- In the instruction mode
Step execution is performed at the instruction level.

The debugging mode can be switched in the main window. When the debugger starts, the debugging mode is the source mode.

3.2 Files

ID handles the following types of files.

- Source files (*.C, *.ASM, *.S)
- Load module files (*.LNK)
- Hexadecimal files (*.HEX)
- Project files (*.PRJ)
- Display files (*.*)

3.3 Current File

The current file is the source file containing the instructions pointed to by the program counter (PC). If a line or a function in the current file is specified in a command, the file name can be omitted.

The file specification format is

- | |
|-------------------------|
| a. path\file
b. file |
|-------------------------|

(path: path name, file: file name)

- In a (when the path was specified)
The file is read from or written to the directory given by path.
- In b (when the path was not specified)
The file is read from or written to the current directory.

3.4 Functions

These functions form a C source program.

The function display and specification format are

- | |
|---------------------------|
| a. file#_func
b. _func |
|---------------------------|

(file: file name, func: function name)

- In a (when the file was specified)
func is interpreted as a valid static function in the specified file.
- In b (when the file was not specified)
Search for the corresponding function name first among the valid static functions and then the global functions in the current file.

Function specification example

test.c#_calc_data	"calc_data" static function in the "test.c" file
_main	"main" function that can be searched from the current file

3.5 Current Function

The current function is the function containing the instruction indicated by the program counter (PC). If local variables are accessed in the current function, the function name specification can be omitted.

3.6 Structures

The word structure refers to both the structures and the unions of the C language.

A structure is called by using a variable in the structure or the union without explicitly specifying a member.

3.7 Stack Frame Number

A stack frame number is a decimal number starting from 1. The functions in the stack are specified by the depth of the stack frame.

The largest stack frame number is for the current function.

3.8 Lines

The line specifies a particular line in the source file.

The line display and the specification format are

file:line

(file: file name, line: line number)

This is interpreted as the line at line number in the specified file.

Line specification example

test.c:100	Line 100 in the "test.c" file
------------	-------------------------------

3.9 Real-time RAM Sampling

Even while executing a user program, if the variables allocated to a space where the memory contents can be read or the memory is displayed, ID reads the memory contents and updates the display in real time. This function is called the real-time RAM sampling function. The memory address space is called the real-time RAM space. The real-time RAM space is mapped as follows depending on the Location instruction.

Location Instruction	Real-time RAM Space
Location 00H	0x00fd00 - 0x00feff
Location 0fH	0x0ffd00 - 0x0ffeff

CHAPTER 4 DEBUGGING WINDOW FUNCTIONS

4.1 Basic Window Operations

The basic operation when using the window interface to debug is a "noun + verb" operation. In other words, after selecting the debugging target (variable, line, task, etc.), the corresponding debugging function is selected by using a button (function button).

In addition, there are menus that have the same functions as the function buttons in the window, so debugging can also be performed by using the keyboard shortcut keys.

Next, the control objects needed to use ID are described.

(1) Mouse



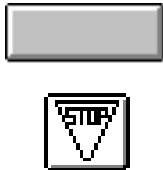
Mouse operation in the integrated debugger mainly uses the left mouse button. When mouse button is mentioned later, unless specifically stated, this means the left button. The three basic mouse operations are as follows.

Click : Press the mouse button once and release.

Double click : Consecutively press the mouse button twice and release.

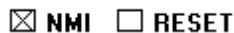
Drag & drop : While continuing to press the button, move to drag, and release the mouse button at the target location to drop.

(2) Push button and function button



The push button is a rectangular button with thickness and displays a bit map or a character string. Click the rectangular shape to start the corresponding process. The function button starts a debugging function.

(3) Check box



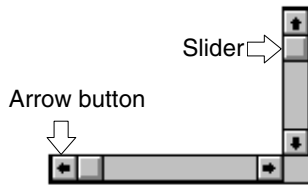
A check box consists of two parts, a square box and the selection text. Click the check box to change the square box display from ☐ to ☒. Multiple selections are possible.

(4) Radio button



A radio button consists of two parts, a circle and the selection text. Click the radio button to change the circle display from ☐ to ☒. If two or more radio buttons are grouped together, only one of them can be selected.

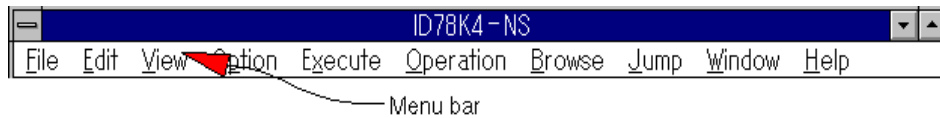
(5) Scroll bar



The scroll bars function to move the display contents vertically (vertical scroll bar) and horizontally (horizontal scroll bar). The slider in the scroll bar shows the proportion of the display of all of the contents that can be scrolled.

Clicking the arrow button moves one line in its direction. Dragging and dropping the slider moves to the relative position in all of the data.

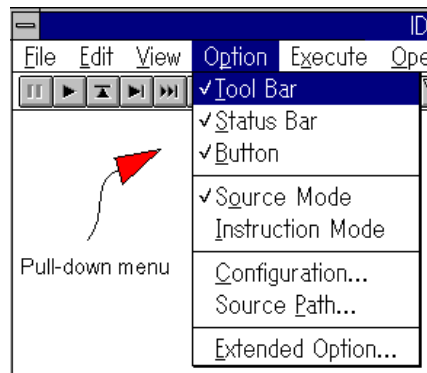
(6) Menu bar



The menu bar is displayed at the top of each window. Click an item in the menu bar to display the pull-down menu from the menu bar.

After pressing the **[GRPH]** key in the PC-9801, 9821 series or the **[Alt]** key in the IBM-PC/AT series, type the underlined letter in each item to obtain the same operation.

(7) Pull-down menu



The pull-down menu is an extension of the menu bar.

Click an item in the menu bar to start the corresponding process.

The same operation is obtained by typing the underlined letter in the item.

Without selecting the menu bar, an item is directly started by using **[CTRL] + letter** in an item of the pull-down menu. An item that can be directly started has "CTRL + letter" displayed to its right.

To facilitate understanding of the operations started by a menu selection, the menu items are classified as follows.

1. "Item"

After selecting this item, the contents of the item operate without doing anything else.

2. "Item..."

After selecting this item, a dialog requiring a response by the user is displayed.

3. "Item >"

After selecting this item, a cascade menu appears.

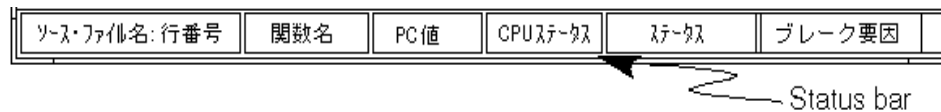
(8) Tool bar



The tool bar is a group of buttons that can execute relatively frequently used commands in one action. Each button is graphically displayed for easy understanding.

Click the button to operate.

(9) Status bar

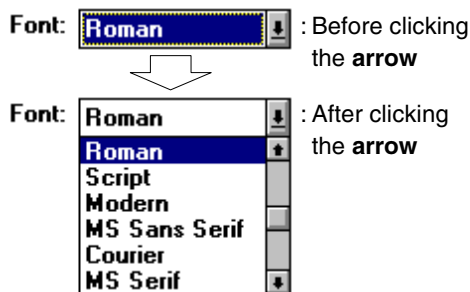


The status bar is an area that shows the states of the debugger and the in-circuit emulator.

The display in the order from the left is

- Source file name and line number indicated by the PC
- Function name indicated by the PC
- PC value
- CPU (μ PD784xxx) status
- Status of the in-circuit emulator
- Cause of a break

(10) Drop-down list



The drop-down list is a single item selection field that only displays the currently selected item. Another selection can be selected by clicking the arrow.

4.2 The Active State and the Hold State

The active state means the values displayed in the window are automatically updated when a user program or a command is executed. The hold state maintains the values regardless of the execution of a user program or a command.

In a window with displayed contents that are sometimes changed by executing the user program (display window, display/setting window), the window can switch to the active state or the hold state.

In **ID**, a window in the active state can only display one type of window. However, by setting a window in the hold state, it can simultaneously display multiple display windows of the same type.

When windows in the hold and active states are displayed together in the same type of display window or when a window in the hold state is switched to the active state, the message "Other view mode windows exit." is displayed and that window closes.

If a window is in the hold state, the background color is highlighted. **[HOLD xx]** is displayed in the title bar.

The following operations can change to the active state or the hold state.

Active state → Hold state

1. Select **Operation → Hold** in the menu bar.
2. Press the **CTRL + H** keys.

Hold state → Active state



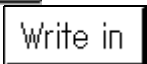
1. Select **Operation → Active** in the menu bar.
2. Press the **CTRL + I** keys.

4.3 View Mode and Modify Mode

The two window modes for ID windows are the view mode and the modify mode. These windows are

- Variable window
- Local variable window
- Memory window
- Register window
- SFR window
- Disassemble window

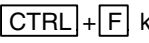
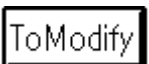
These windows usually function in the view mode, but by switching to the modify mode, the variables in the user program being debugged or the memory contents can be changed.

The values changed in the modify mode are reflected by clicking the  button which can be used in the modify mode. And by clicking the  button, all of the values changed in the modify mode are restored to their original values. However, if the  button has already been clicked, the values only return to a later state.

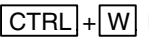

Only a window in the active state can move to the modify mode.

The operations below can move to the view mode and the modify mode.

View mode → Modify mode

1. Select **Operation → ToModify** in the menu bar.
2. Press the  keys.
3. Press the  button.

Modify mode → View mode

1. Select **Operation → ToView** in the menu bar.
2. Press the  keys.
3. Press the  button.

4.4 Errors and Warnings

ID handles errors and warnings differently.

All errors are generated by the debugger.

4.4.1 Errors and warnings during GUI operation

An error in GUI operation is regarded as a warning.

If a warning occurred, the warning tone sounds or the error/warning dialog appears.

4.4.2 Errors and warnings output by the debugger

If an error occurs, the error/warning dialog appears.

CHAPTER 5 DEBUGGING WINDOWS

5.1 Window Type and Layout

ID is composed of windows and dialogs.

A dialog is temporarily displayed to perform some operation. Both windows and dialogs are designed with the window manager. In contrast to a window that can be minimized to an icon, basically, a dialog cannot be minimized.

5.1.1 Windows

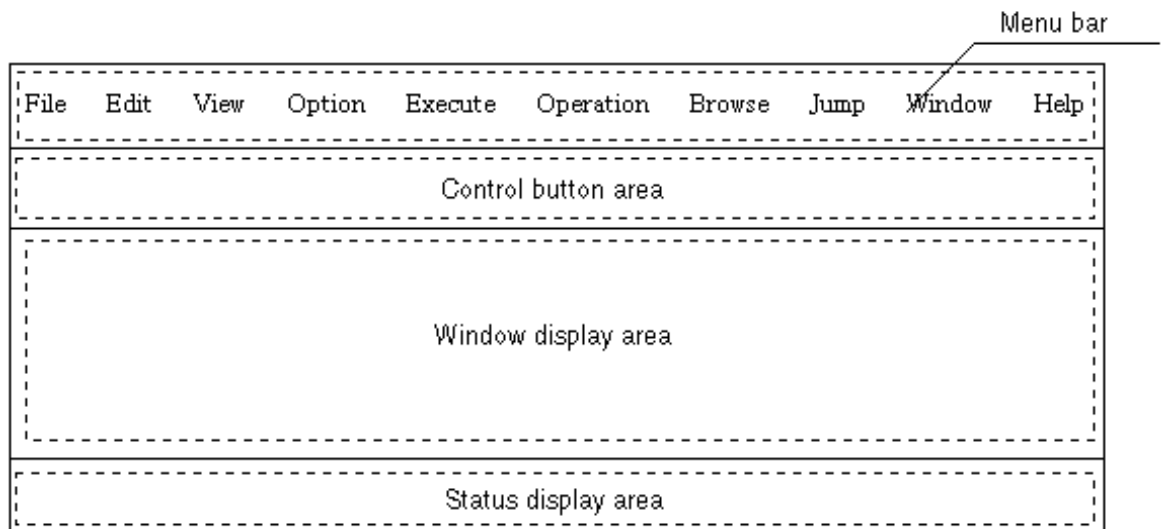
Windows are broadly classified by function into the following four window types.

- Execute window
- Display window
- Display/setting window
- Management window

Next, these windows are describe in detail.

(1) Execute window

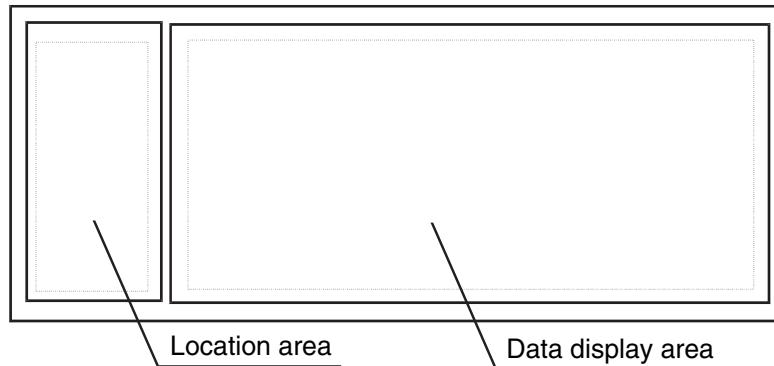
The execute window controls the windows and the execution of programs. It consists of a menu bar, control buttons, a window display area, and a status display area.



The main window has this window type.

(2) Display window

A display window displays the contents of the target. Display-only values cannot be changed. This window consists of a location area and a data display area.



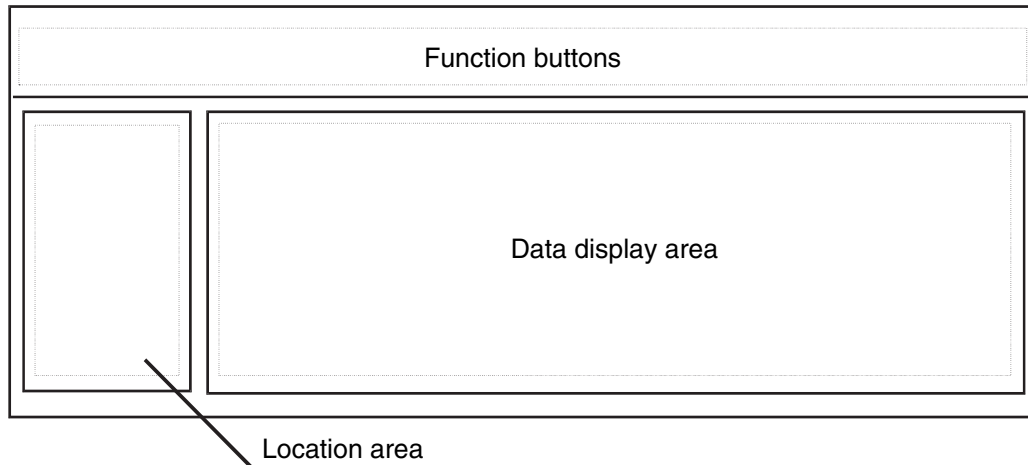
The source text window, stack trace window, and trace view window have this window type.

(3) Display/setting window

A display/setting window is used to display the contents and change the values of the target. Usually, only the contents are displayed, but the values can be changed by entering the modify mode. In addition, the two types of windows are windows opened in the main window and windows opened outside of the main window.

a. Type opened in the main window

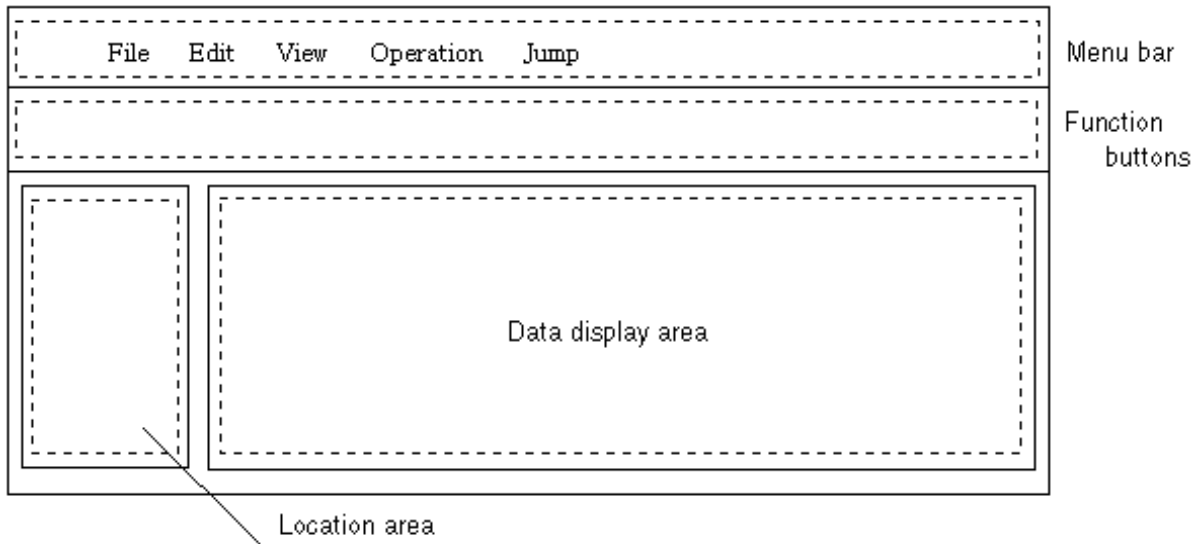
This window consists of function buttons, a location area, and a data display area.



The local variable window, memory window, SFR window, and disassemble window have this window type.

b. Type opened outside the main window

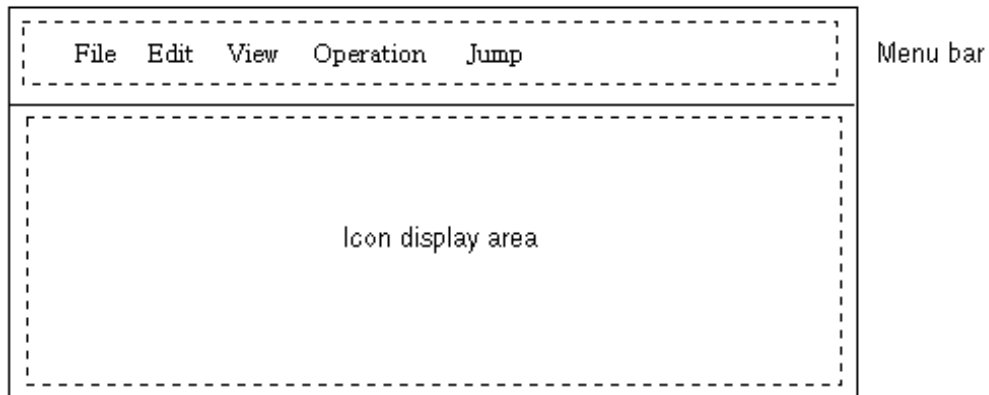
This window can be located anywhere outside the main window. However, the features of this window are it is always displayed in front of the main window and it cannot be minimized. The window consists of a menu bar, function buttons, a location area, and a data display area.



The register window and variable window have this window type.

(4) Management window

A management window manages the debugging settings. It consists of a menu bar and a data display area.



The event manager is this type of window.


5.1.2 Dialogs

Dialogs are broadly divided into the following two types.

- Modal dialogs
- Modeless dialogs

(1) Modal dialogs

If it has not been exited, this type of dialog cannot access other windows or dialogs in the debugger.

When the dialog is closed by finishing the operation of the dialog or selecting the  button in the dialog, other windows or dialogs can be accessed.

(2) Modeless dialogs

In contrast to a modal dialog, this type of dialog can access other windows or dialogs in the debugger even while the operation of the dialog has not ended.

The six types of dialogs by function are

- Selection dialog
- Specification dialog
- Setting dialog
- Confirmation dialog
- Auxiliary dialog
- Display dialog

(a) Selection dialogs

A selection dialog selects the conditions.

The configuration dialog, project file load dialog, upload dialog, view file save dialog, load module selection dialog, project file save dialog, view file load dialog, and source file selection dialog have this dialog type.

(b) Specification dialogs

A specification dialog specifies the conditions. Since conditions are usually specified, a text area is included.

The address specification dialog, source path specification dialog, and trace pick up dialog have this dialog type.

(c) Setting dialogs

A setting dialog sets the conditions.

The extended option setting dialog, event link dialog, trace dialog, event dialog, and break dialog have this dialog type.

(d) Confirmation dialogs

A confirmation dialog prompts for confirmation of the selected action.

The reset confirmation dialog, error/warning dialog, and exit confirmation dialog have this dialog type.

(e) Auxiliary dialogs

An auxiliary dialog is used for the auxiliary operations in each window.

The variable view dialog, memory copy dialog, memory compare dialog, add variable dialog, memory fill dialog, and search dialog have this dialog type.

(f) Display dialogs

A display dialog temporarily displays data.

The memory comparison result dialog and version display dialog have this dialog type.

(g) Display/setting dialogs

A display/setting dialog has an area for setting conditions and displaying data.

The timer dialog has this dialog type.

5.2 List of Debugging Windows

The table below lists the debugging windows.

Table 5-1. List of Debugging Windows (1/2)

Window Name	Description	Page
Main window	After the debugger starts, this is the first window displayed.	53
Configuration dialog	Sets the debugger operation environment.	67
Extended option dialog	Sets various extended options.	73
Open dialog	Reads in the debugging environment.	76
Save dialog	Saves the debugging environment.	79
Load module dialog	Reads in an object file or a symbol file.	82
Upload dialog	Uploads the memory contents to a file.	85
Source path dialog	Specifies the source path.	88
Open dialog	Selects the source file displayed in the source text window.	90
Source window dialog	Displays the source text.	93
Find dialog	Searches for a character string in the current window.	98
Symbol to address dialog	Displays the address allocated to the symbol.	101
Variable view dialog	Temporarily displays the variable values.	103
Variable window dialog	Displays and changes variables.	105
Add variable dialog	Adds the displayed variables to the variable window.	110
Local variable window	Displays and changes the local variables in the current function.	113
Address specification dialog	Specifies the display starting address.	116
Disassemble window	Displays the disassembly of the program and assembles on-line.	119
Memory window	Displays and changes the memory contents.	125
Memory fill dialog	Initializes the memory.	129
Memory copy dialog	Copies the memory.	131
Memory compare dialog	Compares the memory.	133
Memory compare dialog	Displays the result of the memory comparison.	135
Stack trace window	Displays the contents of the function's stack.	137
Event set dialog	Registers the event conditions.	140
Event manager	Manages each registered event condition.	146
Event link dialog	Registers the event link conditions.	155
Break dialog	Registers and sets the break event conditions.	161
Trace dialog	Registers and sets the trace event conditions.	165
Timer window dialog	Displays the result of run time measurements.	170
Trace view window	Displays the trace result.	172
Trace window dialog	Sets the trace display conditions.	177

Table 5-1. List of Debugging Windows (2/2)

Window Name	Description	Page
Register window	Displays and changes the registers	181
SFR window	Displays and changes the SFR.	187
Save dialog	Saves the display contents of the current window in a file	191
Error/warning dialog	Displays errors and warnings.	196
Reset debugger dialog	Resets the debugger and target CPU	197
About dialog	Displays the debugger version.	199
Exit debugger dialog	Exits the debugger.	200

5.3 Descriptions of the Debugging Windows

This section provides an in-depth description of each debugging window in the following format.

Window Name	Window Type (Dialog mode)
-------------	---------------------------

The window name and type (mode type for a dialog) are described in the frames.

Overview

The window is briefly explained.

Window

The layout of the window is shown by a screen image.

Functions

The contents displayed in the window are explained.

Function buttons

The operations by the buttons in the window are described.

Menu bar

The pull-down menu from the target title in the menu bar is listed and each function is described.

Main Window**Execute Window****Overview**

This window is automatically opened first after the debugger starts and the initialization exits. It remains on the screen until the debugger ends. The other windows operate centered on this window.

Execution control of the user program takes place in this window.

The two modes for execution control of user programs are the source mode and the instruction mode.

- **Source Mode**
Debugging is at the source level.
- **Instruction Mode**
Debugging is at the instruction level.

The mode when the debugger starts is the source mode.

Window

Figure 5-1. Main Window

Functions

The main window consists of the following items.

- **Menu bar**
- **Tool bar**
- **Window display area**
- **Status display area**

Each function is described next.

(1) Tool bar

A tool bar is a group of the buttons of commands which are executed relatively frequently and can be executed in one action. Each button is displayed as a graphical image and is easy to understand. Each button function can be executed even from the menu bar. If hide was selected in **Option** → **Tool Bar** in the menu bar, the tool bar is not displayed.



Stops the user program execution.



Executes the user program.

During program execution, the button remains in the pressed state and returns to its original state after the program stops.



Executes in real time until returning to the calling function.



Executes in single steps.

By continuously clicking, that number of steps is executed.

If the debugging mode is the source mode, step execution is in line units. If it is the instruction mode, step execution is in instruction units.



Executes the next step.

By continuously clicking, that number of next steps is executed.

If the debugging mode is the source mode, step execution is in line units. If it is the instruction mode, step execution is in instruction units.



Initializes the debugger, the emulation CPU, and the symbol data.

The reset confirmation dialog opens.



Displays the source text.

The source text window opens.



Displays the stack contents.

The stack trace window opens.



Displays the assembly results.

The disassemble window opens.



Displays the memory contents.

The memory window opens.



Displays the register contents.

The register window opens.



Registers and sets break events.

The break dialog opens.



Displays the trace result.
The trace view window opens.



Registers and sets trace events.
The trace dialog opens.



Displays the SFR contents.
The SFR window opens.



Displays the timer measurements.
The timer dialog opens.

(2) Window display area

This area displays the debugging windows.

In this area, the displayed window has its size changed or is minimized to an icon.

The windows displayed in this area are listed below.

Source text window	Disassemble window
Local variable window	Trace view window
Memory window	SFR window
Stack trace window	

(3) Status display area



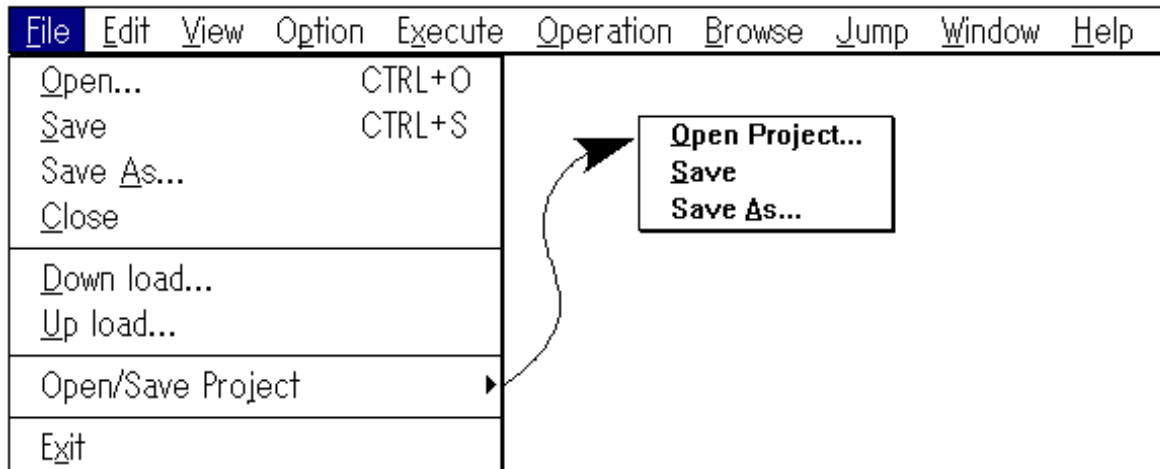
This area displays the debugger and in-circuit emulator states.

Source file name	Displays the source file name and the source line number pointed to by the PC value. If there is no file data, "---" is displayed.
Function name	Displays the function name pointed to by the PC value. If there is no file data, "---" is displayed.
PC value	Displays the current PC value.
CPU status	Displays the CPU state (STOP, HALT, IDLE modes, etc.).
Status	Displays the in-circuit emulator state (RUN, BREAK modes, etc.).
Break cause	his area displays the cause of a break. The types of break causes are as follows.

Cause Display	Meaning
Compulsory Break	Normal break
Event Break	Break by an event
Non Map Break	Accessed the nonmapped area
Relocation Bleak	Executed a relocation instruction different from the initial setting
SFR Illegal	Illegal access to the SFR
Stack Overflow	Break caused by stack overflow
Write Protect	Write to a write-protected region

Menu Bar

(a) File



Open

The operation differs depending on the current window.

When the source text window is current:

The source view file is selected.

The source file selection dialog opens.

Otherwise:

The view file for the current window is displayed.

The view file save dialog opens.

Save

Saves the display contents of the current window to the view file.

Save As...

Saves the display contents of the current window to a file with a different name.
The view file save dialog opens.

Close

Closes the current window.

Download...

Downloads a program. The load module selection dialog opens.

Upload

Uploads a program. The upload dialog opens.

Open/Save Project ►**Open Project...**

Opens the project file. The project file load dialog opens.

Save

Writes the current state to the project file.

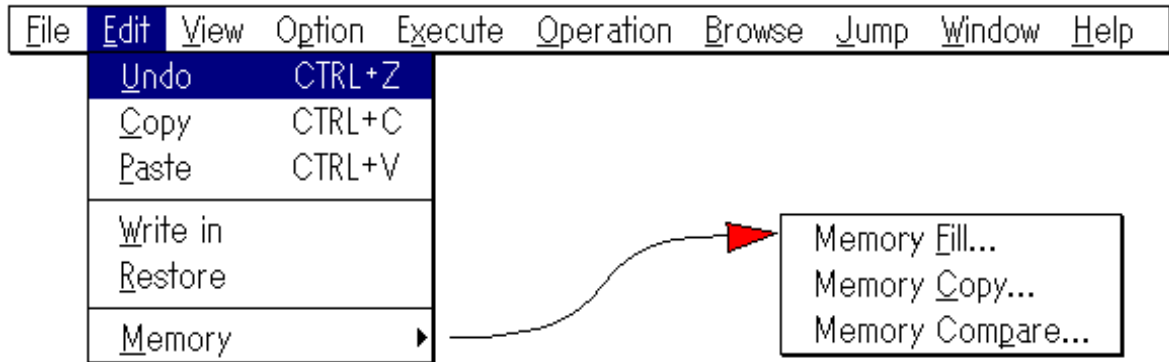
The overwritten file is the file selected by **Open...** or **Save As... in File → **Open/Save Project**** in the menu bar.

Save As...

Saves the current state in the project file. The project file save dialog opens.

Exit

Exits the debugger. The exit confirmation dialog opens.

(b) Edit**Undo**

Undoes the previous editing operation.

Copy

Copies the selected string to the clipboard buffer.

Paste

Pastes the contents of the clipboard buffer to the position of the text cursor.

Write in

Performs the same action as the **Write in** button to write the changes to the target.

Restore

Performs the same action as the **Restore** button to cancel the changes.

Memory ►**Memory Fill...**

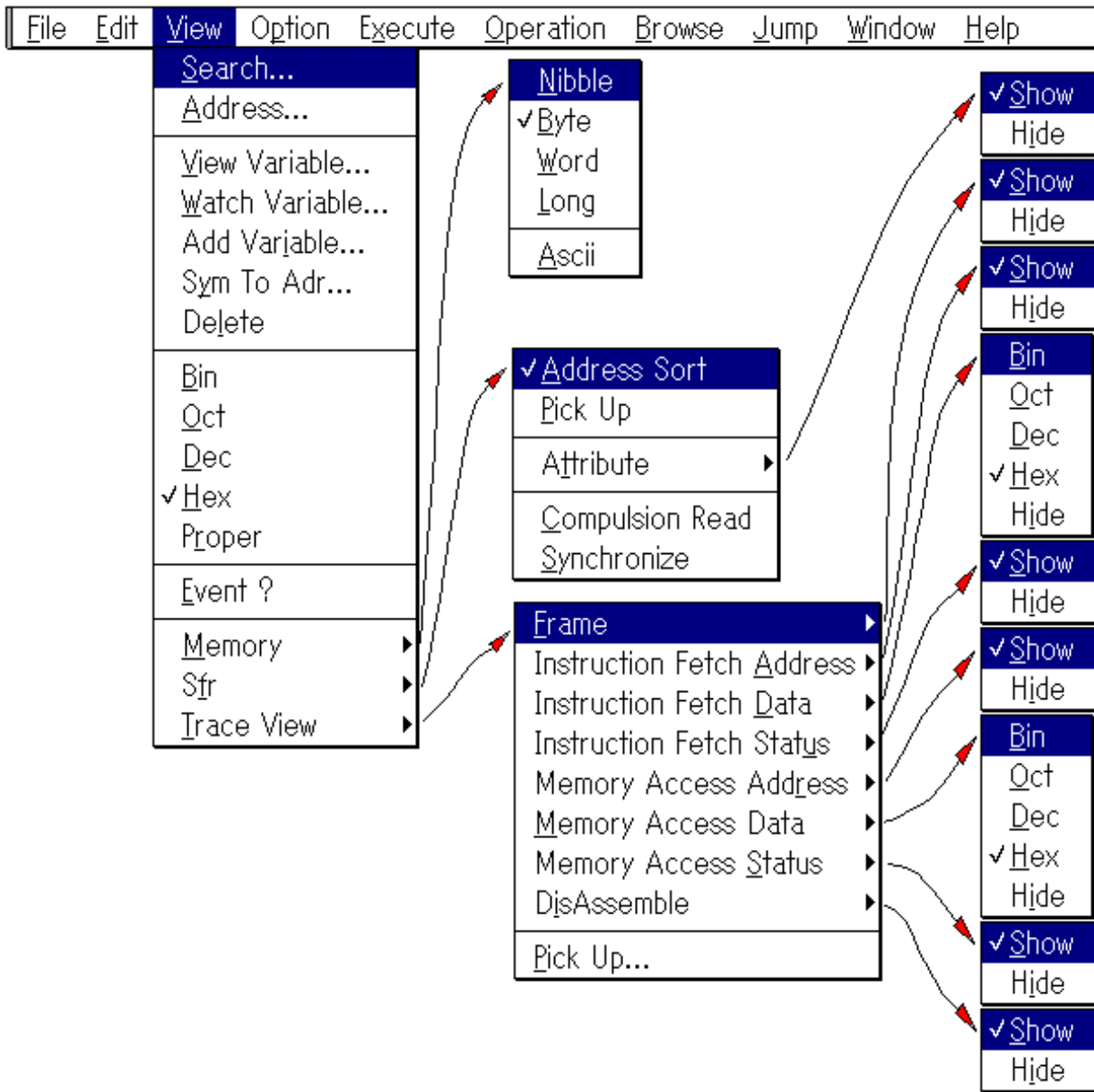
Initializes the memory. Opens the memory fill dialog.

Memory Copy...

Copies the memory. Opens the memory copy dialog.

Memory Compare...

Compares the memory. Opens the memory compare dialog.

(c) **View****Search...**

Searches for strings or numerical values. The search dialog opens. The action is identical to the **Search** button.

Address...

Displays the contents of the specified address. The address specification dialog opens.

View Variable...

Temporarily displays the specified variable. The variable view dialog opens.

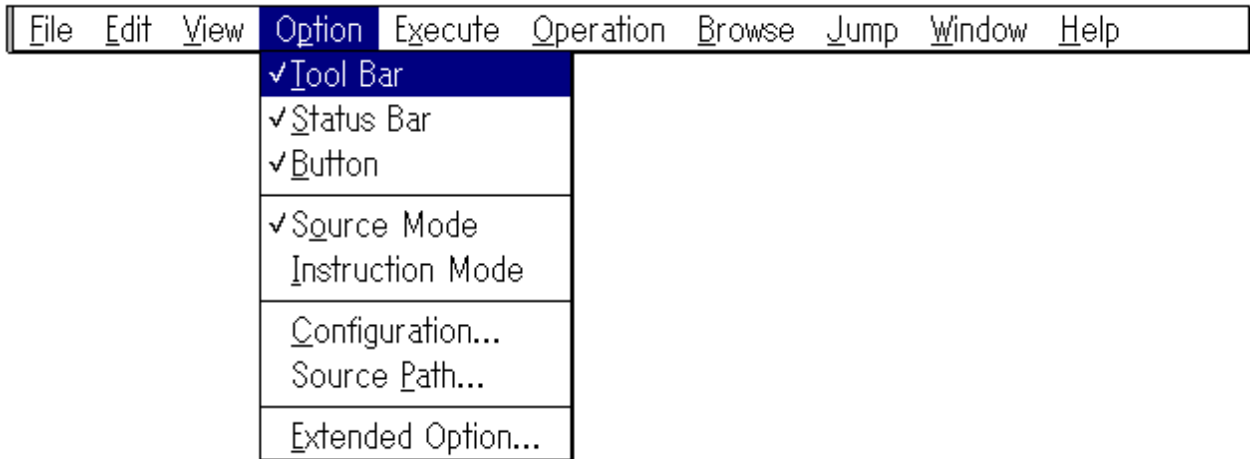
Watch Variable...

Displays a variable. The variable window opens.

Add Variable...	Adds the specified variable in the variable window. The add variable dialog opens.
Sym To Adr...	Displays the specified variable address. The change symbol dialog opens.
Delete	Deletes the specified variable.
<u>B</u>in	Displays in binary.
<u>O</u>ct	Displays in octal.
<u>D</u>ec	Displays in decimal.
<u>H</u>ex	Displays in hexadecimal.
Proper	Displays by using an appropriate value for each variable. (Default)
<u>E</u>vent ?	Displays event data. The event manager opens.
<u>M</u>emory ▶	
<u>N</u>ibble	Displays in 4-bit units.
<u>B</u>yte	Displays in 8-bit units. (Default)
<u>W</u>ord	Displays in 16-bit units.
<u>L</u>ong	Displays in 32-bit units.
<u>A</u>scii	Switches the display of ASCII characters on or off.
<u>S</u>fr ▶	
<u>A</u>ddress Sort	Specifies the order of the SFR display. No check ' ': Displays in alphabetical order. Check 'Ö': Displays in the address order.
Pick U	Displays only changed SFR.
<u>A</u>tttribute ▶	
<u>S</u>how	Displays the SFR attribute. (Default)
<u>H</u>ide	Does not display the SFR attribute.
<u>C</u>ompulsion Read	Forced read of a read-protected SFR.
<u>S</u>ynchronize	Writes the changed SFR to the target.
<u>T</u>race View ▶	
<u>F</u>rame ▶	Selects the addition or deletion of the frame number display field.
<u>S</u>how	Displays the frame number. (Default)
<u>H</u>ide	Does not display the frame number.
Instruction Fetch <u>A</u>ddress ▶	Selects the addition or deletion of the fetch address display field.
<u>S</u>how	Displays the address. (Default)
<u>H</u>ide	Does not display the address.
Instruction Fetch <u>D</u>ata ▶	Selects the addition or deletion of the fetch data display field.
<u>B</u>in	Displays the data in binary.

<u>O</u>ct	Displays the data in octal.
<u>D</u>ec	Displays the data in decimal.
<u>H</u>ex	Displays the data in hexadecimal. (Default)
<u>H</u>ide	Does not display the data.
Instruction Fetch Status ▶	Selects the addition or deletion of the fetch status display field.
<u>S</u>how	Displays the status. (Default)
<u>H</u>ide	Does not display the status.
Memory Access Address ▶	Selects the addition or deletion of the access address display field.
<u>S</u>how	Displays the address. (Default)
<u>H</u>ide	Does not display the address.
Memory Access Data ▶	Selects the addition or deletion of the access data display field.
<u>B</u>in	Displays the data in binary.
<u>O</u>ct	Displays the data in octal.
<u>D</u>ec	Displays the data in decimal.
<u>H</u>ex	Displays the data in hexadecimal. (Default)
<u>H</u>ide	Does not display the data.
Memory Access Status ▶	Selects the addition or removal of the access status display field.
<u>S</u>how	Displays the status. (Default)
<u>H</u>ide	Does not display the status.
DisAssemble ▶	Selects the addition or removal of the disassemble display field.
<u>S</u>how	Displays the disassemble data. (Default)
<u>H</u>ide	Does not display the disassemble data.
<u>P</u>ick Up...	Sets the trace display condition. The trace pick up dialog opens.

(d) Option




Tool Bar	Selects to display or to hide the tool bar.
Status Bar	Selects to display or to hide the status bar.
Button	Selects to display or to hide the buttons in each window.
Source Mode	Step executes at the source level.
Instruction Mode	Step executes at the instruction level.
Configuration...	Sets the environment. The configuration dialog opens.
Source Path...	Sets the source path data. The source path specification dialog opens.
Extended Option...	Sets the extended functions. The extended option setting dialog opens.


(e) Execute

File	Edit	View	Option	Execute	Operation	Browse	Jump	Window	Help
				Stop	CTRL+P				
				Go	CTRL+G				
				Return	CTRL+R				
				Step	CTRL+T				
				Next	CTRL+X				
				Go & Go					
				Come					
				Slowmotion					
				CPU Reset & Go					
				CPU Reset...					
				Set BP	CTRL+B				
				Set PC					
				✓ Uncond. Trace ON					
				Cond. Trace ON					


Stop

Stops the program execution. The action is identical to the  button.


Go

Executes the program. The action is identical to the  button.


Return

Executes in real time until returning to the calling function. The action is identical to the  button.


Step

Executes in steps. The action is identical to the  button.

Next

Executes the next step. The action is identical to the  button.

Go & Go

The program continues to execute. If a break is generated by a break condition, after the window is updated, the program resumes execution. The action is identical to clicking the  button when a break occurs.

Come


Executes in real time until the specified address.

Slowmotion

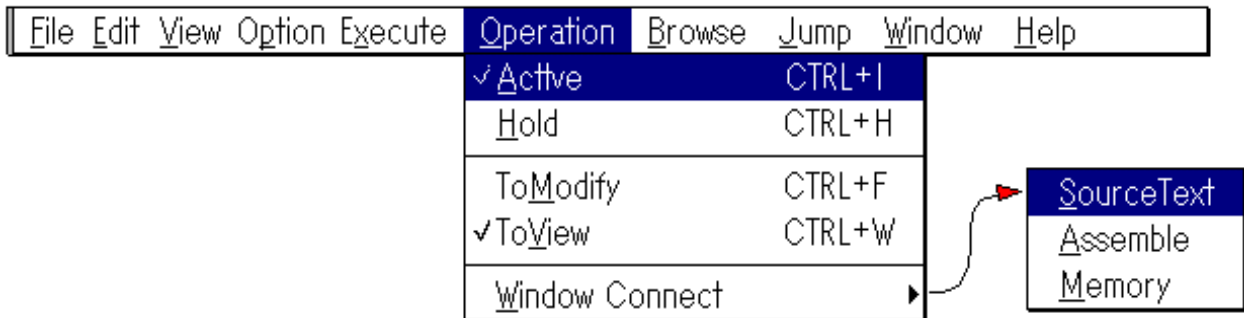
Continues the step execution.



CPU Reset & Go

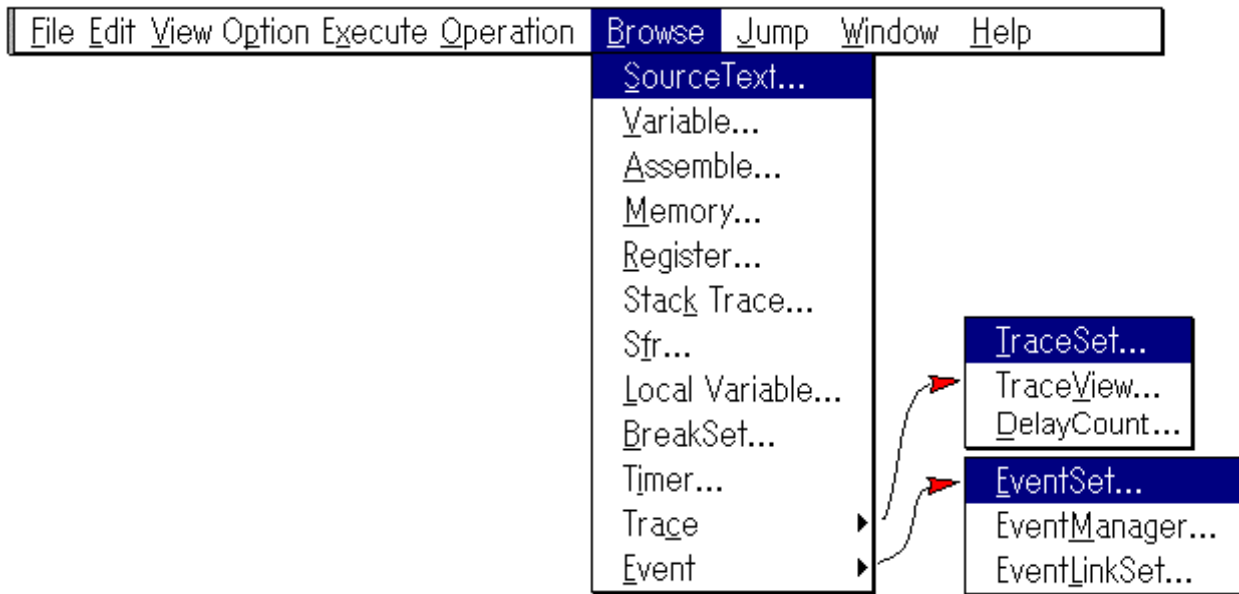
Executes the program after resetting the target.


CPU Reset...	Resets only the target or the entire debugger. The reset confirmation dialog opens. The action is identical to the  button.
Set BP	Sets the breakpoint in the selected line.
Set PC	Sets the PC register to the address at the selected line.
Uncond. Trace ON	Enables the tracer and sets continuous tracing during program execution.
Cond. Trace ON	Enables the tracer and sets tracing that conforms to the trace conditions during program execution.

(f) **Operation**



Active	Switches the window to the active state.
Hold	Switches the window to the hold state.
ToModify	Switches the window to the modify mode. The action is identical to the  button.
ToView	Switches the window to the view mode. The action is identical to the  button.
Window Connect ►	Sets a window connection with the trace window.
Source Text	Connects the trace window to the source text window.
Assemble	Connects the trace window to the assemble window.
Memory	Connects the trace window to the memory window.


(g) **Browse****SourceText...**

Displays the source text. The source text window opens. The action is identical to the  button.


Variable...

Displays the specified variables. The variable window opens.


Assemble...

Displays the assemble. The disassemble window opens. The action is identical to the  button.


Memory...

Displays the memory contents. The memory window opens. The action is identical to the  button.


Register...

Displays the register contents. The register window opens. The action is identical to the  button.

Stack Trace...

Displays the stack contents. The stack trace window opens. The action is identical to the  button.


Sfr...





Displays the SFR contents. The SFR window opens. The action is identical to the  button.

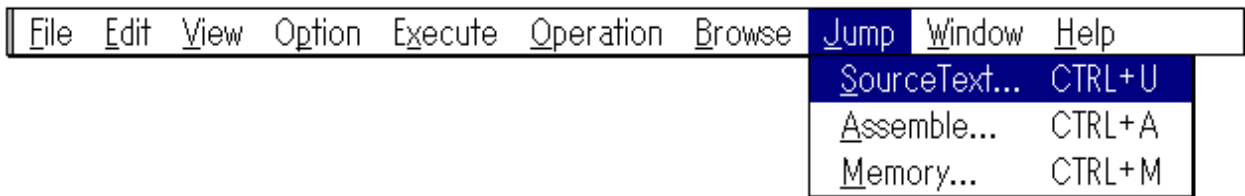
Local Variable...

Displays the local variables. The local variable window opens.

BreakSet

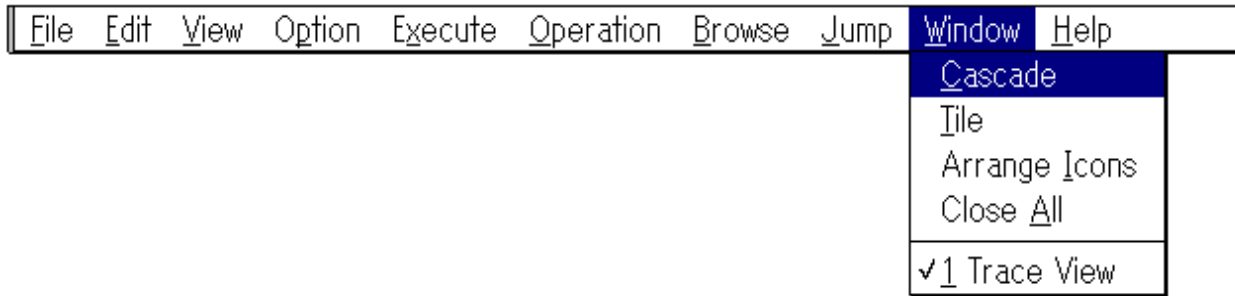
Adds and sets the break conditions. The break dialog opens. The action is identical to the  button.

T imer...	Displays the registration setting of timer event conditions and the measurements. The timer dialog opens. The action is identical to the  button.
T race ▶	The trace-related window opens.
T raceSet...	Registers and sets the trace event conditions. The trace dialog opens. The action is identical to the  button.
T raceView...	Opens a log. The action is identical to the  button. The trace result is displayed. The trace view window opens. The action is identical to the  button.
D elayCount...	Sets the delay count. The delay count setting dialog opens.
E vent ▶	Opens the event-related window.
E ventSet...	Sets the event conditions. The event dialog opens.
E ventM <u>a</u> anager...	Manages various event conditions. The event manager opens.
E ventL <u>i</u> nkSet...	Sets the event link conditions. The event link dialog opens.

(h) **J**ump

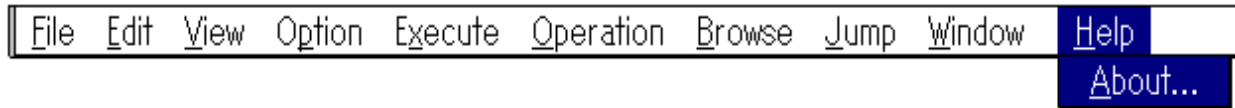
S ourceText...	Displays the source text and the source lines which correspond to the data value selected in the current window as the jump destination address. However, if there is no line data in the jump destination address, jumping is not possible. The source text window opens.
A ssemble...	Displays the disassembly result from the address where the data value selected in the current window is the jump destination address. The disassemble window opens.
M emory...	Displays the memory contents beginning at the address where the data selected in the current window is the jump destination address. The memory window opens.

(i) **Window**




Cascade	Displays the windows in the main window in a cascade.
Tile	Tiles the windows in the main window on the display.
Arrange Icons	Rearranges the icons in the main window.
Close All	Closes all of the windows except for the main window.
[Open Windows]	Displays the list of the names of open windows. By selecting the window name, the selected window becomes the current window.

(j) **Help**



About...	Displays the debugger version.
-----------------	--------------------------------



The main window can be displayed as an icon by clicking the  button in the title bar. The icon itself and the icon title can be freely set and created by the user.



Configuration Dialog	Selection Dialog (Modal)
-----------------------------	---------------------------------

Overview

The operating environment of the in-circuit emulator is displayed and set.

This dialog is displayed first when the debugger starts. **When using the debugger, the operating environment of the in-circuit emulator must first be set in this dialog.**

However, if a project file is read, this setting is not required. The result of reading the project file is reflected in the configuration dialog.

In addition, even during debugging, this dialog can modify and add the pin mask settings, location setting, and memory mapping setting when needed.

This dialog can be opened by the following methods.

- When starting the debugger
This dialog opens automatically.
- In the main window
Select **Option** → **C**onfiguration... in the menu bar.
- In the main window
Press in order the **G**RPH, **P**, and **C** keys.

Window

Configuration

Chip
 Name:
 Location:

Sizing RAM
 Internal RAM: 512 Byte
 Internal ROM: 0 K Byte

Mask
☐ RESET

Clock
☒ Internal
☐ External

Voltage
☒ Internal
☐ External

EmulationMemory
☐ Standard 192t
☒ Extension 512t
☐ Extension 1M

TraceMemory
☒ On Memory
☐ No Memory

Memory Mapping
 Access Size ☐ 8Bit ☒ 16Bit
 Target Access ☒ 8Bit ☐ 16Bit ☐ 32Bit ☐ 64Bit

Memory Attribute

Figure 5-2. Configuration Dialog

Functions

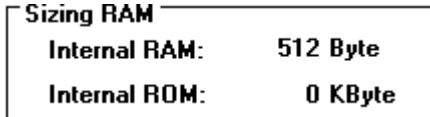
The configuration dialog contains these items.

- Emulation CPU selection area
- Internal ROM/RAM display area
- CPU clock source selection area
- Drive voltage display area
- Mask setting area
- Location setting area
- Mapping setting area
- Loaded emulation/trace memory display area
- Function buttons

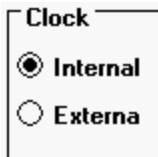
Each function is described next.

(1) Emulation CPU selection area

The emulation CPU is selected in this area.
The selection can only be made at startup.

(2) Internal ROM/RAM display area

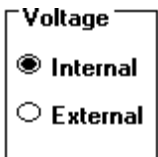
This area displays the sizes of the internal ROM and internal RAM in the emulation CPU.
When the emulation CPU is selected, this is automatically displayed.

(3) CPU clock source selection area

The CPU clock source is selected in this area.
One of the following two choices is selected. The default is Internal.

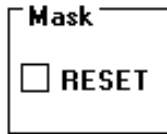
- **Internal:** The clock of the in-circuit emulator is used as the CPU clock. (25 MHz)
- **External:** The target clock is used as the CPU clock.

This can be selected at startup or by loading the project file.

(4) Drive voltage display area

This area displays the voltage input to the emulation CPU.
The voltage input to the emulation CPU is automatically displayed.

- **Internal:** The voltage of the in-circuit emulator is used as the drive voltage.
(The drive voltage is fixed at 5 V.)
- **External:** The voltage at the terminal pin of the I/O board is the drive voltage.
(The drive voltage can be varied within the range of the device specifications at low voltages below 5 V.)

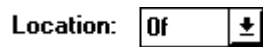
(5) Mask setting area

The mask for the signal sent from the target is set.

This signals at masked pins are not input to the in-circuit emulator. (Mask pins only when the operation of the target is unstable in the debugging phase.)

Only the following pin can be masked.

- RESET pin

(6) Location setting area

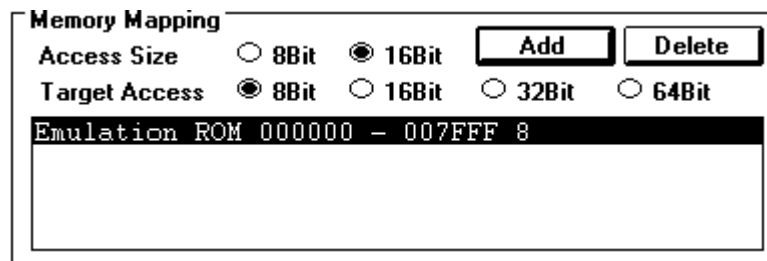
The location is set.

The two possible settings are

- Location 00H
- Location 0fH

Set to fit the environment to be used. If a Location instruction which differs from this setting was executed, a ReLocation Break is generated.

After debugging starts, this selection cannot be made.

(7) Mapping setting area

The bus width of the memory is selected and the mapping is set.

a. Selecting the bus width

These bus widths can be selected.

- 8Bit
- 16Bit

(However, the bus widths that can be selected differ with the target chip.)

b. Setting the memory mapping

- When adding memory mapping

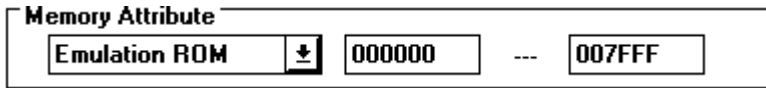
Click the  button.

Memory mapping corresponding to the **Memory Attribute** data and the bus width data is added.

- Deleting memory mapping

Click the  button.

The currently selected mapping is deleted.

(8) Mapping specification area


The **Memory Attribute** section contains a dropdown menu with 'Emulation ROM' selected, a button with a plus-minus icon, and two address fields: '000000' and '007FFF' separated by a range operator '---'.

The type and range for the memory mapping are set.

The three types of mapping shown below can be selected. Select the one suited to the application.

- **Emulation ROM:** Selects the in-circuit emulator substitute ROM.
- **Emulation RAM:** Selects the in-circuit emulator substitute RAM.
- **Target:** Selects the target memory.

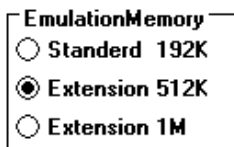
The mapping unit changes depending on the addresses to be mapped. The mapping units are given below. However, the stack can be mapped in 1-byte units.

Mapping Address	Mapping Unit
0x00000 - 0x00ffff	4Kbyte
0x010000 - 0x0fffff	64Kbyte
0x100000 - 0xffffffff	1Mbyte

The size in the in-circuit emulator changes based on the loaded memory of 192 kbytes, 512 kbytes, or 1 Mbyte.

Set the mapping for the loaded memory. If memory mapping to an address above 1 Mbyte, select Target.

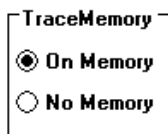
If the mapping unit is not matched, the minimum range that can be set and contains the specified address becomes the mapping target.

(9) Loaded emulation/trace memory display area


The **EmulationMemory** section has three radio buttons: 'Standard 192K', 'Extension 512K' (which is selected), and 'Extension 1M'.

This area displays the loaded state of the emulation memory.

This is automatically displayed based on the loaded state of the emulation memory.




The **TraceMemory** section has two radio buttons: 'On Memory' (which is selected) and 'No Memory'.


This area displays the loaded state of the trace memory.


This is automatically displayed based on the loaded state of the trace memory.


Function Buttons

 button Sets the current environment. The environment is set and then the dialog closes.

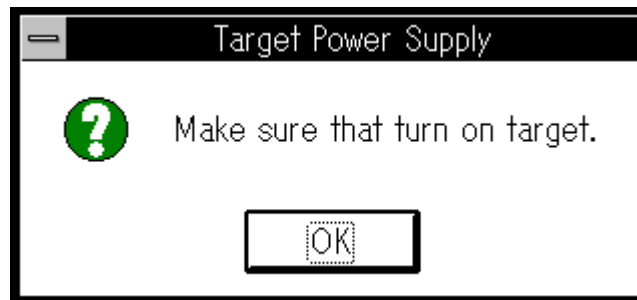
 button Returns to the environment setting state before the dialog was opened.

 button Cancels the changes and closes the dialog.

 button Opens the help window to explain the configuration dialog.

Cautions If even one item of the following settings in the configuration dialog was set, power must be applied to the target. The message prompting to turn on the power is displayed. After verifying that the power is applied, press the  button.

Item	Setting
CPU clock source selection area	When External was selected
Drive voltage selection area	When External is displayed
Mapping setting area	When mapped to Target



If the power is not applied to the target, the following message is displayed and the debugger exits.



Extended Option Dialog**Setting Dialog (Modal)****Overview**

The extended options of the debugger are displayed and set.
This dialog can be opened by the following methods.

- In the main window
Select **Option** → **Extended Option...** in the menu bar
- In the main window
Press in order the **GRPH**, **P**, and **E** keys.

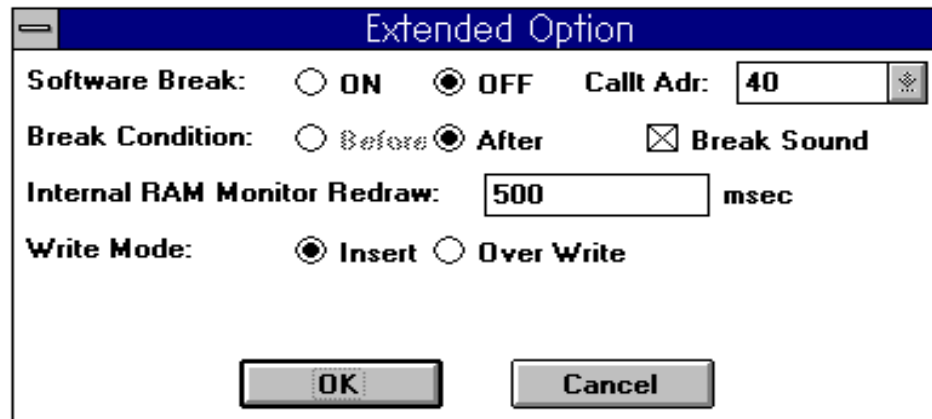
Window

Figure 5-3. Extended Option Setting Dialog

Functions

The extended option setting dialog contains these items.

- Software break setting area
- Break mode setting area
- Real-time internal RAM sampling time setting area
- Write mode selection area
- Function buttons

Each function is described next.

(1) Software break setting area

Software Break: ☐ ON ☒ OFF Callt Adr: 

The software breaks are set.

ON: Use software breaks.

OFF: Do not use software breaks.

Callt Adr: Vector address of the CALLT instruction that is released to the debugger

If software breaks are used, one of the vector address of the CALLT instruction must be released to one debugger.

(2) Break mode setting area

Break Condition: ☒ Befor ☐ After ☒ Break Sound

The event mode and the sound when a break occurs are set.

Befor: Sets the RUN event to a before-execution event. The event is generated before the instruction is executed.

After: Sets the RUN event to an after-execution event. The event is generated after the instruction is executed.

Break Sound: Specifies the sound generated when the in-circuit emulator breaks.

☒ **Break Sound:** Sound is emitted.

☐ **Break Sound:** No sound is emitted.

(3) Real-time internal RAM sampling time setting area

Internal RAM Monitor Redraw: msec

The real-time internal RAM sampling time is set.

The range of the real-time sampling time is given below.

Location	Sampling Range
00H	512 bytes in addresses 0x00fd00 - 0x00feff
0FH	512 bytes in addresses 0x0ffd00 - 0x0ffe00

Variables or data allocated in this range can be displayed in real time in the variable window and the main window.

The sampling time can be set in 1 millisecond units.

(4) Write mode selection area

Write Mode: ☒ **Insert** ☐ **Over Write**

The write mode is specified in the modify mode. The two types are

- **Insert:** Modify in the insert mode.
- **Over Write:** Modify in the overwrite mode.

Function buttons



button Accepts this change and closes the dialog.



button Cancels this change and closes the dialog.

Open Dialog	Selection Dialog (Modal)
-------------	--------------------------

Overview

The debugging environment is restored to the previous environment.

After the file is loaded, the displayed window size and position are restored to their previous states. (The analyzer relationship is not restored.)

This dialog can be opened by the following methods.

- In the main window
Select **File** → **Open/Save Project** → **Open Project...** in the menu bar.
- In the main window
Press in order the **GRPH**, **F**, **J**, and **O** keys.

Window

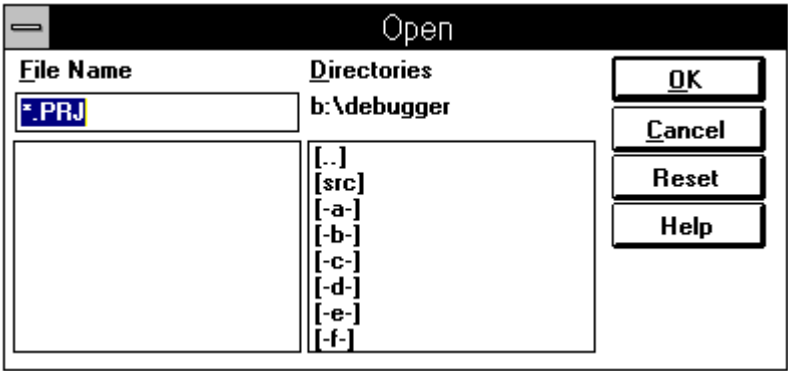


Figure 5-4. Project File Load Dialog

Functions

The project file load dialog contains these items.

- File selection area
- Path setting area
- Function buttons

Each function is described next.

(1) File selection area**File Name**

The image shows a graphical user interface for file selection. At the top, there is a text box labeled 'File Name' containing the text '.PRJ'. Below this text box is a large, empty rectangular area intended for displaying a list of project files.

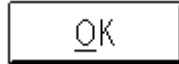
The file name of the project file to be loaded is specified.

The project file is selected by clicking the desired project file in the project file list.

The selected file name is highlighted and displayed in the display area for the selected project file.

The default extension is **.PRJ**.

Double clicking the file name in the project file list is identical to clicking the



button.

(2) Path setting area**Directories**

b:\debugger

The image shows a graphical user interface for path setting. It features a list box containing several directory names: '[..]', '[src]', '[-a-]', '[-b-]', '[-c-]', '[-d-]', '[-e-]', and '[-f-]'. The list box is rectangular and has a thin border.

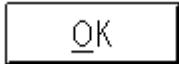
The path of the project file to be loaded is specified.

Double click the desired path name to display the project files in that path in the project file display area.

The display formats are

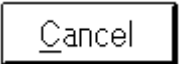
[xxx] : Indicates a directory name

[-x-] : Indicates a drive name

Function Buttons

button

Loads the selected project file and sets the environment.



button

Closes the project file load dialog.



button

Cancels the selections.



button

Opens the help window.

Load Description

The following items are set by loading the project file. However, the target device and the location data are unchanged from when the debugger started.

Window	Setting Data
Configuration dialog	All of the items
Main window	Display position; tool bar, status bar, and button display data; execution mode data; trace on/off data
Load module selection dialog	Download file data
Extended option setting dialog	Setting data
Source path specification dialog	Source path data
Source text window	Window display data, font data
Disassemble window	Window display data, display start address
Memory window	Window display data, display start address
Stack trace window	Window display data
SFR window	Window display data
Local variable window	Window display data
Trace view window	Window display data
Event manager	Window display data, all of the event data
Event link dialog	Window display data
Break dialog	Window display data
Trace dialog	Window display data
Event dialog	Window display data
Register window	Window display data, display bank
Variable window	Window display data, displayed variable data

Save Dialog**Selection Dialog (Modal)****Overview**

The debugging environment is saved.

When saving, the size and position of the displayed window are saved. However, only the active window becomes the save target.

This dialog can be opened by the following methods.

- In the main window

Select **F**ile → **O**pen/Save Project → **S**ave **A**s... in the menu bar.

- In the main window

Press in order the **G**RPH, **F**, **J**, and **A** keys.

If a project file that was previously loaded or saved will be saved with the same file name, the following methods are effective.

- In the main window

Select **F**ile → **O**pen/Save Project → **S**ave... in the menu bar.

- In the main window

Press in order the **G**RPH, **F**, **J**, and **S** keys.

In these methods, the project file save dialog does not open and the save is to the previous file name.

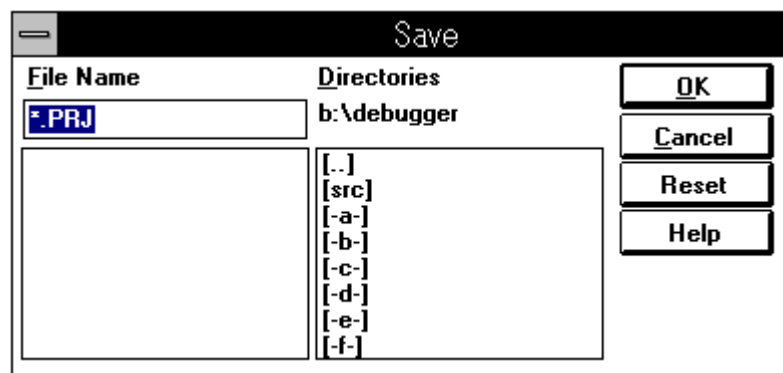
Window

Figure 5-5. Project File Save Dialog

Functions

The project file save dialog contains these items.

- File selection area
- Path setting area
- Function buttons

Each function is described next.

(1) File selection area

File Name



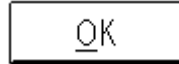
The file name of the project file to be saved is specified.

The project file is selected by clicking the desired project file in the project file list.

The selected file name is highlighted and displayed in the display area for the selected project file.

The default extension is **.PRJ**.

Double clicking the file name in the project file list is identical to clicking the

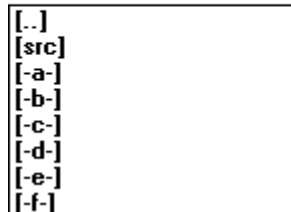


button.

(2) Path setting area

Directories

b:\debugger



The path where the saved project file will be stored is specified.

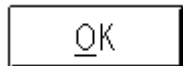
By double clicking the desired path name, the project file in the path is displayed in the project file display area.

The display formats are

[xxx] : Indicates a directory name

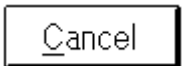
[-x-] : Indicates a drive name

Function Buttons



button

Saves the environment in the selected project file name.



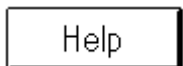
button

Closes the project file save dialog.



button

Cancels the selections.



button

Opens the help window.

Save Description

The following items are saved in the project file.

Window	Setting Data
Configuration dialog	All of the items (target device, clock setting, pin mask setting, mapping data)
Main window	Display position; tool bar, status bar, and button display data; execution mode data, trace on/off data
Load module selection dialog	Download file data
Extended option setting dialog	Setting data
Source path setting dialog	Source path data
Source text window	Window display data, font data
Disassemble window	Window display data, display start address
Memory window	Window display data, display start address
Stack trace window	Window display data
SFR window	Window display data
Local variable window	Window display data
Trace view window	Window display data
Event manager	Window display data, all of the event data
Event link dialog	Window display data
Break dialog	Window display data
Trace dialog	Window display data
Event dialog	Window display data
Register window	Window display data, display bank
Variable window	Window display data, displayed variable data

Load Module Dialog**Selection Dialog (Modal)****Overview**

The name of the file to be downloaded and the file format are selected, and the file is downloaded to the in-circuit emulator or the target.

The file formats that can be downloaded are

- Object file in the load module format (*.LNK)
- Intel extended hexadecimal format (*.HEX)
- Motorola hexadecimal format S-type format (standard address) (*.HEX)
- Extended Tektronix hexadecimal format (*.HEX)

Note that if a file other than an object file in the load module format was loaded, source debugging is not possible. This dialog can be opened by the following methods.

- In the main window
Select **F**ile → **D**own Load... in the menu bar.
- In the main window
Press in order the **GRPH**, **F**, and **D** keys.

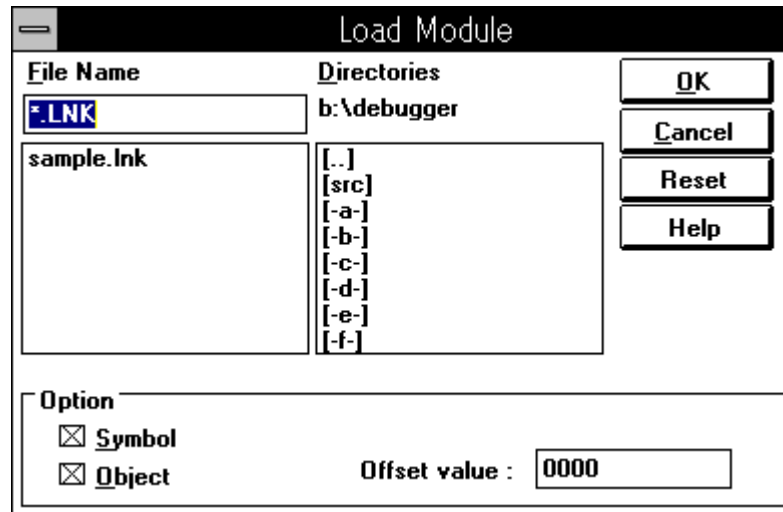
Window

Figure 5-6. Load Module Selection Dialog

Functions

The load module selection dialog contains these items.

- file selection area
- path setting area
- load condition specification area
- function buttons

Each function is described next.

(1) file selection area

File Name

.LNK

sample.lnk

The file name of the load module file to be loaded is specified.

The load module file is selected by clicking the desired load module file in the list of load module files.

The selected file name is highlighted and displayed in the display area for the selected load module file.

The default extension is **.LNK**.

Double clicking the file name in the list of load module files is identical to clicking the **OK** button.

(2) path setting area

Directories

b:\debugger

[..]
[src]
[-a-]
[-b-]
[-c-]
[-d-]
[-e-]
[-f-]

The path of the load module file to be loaded is specified.

By double clicking the desired path name, the load module file in the path is displayed in the load module file display area.

The display formats are

[xxx] : Indicates a directory name

[-x-] : Indicates a drive name

(3) Load condition specification area

Option

☒ **S**ymbol

☒ **O**bject

Offset value : **0000**

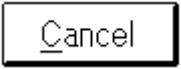
The load conditions are set.

- | | |
|---------------------|--|
| S ymbol | Specifies to read or not read symbol data. |
| O bject | Specifies to read or not read object data. |
| Offset value | Specifies the offset address. |

Function Buttons

button

Reads the selected file.



button

Cancels the changes and closes the dialog.



button

Returns to the initial state.



button

Opens the help window to explain the load module selection dialog.

Upload Dialog**Selection Dialog (Modal)****Overview**

The name of the file to be saved and the file format are set. The memory contents are saved to the file. The format of the file to be saved can be selected from the following three choices.

- Intel extended hexadecimal format (*.HEX)
- Motorola hexadecimal format S-type format (standard address) (*.HEX)
- Extended Tektronix hexadecimal format (*.HEX)

This dialog can be opened by the following methods.

- In the main window
Select **F**ile → **U**p Load... in the menu bar.
- In the main window
Press in order the **G**RPH, **F**, and **U** keys.

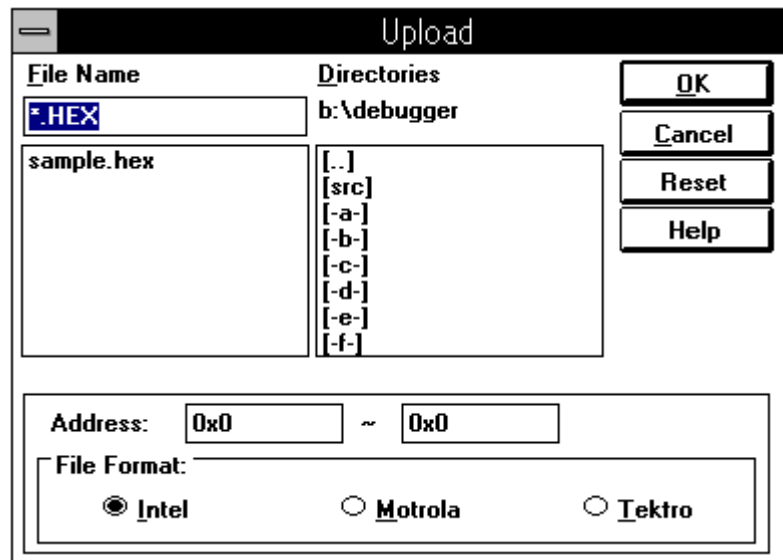
Window

Figure 5-7. Upload Dialog

Functions

This upload dialog contains these items.

- File selection area
- Path setting area
- Upload condition specification area
- Function buttons

Each function is described next.

(1) File selection area

File Name


*.HEX

sample.hex

The name of the object file to be uploaded is specified.

The file name specification is typed in from the keyboard. If a previously saved file will be overwritten, the desired object file can be clicked in the list of object files.

The default extension is **.HEX**.

Double clicking the file name in the list of object files is identical to clicking the  button.

(2) Path setting area

Directories

b:¥debugger

[..
[src]
[-a-]
[-b-]
[-c-]
[-d-]
[-e-]
[-f-]

The path of the object file to be uploaded is specified.

By double clicking the desired path name, the object files in the path are displayed in the object file display area.

The display formats are

[xxx] : Indicates a directory name

[-x-] : Indicates a drive name

(3) Upload condition specification area

Address: ~

File Format:

☒ Intel ☐ Motrola ☐ Tektro

The upload conditions are set.

■ **Address:**

Specifies the address range in memory to be uploaded.

■ **File Format:**

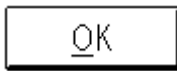
Specifies the file format of the object file to be uploaded. One of the following three choices is selected.

Intel: Intel extended hexadecimal format

Motrola: Motorola hexadecimal format S-type format (standard address)

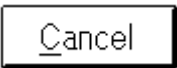
Tektro: Extended Tektronix hexadecimal format

Function Buttons



button

Saves the memory contents in address range to the file in the specified directory, with the file name, and with the file format.



button

Closes the upload dialog.



button

Returns to the initial state.



button

Opens the help window.

Source Path Dialog	Specification Dialog (Modal)
---------------------------	-------------------------------------

Overview

The source path is specified.

By specifying the source path, the source located in multiple directories can be source debugged.

This dialog can be opened by the following methods.

- In the main window
Select **Option** → **Source Path...** in the menu bar.
- In the main window
Press in order the **GRPH**, **P**, and **P** keys.

Window

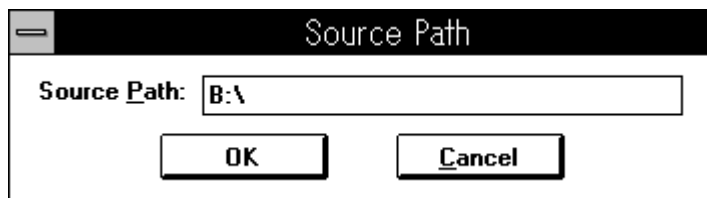


Figure 5-8. Source Path Specification Dialog

Functions

The source path specification dialog contains these items.

- Source path specification area
- Function buttons

Each function is described next.

(1) Source path specification area

Source Path:

The source path is specified.

The delimiter for path information is a space.

A maximum of 256 characters (including half-width characters and delimiters) can be specified in the path information.

Note that specifications other than the above are not possible.

Example: When the source is in the following directories

a:\78k\c
b:\src
c:\asm

The source path is specified by

Source Path:	<input type="text" value="a:\78k\c b:\src c:\asm"/>
---------------------	---

Function Buttons

OK

button

Accepts the source path that was set and closes the dialog.

Cancel

button

Cancels the source path set here and closes the dialog.

Open Dialog	Selection Dialog (Modal)
-------------	--------------------------

Overview

The file to be displayed is selected in the source text window. The following two selection formats can be selected.

- Select by the source file name
- Select by the function name

This dialog can be opened by the following methods when the current window is a source text window.

- In the main window
Select **F**ile → **O**pen... in the menu bar.
- In the main window
Press in order the **GRPH**, **F**, and **O** keys.
- Using shortcut keys
Press **CTRL** + **O**.

Window

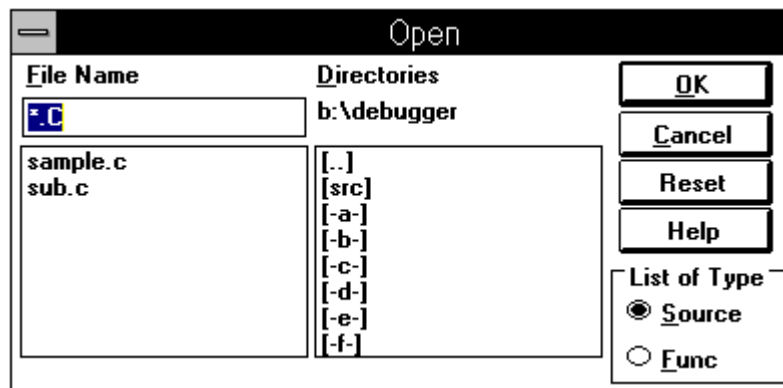


Figure 5-9. Source File Selection Dialog

Functions

The layout of the source file selection dialog changes based on the selection mode.

When the file selection mode was selected

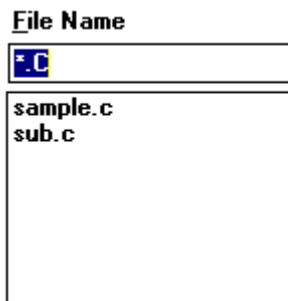
- File selection area
- Path setting area
- Mode selection area
- Function buttons

When the function selection mode was selected

- Function selection area
- Path setting area
- Mode selection area
- Function buttons

Each function is explained next.

(1) File selection area

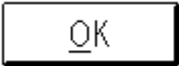


The name of the source file to be displayed in the source text window is selected.

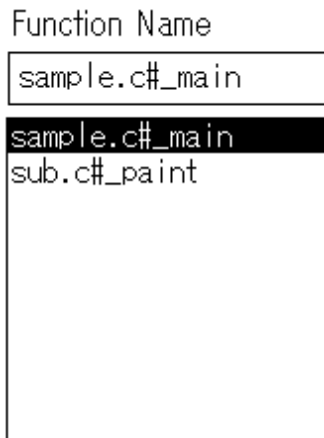
The source file is selected by clicking the desired source file in the list of source files.

The selected file name is highlighted and displayed in the display area for the selected source file.

The default extension is **.C**.

Double clicking the file name in the list of source files is identical to clicking the  button.

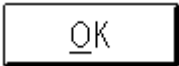
(2) Function selection area



The name of the function to be displayed in the source text window is selected.

The function name is selected by clicking the desired function name in the list of functions.

The selected function name is highlighted and displayed in the display area for the selected function.

Double clicking the function name in the list of functions is identical to clicking the  button.

(3) Path setting area

Directories
b:\debugger

[..]
[src]
[-a-]
[-b-]
[-c-]
[-d-]
[-e-]
[-f-]

The path of the source file to be displayed in the source text window is specified.

By double clicking the desired path name, the source files in the path are displayed in the recorded file display area.

The display formats are

[xxx] : Indicates a directory name

[-x-] : Indicates a drive name

(4) Mode selection area

List of Type
☒ **S**ource
☐ **F**unc

The selection mode is switched.

The following two selection modes can be selected.

Source A source file is the selection target.

Func A function name is the selection target.

Function Buttons

OK

button

The selected source file or function is displayed in the source text window.

Cancel

button

Closes the source file selection dialog.

Reset

button

Returns to the initial state.

Help

button

Opens the help window.


Source Window Dialog

Display Window

Overview

The source text is displayed.

This window can be opened by the following methods.

- In the main window
Select **B**rowse → **S**ource Text... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **S** keys.
- Click the  button in the tool bar.

If you want to display the source from another window, the jump function is useful.

By using the jump function, the target source and source lines can be quickly displayed. The jump function operates as follows after the pointer is selected.

- (1) Select **J**ump → **S**ource Text... in the menu bar.
- (2) Press in order the **GRPH**, **J**, and **S** keys.
- (3) Press the **CTRL** + **U** keys.

The jump functions are listed below.

Window	Pointer	Operating Method		
		(1)	(2)	(3)
Disassemble window	Address display area	○	○	○
Memory window	Address display area	○	○	○
Trace view window	Trace result display area	○	○	○
Stack trace window	Stack frame number display area	○	○	○
Event manager	Event	○	○	—
Register window	Register	○	○	—

Window

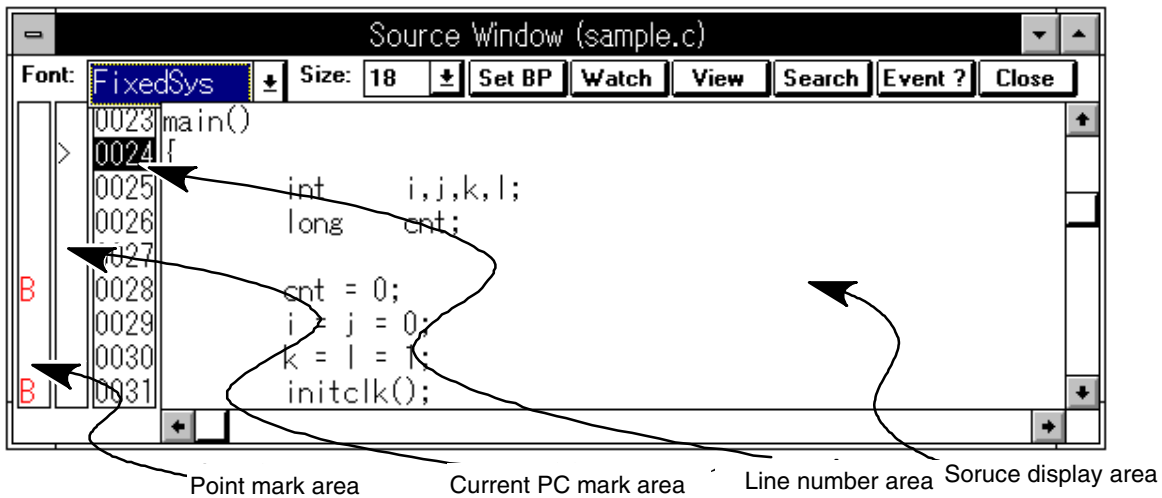


Figure 5-10. Source Text Window

Functions

The source text window contains the following items.

- Display font specification area
- Point mark area
- Current PC mark area
- Line number area
- Source display area
- Function buttons

Each function is described next.

(1) Display font specification area

Font: FixedSys Size: 18

The font and character size of the source text displayed in the source display area are specified in this area.

Font: Specifies the font. (Default: FixedSys)

Size: Specifies the character size.

(2) Point mark area

The point mark area is used to set and delete breakpoints and software breakpoints, and to display the event setting state.

a. Breakpoint set and delete functions

By clicking the mouse in this area, breakpoints can be set and deleted. The following action occurs depending on the location where the mouse is clicked.

Location	Color	Click Button	Operation
'B' mark is displayed.	Red, black	Left click.	Delete the breakpoint.
'B' mark is not displayed or something else is displayed.	—	Left click.	Set the breakpoint
	—	Right click.	Set the software breakpoint.

b. Event display function

The settings of the events are displayed. If an execution event or an access fetch event is set in the corresponding source line, the mark corresponding to the event type is displayed.

Mark	Mark Meaning
E	Indicates that an event condition is set.
L	Indicates that the last phase in the event link is set.
B	Indicates that a break event is set.
T	Indicates that a trace event is set.
A	Indicates that multiple events are set.

(3) Current PC mark area

The '>' mark that points to the current PC value (PC register value) is displayed in the current PC mark area. By continuously pressing the mouse at the position displaying this mark, the PC register value is displayed in a pop-up window.

(4) Line number area

The line numbers of the source text are displayed in the line number area.

This area has five functions in addition to displaying the line number.

a. Come function

This function executes the user program until the selected line.

While executing a user program in this mode, the currently set break events are not generated.

This function is executed by the following operations.

1. The line numbers for the desired breaks are selected.
2. In the main window

Select **Execute** → **Come** in the menu bar, or press in order the **GRPH**, **X**, and **M** keys.

b. Break event setting function

The break event is set at the first address corresponding to the selected line numbers. The set breakpoint uses the execution events.

This function is executed by the following operations.

1. Select the line number where the break event is set.
2. In the main window

Select **E**xecute → **S**et**B**P in the menu bar, or press in order the **GRPH**, **X**, and **B** keys, or press the **CTRL** + **B** shortcut keys.

c. Program counter setting function

The first address corresponding to the selected line numbers is set in the program counter (PC).

This function is executed by the following operations.

1. Select the line numbers you want to set.
2. In the main window

Select **E**xecute → **S**et **P**C in the menu bar, or press in order the **GRPH**, **X**, and **E** keys.

d. Jump function

With the first address corresponding to the selected line numbers as the jump pointer, the disassemble window or the memory window is jumped to. The jump destination window is displayed from the jump pointer.

This function is executed by the following operations.

1. Select the line number.
2. In the main window

When the jump destination is the disassemble window

Select **J**ump → **A**ssemble... in the menu bar, or press in order the **GRPH**, **J**, and **A** keys, or press the **CTRL** + **A** shortcut keys.

e. Window connect function

This function displays the connection relationship with another window (disassemble window, memory window, trace view window) and the source by line number. The line number for the connect target is highlighted.

(5) Source display area

The source text is displayed in the source display area. The symbol can be selected by double clicking or by dragging the symbol to be displayed.

Function Buttons button

Sets a breakpoint in the currently selected source text line.

 button


Opens the variable view window, and displays the value of the specified variable.

 button

Opens the variable view dialog, and displays the variable value.

 button

Opens the search dialog, and searches for the string in the source text.


 button

If marks are set in the selected source text line, the event manager opens and the settings are displayed. If marks are not set, nothing happens.

 button

Closes the source text window.

Icon

The source text window can be displayed as an icon by clicking the  button in the title bar.



Source Window
(sample.c)

Find Dialog	Auxiliary Dialog (Modeless)
-------------	-----------------------------

Overview

A data search is performed.

The search result is reflected in the calling window.

If called from the source text window, the file is searched.

If called from the disassemble window, the disassembled contents are searched.

If called from the main window, the memory is searched.

This dialog can be opened by the following methods.

- In the main window
Select **V**iew → **S**earch... in the menu bar.
- In the main window
Press in order the **GRPH**, **V**, and **S** keys.
- In the source text window
Click the **Search** button.
- In the disassemble window
Click the **Search** button.
- In the memory window
Click the **Search** button.

Window

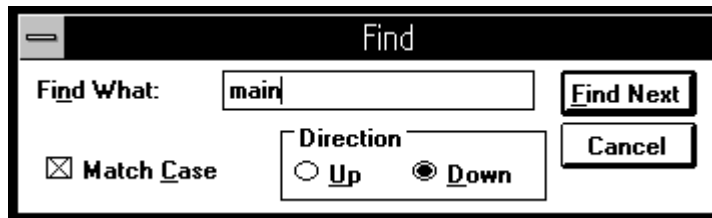


Figure 5-11. Find Dialog

Functions

The search dialog contains these items.

- Search data specification area
- Search condition specification area
- Search direction specification area
- Function buttons

Each function is described next.

(1) Search data specification area

Find What:

The search data is specified.

By default, the string selected in the calling window is displayed, but when needed changes can be typed in from the keyboard.

(2) Search condition specification area

☒ Match Case

The radio buttons specify whether to distinguish between uppercase and lowercase in the specified search data during a search. The default is to distinguish case in a search.

☐ Match Case Do not distinguish.
☒ Match Case Distinguish.

(3) Search direction specification area

Direction
☐ Up ☒ Down


The search direction is specified.


The two search directions are a forward search and a backward search.

Up : Backward search


Down : Forward search


Function Buttons

 button Searches for the specified search data in accordance with the conditions.

 button Stops the data search.

During a data search, the  button changes to a  button.

 button Exits the search dialog.

During a data search, this changes to a  button.

Symbol to Address Dialog**Auxiliary Dialog (Modeless)****Overview**

The address of the specified variable is displayed.

This dialog can be opened by the following methods.

- In the main window
Select the **View** → **Sym To Adr...** in the menu bar.
- In the main window
Press in order the **GRPH**, **V**, and **Y** keys.

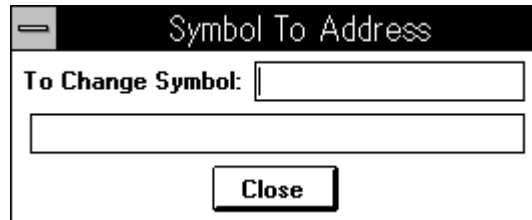
Window

Figure 5-12. Change Symbol Dialog

Functions

The change symbol dialog contains these items.

- Variable specification area
- Variable address display area
- Function buttons

Each function is described next.

(1) Variable specification area

To Change

The variable name and line number for the address conversion are specified.

After the data is input, press the return key to display the address value in the variable address display area.

The specification method is shown below.

Function and Variable	<u>_</u> fnc file#_fnc (for a static function and variable)
SFR	sfrname
Line number of the source text	file:no

fnc: function, variable name; sfrname: SFR name; file: file name; no: line number

If a function or a variable name is specified, specify with an underline (_) added at the beginning.

The sharp (#) is used as the separator between a file name and a function or variable name. The colon (:) is the separator between a file name and a line number.

(2) Variable address display area

This area displays the address of the variable specified in the variable specification area.

Function Buttons

button

Closes the dialog.

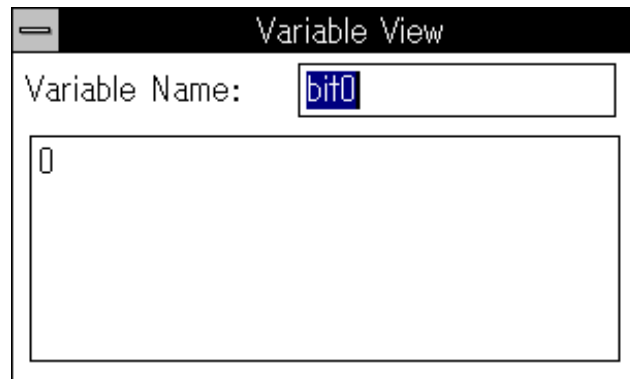
Variable View Dialog

Auxiliary Dialog (Modal)

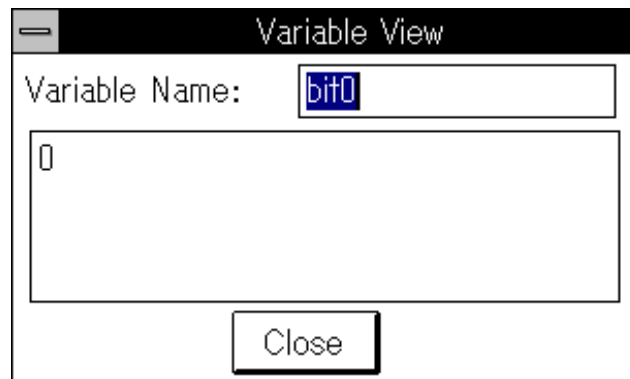
Overview

The value of the variable specified in the source text window is temporarily displayed. This dialog can be opened by the following methods.

- Select the variable in the source text window.
Select the **View** → **View Variable...** in the menu bar.
- Select the variable in the source text window.
Press in order the **GRPH**, **V**, and **V** keys.
- Select the variable in the source text window.
Press the **View** button in the source text window.

Window

When opened by a button in the source text window



When opened from the menu bar

Figure 5-13. Variable View Dialog

Functions

This variable view dialog contains these items.

- Variable specification area
- Variable value display area
- Function buttons

Each function is described next.

(1) Variable specification area

Variable Name: 

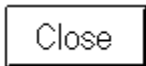
The default is to specify the variable name selected in the source text window.

If you want to display another variable, the variable name can be displayed by typing it in from the keyboard.

(2) Variable value display area

The variable value specified in the variable specification area is displayed.

Function Buttons

 button

Closes the dialog.

Variable Window Dialog

Display/Setting Window

Overview

The value of the variable specified in the source text window is displayed and changed.
This window can be opened by the following methods.

- In the main window
Select **Browse** → **Variable...** in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **V** keys.
- Select the variable in the source window.
Select **View** → **Watch Variable...** in the menu bar.
- Select the variable in the source window.
Press in order the **GRPH**, **V**, and **W** keys.
- Select the variable in the source window.
Press the **Watch** button.

Window

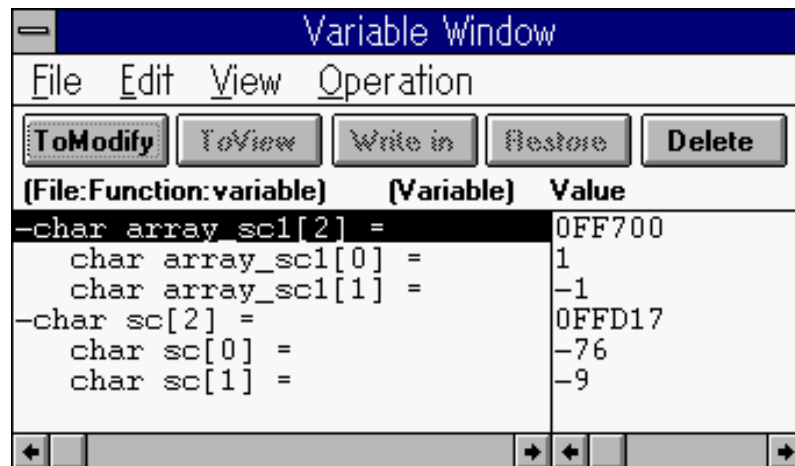


Figure 5-14. Variable Window

Functions

Variables are displayed and changed.

The variable display is added for each display requirement. If the same variable is added, the addition is not displayed.

This window has a view mode and a modify mode.

This variable window contains these items.

- Menu bar
- Function button
- Variable name display area
- Variable value display/setting area

Each function is described next.

(1) Variable name display area

(File:Function:variable)	(Variable)
+timecnt =	
-timedsp =	
unsigned char exul =	
+unsigned char hore[2] =	
unsigned char hcoron =	
+unsigned char minute[2] =	
unsigned char mpoint =	

This area displays the variable names.

The variables displayed with a "+" at the beginning are pointer variables. By double clicking a pointer variable, the data value indicated by the pointer is displayed in the variable value display/setting area. The "+" display switches to a "-" display.

(2) Variable value display/setting area

Value
{...}
{}
252
FE81
255
FE84
0

This area displays the variable values.

When the variable is a pointer variable, the address value or data value is displayed.

Function Buttons

ToModify button

Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the variable value can be changed.

When the modify mode is entered, the background color of the window changes, and this button can no longer be selected.

A variable value is changed by clicking the variable value. After the text cursor is displayed, the change can be typed in from the keyboard. The changes are fixed by clicking the **Write in** button.

ToView button

Switches to the view mode.

This button can only be selected when the window is in the modify mode. The view mode can be moved to by clicking this button.

When the view mode is entered, the background color of the window changes, and this button can no longer be selected.

Write in button

Writes in the changes.

Restore button

Cancels the changes.

All of the values changed in the modify mode are restored to their original values.

However, if the **Write in** button has already been clicked, the values in a later state are restored.

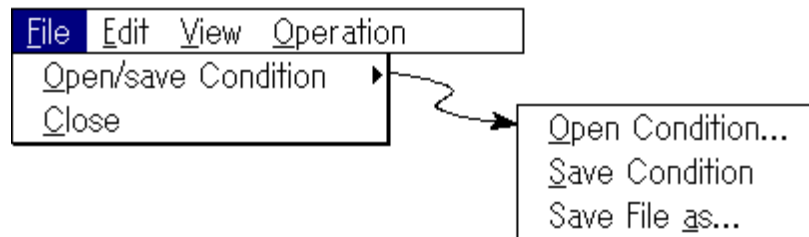
Delete button

The specified variable is deleted from the variable view window.

Menu Bar

Clicking the mouse in the menu bar displays a pull-down menu.

(a) File



Open/Save Condition ► Loads and saves variable values.

Open Condition

Opens the selected file for reference. The view file selection dialog is opened.

Save Condition

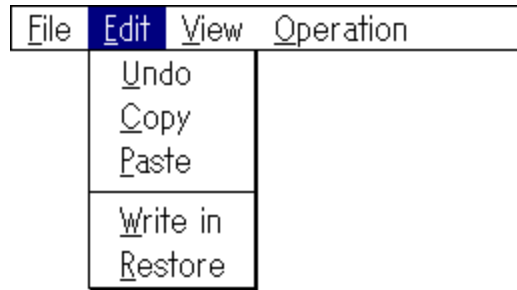
Saves the window contents in the view file.

Save File As...

Saves the window contents in the view file. The view file selection dialog is opened.

Close

Closes the variable view window.

(b) EditUndo

Undoes the previous editing operation.

Copy

Copies the selected string to the clipboard buffer.

Paste

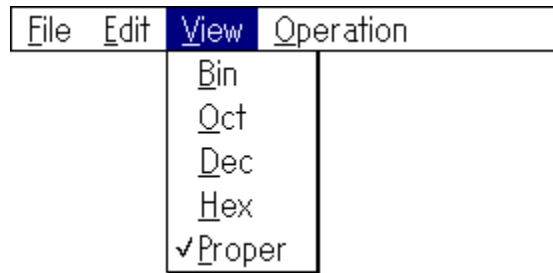
Pastes the contents of the clipboard buffer.

Write in

Writes the changes.

Restore

Cancels the changes.

(c) ViewBin

Displays the variables in binary.

Oct

Displays the variables in octal.

Dec

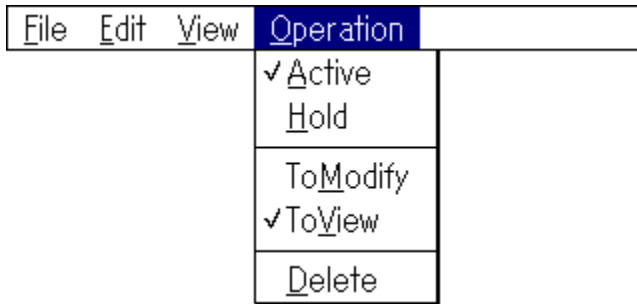
Displays the variables in decimal.

Hex

Displays the variables in hexadecimal.

Proper

Displays the variables using appropriate values.

(d) Operation

<u>A</u>ctive	Switches the variable window to the active state.
<u>H</u>old	Switches the variable window to the hold state.
To<u>M</u>odify	Switches the variable window to the modify mode.
To<u>V</u>iew	Switches the variable window to the view mode.
<u>D</u>elete	Deletes the selected variable from the variable window.

Add Variable Dialog

Auxiliary Dialog (Modeless)

Overview

The variable displayed in the variable window is added and registered.
This dialog can be opened by the following methods.

- In the main window
Select **V**iew → **A**dd **V**ariable... in the menu bar.
- In the main window
Press in order the **GRPH**, **V**, and **I** keys.

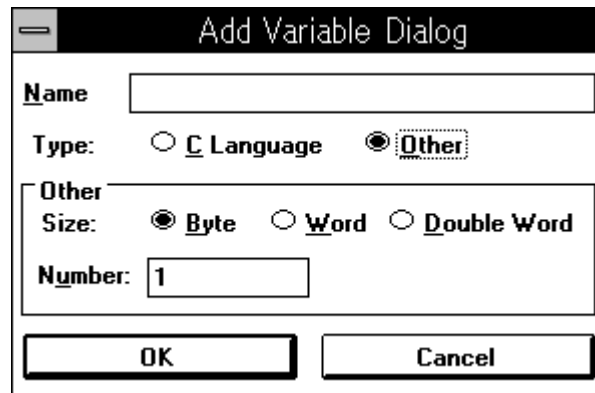
Window

Figure 5-15. Add Variable Dialog

Functions

The add variable dialog contains these items.

- Variable name specification area
- Variable type selection area
- Variable size specification area
- Function buttons

Each function is described next.

(1) Variable name specification area

Name

The variable name to be added is specified.

Variables	<u>_</u> fnc file#_fnc
SFR	sfrname

fnc: function, variable name; sfrname: SFR name; file: file name

When a variable name is specified, specify with an underline (_) added at the beginning.

The separator between a file name and a variable name is the sharp (#).

(2) Variable type selection area

Type: ☐ C Language ☒ Other

The type of language of the variable specified in the variable name specification area is selected.

C Language	Variable registered in the C language
Other	Variable registered in a language other than the C language (SFR, assembler variable)

(3) Variable size specification area

Other

Size: ☒ Byte ☐ Word ☐ Double Word

Number:

The size and number of the added variable are specified. Selecting the C language in the variable type selection area cannot be specified.

a. Size

This specifies the variable size. These three types can be selected.

Byte

Word

Double Word

b. Number

This specifies the number of variables.

Function ButtonsA rectangular button with a black border and the text "OK" in bold black font.

button

The variable is added and registered to the variable window.

A rectangular button with a black border and the text "Cancel" in bold black font.

button

Closes the dialog.

Local Variable Window**Display/Setting Window****Overview**

The local variables in the current function are displayed and changed.

This window can be opened by the following methods.

- In the main window
Select **Browse** → **Local Variable...** in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **L** keys.

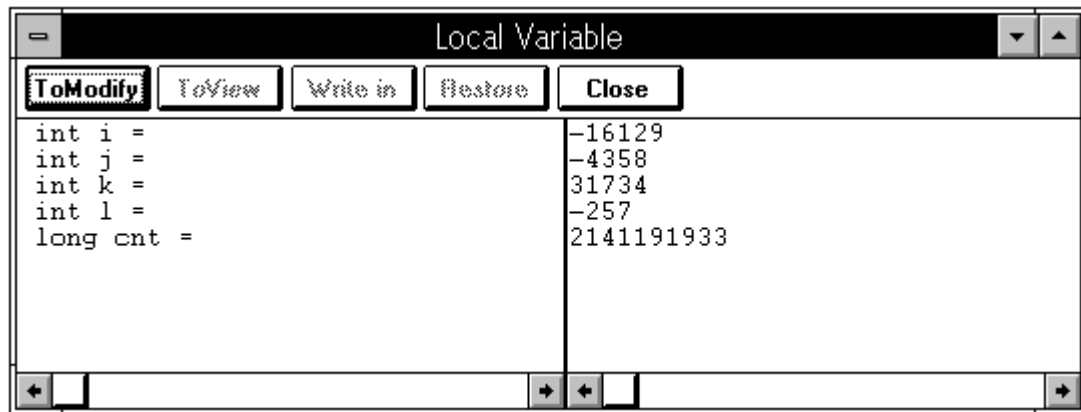


Window

Figure 5-16. Local Variable Window

Functions

Local variables are displayed and changed.

This window automatically displays the local variables in the current function. An added display of variables is not possible.

The boundary line between the local variable name display area and the local variable value display/setting area can be moved by the mouse. The boundary line can be moved by dragging and dropping when the mouse cursor has changed from  to .

This window has a view mode and a modify mode.

The local variable window contains the following items.

- Local variable name display area
- Local variable value display/setting area
- Function buttons

Each function is described next.

(1) Local variable name display area

This area displays the local variable names.

```
int i =
int j =
int k =
int l =
long cnt =
```

A variable displayed with a "+" at the beginning indicates a pointer variable. By double clicking a pointer variable, the data value pointed to by the pointer is displayed in the variable value display/modify area. The "+" display switches to the "-" display.

(2) Local variable value display/setting area

```
-16129
-4358
31734
-257
2141191933
```

This area displays local variable values.

When the variable is a pointer variable, the address value or the data value is displayed.

Function Buttons

ToModify button

Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the variable value can be changed.

When in the modify mode, the window's background color changes, and this button can no longer be selected.

A variable value can be changed by typing in from the keyboard after clicking the variable value to display the text cursor. The changes are fixed by clicking the

Write in button.

ToView button

Switches to the view mode.

This button can be selected only when the window is in the modify mode. By clicking this button, the view mode can be moved to.

When the view mode is entered, the window's background color changes, and this button can no longer be selected.


Write in button

Writes the changes.



button

Cancels the changes.


All of the changed values in the modify mode are restored to their original values. However, if the  button has already been clicked, the values in a later state are restored.



button

Deletes the specified variable from the variable view window.



The local variable window can be displayed as an icon by clicking the  button in the title bar.



Address Specification Dialog


Setting Dialog (Modal)

Overview


The display starting addresses of the memory display and the disassemble display are specified.

This dialog can be opened by the following methods.

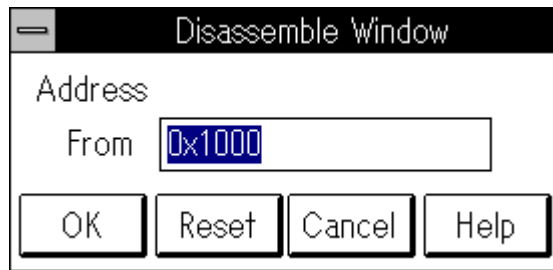
For the disassemble window

- In the main window
Select **B**rowse → **A**ssemble... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **A** keys.
- Press the  button in the tool bar.

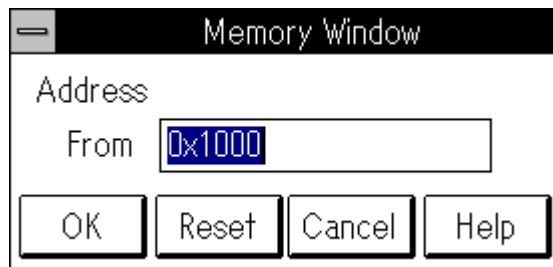
For the main window

- In the main window
Select **B**rowse → **M**emory... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **M** keys.
- Press the  button in the tool bar.

Window



Disassemble window address specification



Memory window address specification

Figure 5-17. Address Specification Dialog

Functions

The display starting address is specified.

The address specification dialog contains the following items.

- Address specification area
- Function buttons

Each function is described next.

(1) Address specification area

Address

From

The address is specified in this area. The default is to specify the current PC value, but when needed changes can be typed in from the keyboard. The address specification can be specified by symbols. The specification methods are shown below.

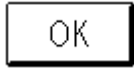
Function and Variable	_fnc file#_fnc (for static function and variable)
Line number in source text	file:no


fnc: function, variable name; file: file name; no: line number

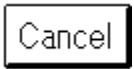
If a function or variable name is specified, an underline (_) is specified at the beginning.

The separator between the file name and the function or variable name is the sharp (#). The separator between the file name and the line number is the colon (:).

Function Buttons

	button	Displays the memory from the specified address and displays the disassembly result.
---	--------	---

	button	Restores the address value to its default.
---	--------	--

	button	Closes the address specification dialog.
---	--------	--


	button	Opens the help window.
---	--------	------------------------

Disassemble Window

Display/Setting Window

Overview

The disassembly result of the program is displayed, and on-line assembly is performed.
This window can be opened by the following methods.

- In the main window
Select **B**rowse → **A**ssemble... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **A** keys.
- Click the  button in the tool bar.

If you want to display the corresponding assemble line from another window, the jump function is useful. By using the jump function, the target assemble line can be quickly displayed. The jump function is operated in the following ways after the pointer is selected.

- (1) Select **J**ump → **A**ssemble... in the menu bar.
- (2) Press in order the **GRPH**, **J**, and **A** keys.
- (3) Press the **CTRL** + **A** keys.

The jump functions are listed below.

Window	Pointer	Operating Method		
		(1)	(2)	(3)
Source text window	Line number area	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Memory window	Address display area	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Trace view window	Trace result display area	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stack trace window	Stack frame number display area	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Event manager	Event	<input type="radio"/>	<input type="radio"/>	–
Register window	Register	<input type="radio"/>	<input type="radio"/>	–

Window

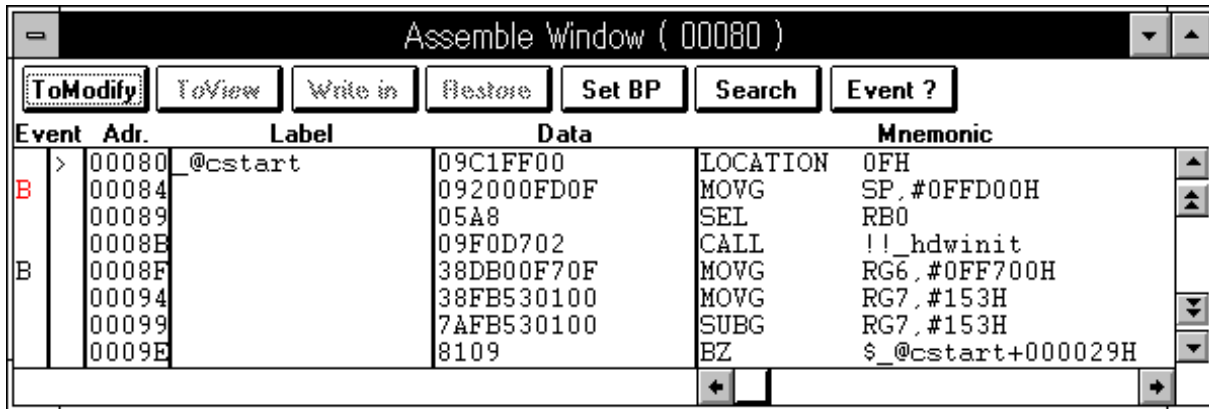


Figure 5-18. Disassemble Window

Functions

The disassembly result is display, and on-line assembly is performed.

This window has a view mode and a modify mode.

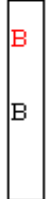
The disassemble window contains the following items.

- Point mark area
- Current PC mark area
- Address display area
- Label display area
- Data display area
- Mnemonic display/modify area
- Function buttons

Each function is described next.

(1) Point mark area

Event



Breakpoints are set and deleted, and events are displayed in the point mark area.

a. Breakpoint set/delete function

By clicking the mouse, breakpoints can be set and deleted in this area. The locations for clicking the mouse are as follows.

Location	Color	Click Button	Operation
'B' mark is displayed.	Red, black	Left click.	Delete the breakpoint.
'B' mark is not displayed or something else is displayed.	—	Left click.	Set the breakpoint.
	—		

b. Event display function

The setting states of various events are displayed. If an execution event or an access fetch event is set at the corresponding assemble line, the mark corresponding to the event type is displayed.

Mark	Mark Meaning
E	Indicates that an event condition is set.
L	Indicates that the last phase in the event link is set.
B	Indicates that a break event is set.
T	Indicates that a trace event is set.
A	Indicates that multiple events are set.

(2) Current PC mark area

The '>' mark pointing to the current PC value (PC register value) is displayed in the current PC mark area. By continuously pressing the mouse at the position displaying this mark, the PC register value is displayed in a pop-up window.

(3) Address display area

Adr.
 00080
 00084
 00088
 0008C
 0008F
 00094
 00098
 0009C

The disassemble starting address is displayed in the address display area.

This area has five functions in addition to displaying the address.

a. Come function

This function executes the user program up to the selected address.

While executing a user program in this mode, the currently set break events are not generated.

This function is executed by the following operations.

1. Select the address for the desired break.
2. In the main window

Select **Execute** → **Come** in the menu bar, or press in order the **GRPH**, **X**, and **M** keys.

b. Break event setting function

A breakpoint is set at the selected address. The set breakpoint uses an execution event.

This function is executed by the following operations.

1. Select the address where the breakpoint will be set.
2. In the main window

Select **Execute** → **SetBP** in the menu bar, or press in order the **GRPH**, **X**, and **B** keys, or press the **CTRL** + **B** shortcut keys.

c. Program counter setting function

The selected address is set in the program counter (PC).

This function is executed by the following operations.

1. Select the address you want to set.
2. In the main window

Select **Execute** → **Set PC** in the menu bar, or press in order the **GRPH**, **X**, and **E** keys.

d. Jump function

With the selected address as the jump pointer, the source text window or the memory window is jumped to. The jump destination window is displayed from the jump pointer.

This function is executed by the following operations.

1. Select the address.
2. In the main window

When the jump destination is the source text window

Select **Jump** → **Source Text...** in the menu bar, or press in order the **GRPH**, **J**, and **S** keys, or press the **CTRL** + **U** shortcut keys.

e. Window connect function

This function represents the connection relationship between another window (source text window, memory window, trace view window) and the disassemble display by the address. The target connect address is highlighted.

(4) Label display area

This area displays labels.

(5) Data display area

Data
09C1FF00
092000FD0F
05A8
09F0D702
38DB00F70F
38FB530100
7AFB530100
8109

This area displays mnemonic data.

(6) Mnemonic display/modify area

Mnemonic	
LOCATION	0FH
MOVG	SP, #0FFD00H
SEL	RB0
CALL	!!_hdwinit
MOVG	RG6, #0FF700H
MOVG	RG7, #153H
SUBG	RG7, #153H
BZ	\$_@cstart+000029H

The disassemble result is displayed.

When in the modify mode, this area is directly rewritten.

If a mnemonic changed in the modify mode is longer than the original mnemonic at the specified address, the next mnemonic is corrupted. In addition, if shorter than the original mnemonic, note that the next mnemonic becomes an illegal mnemonic.

Function Buttons

ToModify button

Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the mnemonic can be changed.

When the modify mode is entered, the window's background color changes, and this button can no longer be selected.

To change a mnemonic, click the mnemonic you want to change. After the text cursor is displayed, changes can be typed in from the keyboard. The changes are fixed by clicking the **Write in** button.

ToView button

Switches to the view mode.

This button can be selected only when the window is in the modify mode. By clicking this button, the view mode can be moved to.

When the view mode is entered, the window's background color changes, and this button can no longer be selected.

Write in button

Writes in the changes.

Restore button

Cancels the changes.

All of the values changed in the modify mode are restored to their original values.

However, if the **Write in** button has already been clicked, the values in a later state are restored.



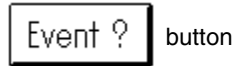
button

Sets a breakpoint at the selected assemble line.



button


Opens the search dialog to search for a mnemonic string.



button

If an event mark is set in the selected assemble line, the event manager related to that point is opened, and the settings are displayed. If a mark is not set, nothing happens.

Icon

The assemble window can be displayed as an icon by clicking the  button in the title bar.




Assemble
Window

Memory Window**Display/Setting Window****Overview**

The memory contents are display and changed.

This window can be opened by the following methods.

- In the main window
Select **B**rowse → **M**emory... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **M** keys.
- Click the  button in the tool bar.

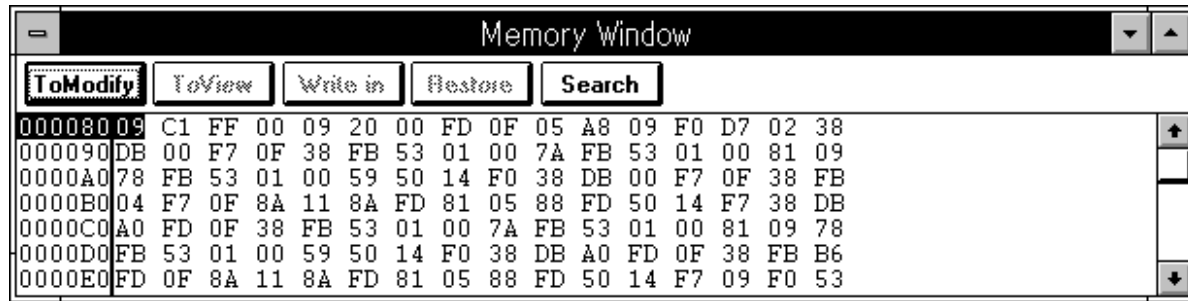
If you want to display the corresponding memory contents from another window, the jump function is useful. By using the jump function, the target memory contents can be quickly displayed. The jump function is operated in the following ways after the pointer is selected.

- (1) Select **J**ump → **M**emory... in the menu bar.
- (2) Press in order the **GRPH**, **J**, and **M** keys.
- (3) Press the **CTRL** + **M** keys.

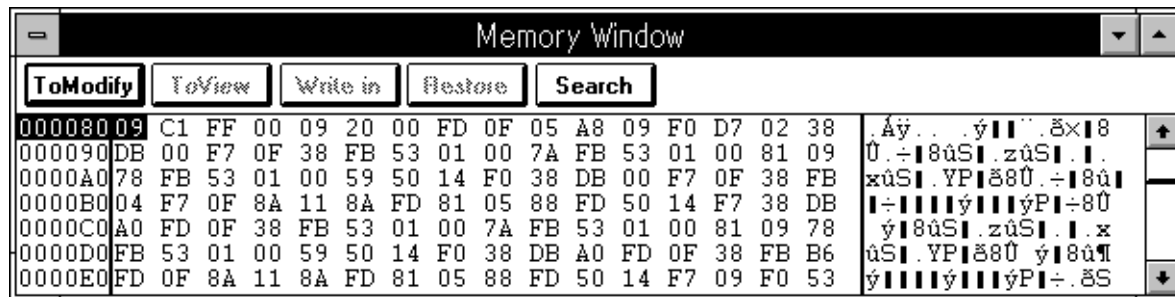
The jump function is listed below.

Window	Pointer	Operating Method		
		(1)	(2)	(3)
Source text window	Line number area	○	○	○
Disassemble window	Address display area	○	○	○
Trace view window	Trace result display area	○	○	○
Event manager	Event	○	○	—
Register window	Register	○	○	—

Window



No ASCII display



ASCII display

Figure 5-19. Memory Window

Functions

The memory contents are displayed and changed.
 This window has a view mode and a modify mode.
 The memory window contains the following items.

- Address display area
- Memory display area
- ASCII display area
- Function buttons

Each function is described next.

(1) Address display area

```

000080
000090
0000A0
0000B0
0000C0
0000D0
0000E0

```

This area displays the memory addresses.

This area has two functions in addition to displaying the memory address.

a. Jump function

With the selected address as the jump pointer, the source text window or the disassemble window is jumped to. The jump destination window is displayed from the jump pointer.

This function is executed by the following operations.

1. Select the address.
2. In the main window

When the jump destination is the source text window

Select **J**ump → **S**ource Text... in the menu bar, or press in order the **G**RPH, **J**, and **S** keys, or press the **CTRL**+**U** shortcut keys.

b. Window connect function

This function represents the connection relationship between another window (source text window, assemble window, trace view window) and the memory display by the memory address. The target connect address is highlighted.

(2) Memory display area

```

09 C1 FF 00 09 20 00 FD 0F 05 A8 09 F0 D7 02 38
DB 00 F7 0F 38 FB 53 01 00 7A FB 53 01 00 81 09
78 FB 53 01 00 59 50 14 F0 38 DB 00 F7 0F 38 FB
04 F7 0F 8A 11 8A FD 81 05 88 FD 50 14 F7 38 DB
A0 FD 0F 38 FB 53 01 00 7A FB 53 01 00 81 09 78
FB 53 01 00 59 50 14 F0 38 DB A0 FD 0F 38 FB B6
FD 0F 8A 11 8A FD 81 05 88 FD 50 14 F7 09 F0 53

```

The memory contents are displayed in this area.

In the modify mode, the memory contents can be changed.

(3) ASCII display area

```

.Äÿ...ÿIII.ŠxI8
0.÷I8ûS|.zûS|.I.
xûS|.YP|Š8Û.÷I8ûI
I÷IIIIÿIIIIÿPI÷8Û
ÿI8ûS|.zûS|.I.x
ûS|.YP|Š8Û ÿI8ûI
ÿIIIIÿIIIIÿPI÷.ŠS

```

The memory contents are displayed in ASCII in this area.

In the modify mode, the memory contents can be changed by ASCII characters.


By selecting **V**iew → **M**emory → **A**scii in the menu bar, the display can be switched on and off.

Function Buttons button


Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the memory contents can be changed.

When the modify mode is entered, the window's background color changes, and this button can no longer be selected.

To change the memory contents, click the memory display position you want to change. After the text cursor is displayed, changes can be typed in from the keyboard. The changes are fixed by clicking the  button.


If the ASCII display area is displayed, ASCII characters can be changed in the ASCII display area.

 button


Switches to the view mode.

This button can be selected only when the window is in the modify mode. By clicking this button, the view mode can be moved to.

When the view mode is entered, the window's background color changes, and this button can no longer be selected.


 button

Writes in the changes.

 button

Cancels the changes.

All of the values changed in the modify mode are restored to their original values.

However, if the  button has already been clicked, the values in a later state are restored.

 button

Opens the search dialog to search for a string in the memory display contents.

Icon

The memory window can be displayed as an icon by clicking the  button in the title bar.



**Memory
Window**

Memory Fill Dialog

Auxiliary Dialog (Modal)

Overview

The memory contents are initialized to the specified code.

This dialog is in the active window state and can be opened by the following methods.

- In the main window
Select **E**dit → **M**emory → **M**emory **F**ill... in the menu bar.
- In the main window
Press in order the **GRPH**, **E**, **M**, and **F** keys.

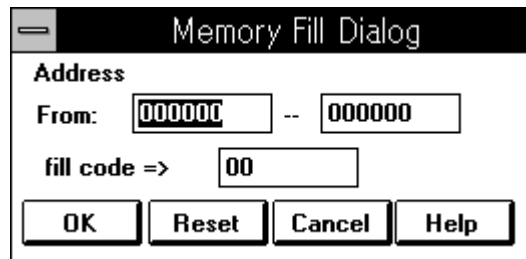
Window

Figure 5-20. Memory Fill Dialog

Functions

The memory contents are initialized.

The memory fill dialog contains the following items.

- Address range specification area
- Data specification area
- Function buttons

Each function is described next.

(1) Address range specification area**Address**

From: --

The address range for the memory contents to be initialized is specified.

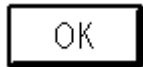
The input is from to .

(2) Data specification area

fill code =>

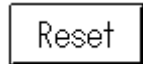
The initialization data is specified.

The data can be specified as up to 16 bytes of string data.

Function Buttons

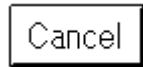
button

Initializes the memory.



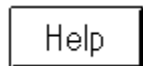
button

Restores the input data to the original value.



button

Closes the memory fill dialog.



button

Opens the help window.

Memory Copy Dialog**Auxiliary Dialog (Modal)****Overview**

The memory is copied.

This dialog is in the active window state and can be opened by the following methods.

- In the main window
Select **E**dit → **M**emory → **M**emory **C**opy... in the menu bar.
- In the main window
Press in order the **G**RPH, **E**, **M**, and **C** keys.

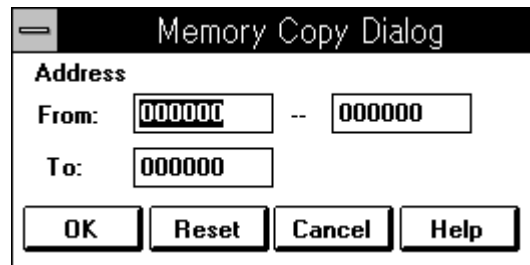
Window

Figure 5-21. Memory Copy Dialog

Functions

The memory contents are initialized.

The memory copy dialog contains the following items.

- Address range specification area
- Function buttons

Each function is described next.

(1) Address range specification area**Address****From:** -- **To:**

The addresses of the copy source and copy destination for the memory contents are specified.

From: The address range of the copy source is specified.

Input in order – .

To: The starting address of the copy destination is specified.

Function Buttons

button

Copies the memory.

button

Restores the input data to its original values.

button

Closes the memory copy dialog.

button

Opens the help window.

Memory Compare Dialog

Auxiliary Dialog (Modal)

Overview

Memories are compared.

This dialog is in the active window state and can be opened by the following methods.

- In the main window
Select **E**dit → **M**emory → **M**emory **C**ompare... in the menu bar.
- In the main window
Press in order the **GRPH**, **E**, **M**, and **P** keys.

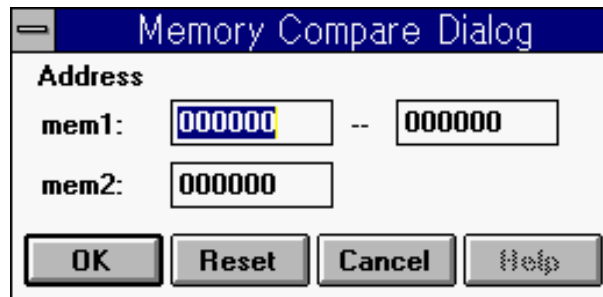
Window

Figure 5-22. Memory Compare Dialog

Functions

The contents of memories are compared.

The memory compare dialog contains the following items.

- Comparison range specification area
- Function buttons

Each function is described next.

(1) Comparison range specification area

Address	
mem1:	<input type="text" value="000000"/> -- <input type="text" value="000000"/>
mem2:	<input type="text" value="000000"/>

The comparison source address and the comparison destination address of the memory contents are specified.

mem1: Specifies the address range of the comparison source.

Input in order – .

mem2: Input the address of the compare destination.

Function Buttons



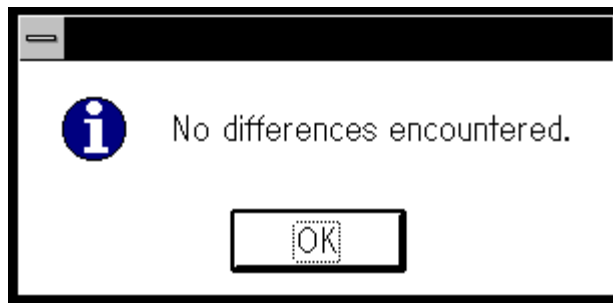
button Compares the memories.


Memory comparison result

If there are no differences, the confirmation dialog appears.

If there are differences, the memory comparison result dialog opens.

The result of the memory comparison displays the following confirmation dialog when there are no differences.



By pressing the  button, the memory comparison ends.



button The input data are restored to their original values.



button Closes the memory compare dialog.



button Opens the help window.

Memory Compare Dialog

Display Dialog (Modal)

Overview

The result of a memory comparison is displayed.

This dialog is displayed when the result of the memory comparison in the memory compare dialog shows differences in the memory contents. If there were no differences, this dialog does not appear, and the confirmation dialog is displayed.

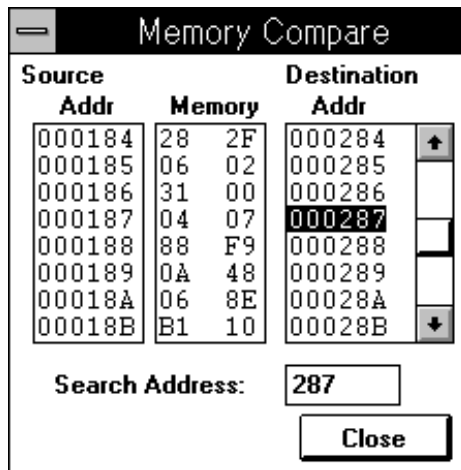
Window

Figure 5-23. Memory Comparison Result Dialog

Functions

The memory contents are compared.

The memory compare result dialog contains the following items.

- Comparison result display area
- Address search specification area
- Function buttons

Each function is described next.

(1) Comparison result display area

Source		Destination	
Addr	Memory	Addr	
000184	28 2F	000284	↑
000185	06 02	000285	
000186	31 00	000286	
000187	04 07	000287	
000188	88 F9	000288	
000189	0A 48	000289	
00018A	06 8E	00028A	
00018B	B1 10	00028B	↓

The memory comparison result is displayed.

Source Addr

This area displays the compare source address where there was a comparison error.

Memory

This area displays the data where there was a comparison error. The comparison source data are displayed on the left. The comparison destination data are displayed on the right.

Destination Addr

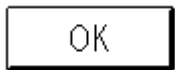
This area displays the comparison destination address where there was a comparison error.

(2) Address search specification area

Search Address:

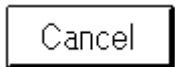
Address containing the comparison error is searched for. If the specified address was found, it is displayed in the comparison result display area.

The search method can be to search for the address by entering it at the keyboard. (After the address is input, the return key does not have to be pressed.)

Function Buttons

button

Closes the memory comparison result dialog. The address in Search Address: that was searched for is highlighted in the memory window.



button


Closes the memory comparison result dialog.

Stack Trace Window	Display Window
---------------------------	-----------------------

Overview

The stack of the current user program is displayed.

This window can be opened by the following methods.

- In the main window
Select **B**rowse → **S**tack Trace... in the menu bar.
- In the main window
Press in order the **G**RPH, **B**, and **K** keys.
- Click the  button in the tool bar.

Window

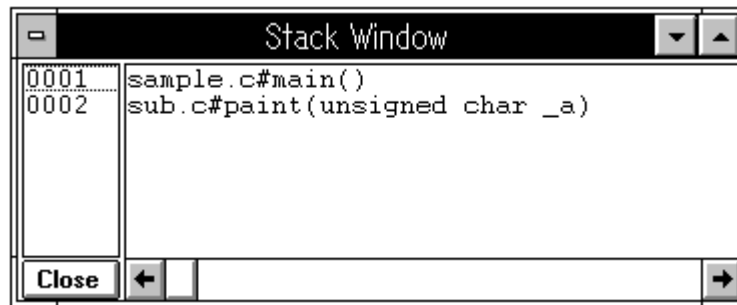


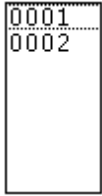
Figure 5-24. Stack Trace Window

Functions

The stack trace window contains the following items.

- Stack frame number display area
- Stack contents display area
- Function buttons

Each function is described next.

(1) Stack frame number display area

A number is assigned to the stack contents and displayed in this area.

The stack frame number is a natural number starting at 1 and becomes larger as the nesting of the stack becomes shallow. In other words, a function with a stack number which is one less than the stack number for some function becomes the calling function of the function.

In addition to displaying the stack frame number, this area has the following functions.

a. Jump function

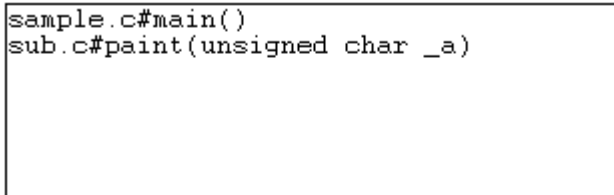
With the starting address of the function pointed to by the selected stack frame number as the jump pointer, the source text window or the disassemble window is jumped to. The jump destination window is displayed from the jump pointer.

This function is executed by the following operations.

1. Select the stack frame number.
2. In the main window

When the jump destination is the source text window

Select **Jump → Source Text...** in the menu bar, or press in order the **GRPH**, **J**, and **S** keys, or press the **CTRL** + **U** shortcut keys.

(2) Stack contents display area

The stack contents are displayed in this area.


The display contents is [file-name#function-name(parameters)]. The separator between a file name and a function name is indicated by (#).

Function Buttons

button

Closes the stack trace window.

Icon

The stack trace window can be displayed as an icon by clicking the  button in the title bar.



Stack Window

Cautions

The stack trace display function is not properly displayed when the function (such as the `noauto` or `norec` function) does not push the frame pointer on the stack or the `-qf` options are included as the optimization options.

Event Set Dialog	Setting Dialog (Modeless)
-------------------------	----------------------------------

Overview

The event conditions are registered and displayed.

The event conditions registered in this dialog are automatically registered in the event manager.

This dialog can be opened by the following methods.

- In the main window
Select **B**rowse → **E**vent → **E**ventSet... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, **E**, and **E** keys.
- In the event manager
Select **O**peration → **E**ventSet... in the menu bar.
- In the event manager
Press in order the **GRPH**, **O**, and **E** keys.

Window

The Event Set dialog box contains the following elements:

- Title Bar:** Event Set
- Buttons:** Restore Evnt, Make Evnt, Close
- Event Name:** A text field with a red 'E' icon, containing the text "**New**", and a dropdown arrow.
- External Trigger:** A checkbox labeled "External Trigger".
- Address:** Two text boxes separated by a hyphen.
- Mask:** A text field containing "0000".
- Status:** A text field containing "Run" and a dropdown arrow.
- Data:** A text field.
- Mask:** A text field containing "FFFF".
- Data Size:** A text field containing "Byte" and a dropdown arrow.
- Pass count:** A slider with left and right arrows, and a text field containing "1".

Figure 5-25. Event Set Dialog

Functions

The event conditions are registered and displayed.

A maximum of 32,767 events can be registered as the event conditions. However, the number of events that can actually be used in a break, timer, and tracer is nine or fewer points (4 execution event points, 4 access event points, and 1 external trigger point).

The events that can be simultaneously used have nine points and can be set in multiple event conditions, such as breaks, tracers, and event links.

An event dialog contains the following items.

- Event name setting area
- Address setting area
- Status selection area
- Data setting area
- Data size selection area
- Pass count setting area
- External trigger condition setting area
- Function buttons

Each function is described next.

(1) Event name setting area

Event Name:  

The event name is set and selected. "***NEW**" is displayed by default.

By pressing the  button in the event name selection, the selection is made from a drop-down list.

An event name with a maximum of eight letters can be set.

(2) Address setting area

Address - Mask

The address conditions are set.

Setting range: $0 \leq \text{Address value} \leq 0\text{xfffff}$

The two types of address conditions are Address that sets the address value and Mask that inputs the mask value of the address. The setting methods are explained next.

a. Address

Input in the order of – .

The address condition has the following two possible settings.

(1) Point setting

Set the point setting to only the low-order address, or set the same values in the low-order address and the high-order address. The **Mask setting** can also be made at this time.

(2) Range setting

Set the address range in the low-order address and the high-order address. The **Mask setting** cannot be made at this time. When the address range is set, the starting address can only be set to an even address and the ending address to an odd address.

An address condition can also be specified by a symbol. The specification methods are described below.

Function and Variable	<u>_</u> fnc file#_fnc (for static functions and variables)
SFR	sfrname
Line number in the source text	file:no

fnc: function or variable name; sfrname: SFR name; file: file name; no: line number

If a function or variable name is specified, an underline (_) is specified at the beginning.

The separator between a file name and a function or variable name is the sharp (#). The separator between a file name and a line number is the colon (:).

b. Mask

A mask can be set as an address condition.

The default is 0x00000000 which is the setting for no mask.

The mask is set by an OR condition.


Example When the settings are Address – Mask

The condition is matched for addresses 0x4000 to 0x40FF.

When the settings are Address – Mask

The condition is matched for addresses 0x4000, 0x4001, 0x4100, and 0x4101.

(3) Status selection area

Status 

The status condition is selected. By selecting the status condition, the execution event and access event types can be simultaneously determined.

The contents of the status condition are shown below.

Status	Event Type	Meaning
Run	Execution Event	Program execution
OPFetch	Access event	Program fetch (including prefetch)
Program Read		Program data read
Program Write		Program data write
Program R/W		Program data read/write
Macro Read		Data read in a macro service
Macro Write		Data write in a macro service
Macro R/W		Data read/write in a macro service
Program/Macro Read		Data read
Program/Macro Write		Data write
Program/Macro R/W		Data read/write
BRM1		First instruction after a branch
VECT		Vector read by an interrupt
ALL (No Condition)		All access

(4) ata setting area

Data Mask

The data condition is set.

Setting range: $0 \leq \text{Data} \leq 0\text{xffff}$

The two types of data conditions are the Data that sets the data value and the Mask that inputs the data mask value. The setting methods are described below.

a. Data

The data value is specified.

The data value can be specified by a symbol. The specification methods are described below.

Function and Variable	_fnc file#_fnc (for static functions and variables)
SFR	sfrname
Line number in the source text	file:no

fnc: function or variable name; sfrname: SFR name; file: file name; no: line number

When a function or a variable name is specified, an underline () is specified at the beginning. The separator between a file name and a function or variable name is the sharp (#). The separator between a file name and a line number is the colon (:).

b. Mask

A mask can be set for the data value.

The default is 0xffff. The data condition becomes invalid (matches the condition for any data).

The mask is set by an OR condition.


Example When the settings are Data Mask

The condition is matched for the data values from 0x4000 to 0x40FF.

When the settings are Data Mask

The condition is matched for the data values of 0x4000, 0x4001, 0x4100, and 0x4101.

(5) Data size selection area

Data Size 


The data size is selected.

When an odd address was specified for the address, Word cannot be set for the data size.

The contents of the data size are shown below.

Data Size	Meaning
Byte	Data size byte condition
Word	Data size word condition
All (No condition)	Data size byte/word condition

(6) Pass count setting area

Pass count 

The pass count condition is set.

Setting range: $1 \leq \text{Pass count} \leq 0xff$

This area sets pass count condition that is satisfied when the event condition (address condition, status condition, data condition, external sense data condition) is matched multiple times during the execution of the user program.

If the pass count is set to 1, the condition is satisfied when the condition is matched.

If the pass count is set to 2 or more, the number of events that can be simultaneously valid is one.

(7) External trigger condition setting area☐ **External Trigger**

An external trigger condition is set.

If an external trigger condition is set, an address, status, and data cannot be specified.

Function Buttons**Restore Evnt**

button

Restores the event condition.

Make Evnt

button

Registers the event condition.

By clicking this button, the event condition is register in the event manager. (Red ☐ mark)

Close

button

Closes the event dialog.

Event Manager**Management Window****Overview**

Various events are displayed and deleted.

By opening this window, an event condition registered in the event dialog or event link dialog can be assigned to a break or a trace.

This window can be opened by the following methods.

■ In the main window

Select **B**rowse → **E**vent → **E**ventManager... in the menu bar.

■ In the main window

Press in order the **G**RAPH, **B**, **E**, and **M** keys.

■ In the source text window

The line number area where events are set is selected.

Select **V**iew → **E**vent? in the menu bar.

■ In the source text window

The line number area where events are set is selected. Press the **Event ?** button.

■ In the disassemble window

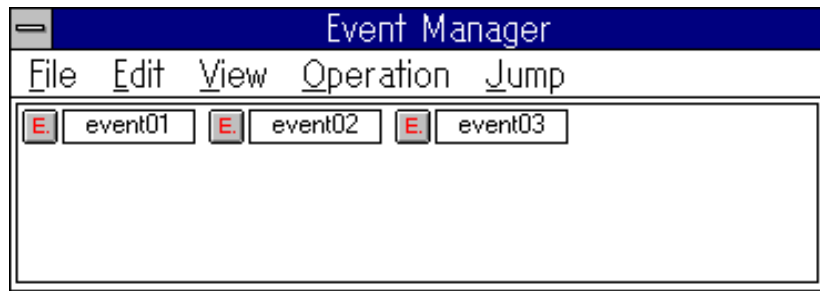
The address display area where events are set is selected.

Select **V**iew → **E**vent? in the menu bar.

■ In the disassemble window

The address display area where events are set is selected. Press the **Event ?** button.

Window



Normal display



Detailed display

Figure 5-26. Event Manager

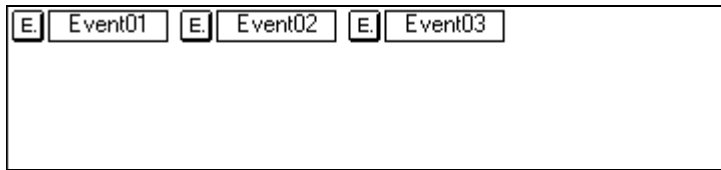
Functions

Various events are displayed and deleted. The event conditions are managed when registering and setting various event conditions (event links, breaks, traces).

The event manager contains the following items.

- Menu bar
- Event display area
- Detailed event display area

Each function is described next.

(1) Event display area

In the normal view mode







In the detailed view mode

The event icons of registered and set events are displayed in the event display area.







An icon consists of the mark that displays the event type and the event name.



The marks are listed below.

Mark	Mark Meaning
	Indicates an event condition.
	Indicates an event link condition.
	Indicates a break event.
	Indicates a trace event.

The color of the letter displayed in the mark indicates the setting state and type of the event.

Letter Color in Mark	Applicable Marks	Description
Red	 , 	The event and event link conditions are always displayed in red when registered.
	 , 	Indicates that an event is set. Satisfying a condition generates various events
Black	 , 	Indicates that an event is registered. Even if the condition is satisfied, an event is not generated

This area has two functions in addition to displaying the event icon.

a. Jump function

With the address condition of the selected icon as the jump pointer, the source text window, disassemble window, or memory window is jumped to. The jump destination window is displayed from the jump pointer. This function is executed in the following operation.

1. Select the icon.
2. In the event manager

When the jump destination is the source text window

Select **J**ump → **S**ourceText... in the menu bar, or press in order the **GRAPH**, **J**, and **S** keys, or press the **CTRL**+**U** shortcut keys.

b. Delete function

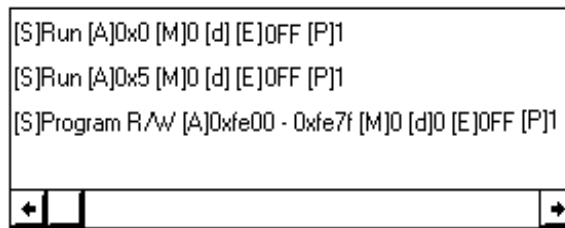
The event registration and setting of the selected icon are deleted.

If an event condition **E.** and an event link condition **L.** are deleted, they cannot be used in other events (**B.**, **T.**). If they will be used in other events, do so after deleting the events being used.

This function is executed by the following operations.

1. Select the icon.
2. In the event manager

Select **E**dit → **D**efine in the menu bar, or press in order the **GRAPH**, **E**, and **D** keys.

(2) Detailed event display area

A detailed event display area is displayed only when in the detailed view mode. The details about each event icon are displayed.

For the event conditions, the display contents are in order the status condition, address condition, address mask condition, data condition, external sense data condition, and pass count condition with the various keys as the separators. The correspondences with the key data are shown below.

For event conditions

Key data	Description
[S]	Status condition
[A]	Address condition
[M]	Address mask condition
[d]	Data condition
[P]	Pass count condition
[E]	External trigger condition

For event link conditions

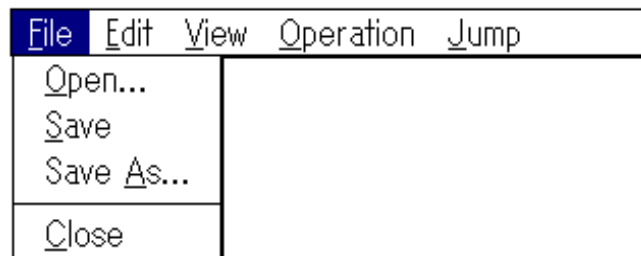
Key data	Description
[E] – [P4]	nth event link condition

For breaks, traces, and the timer

Key data	Description
[B]	Break condition
[SS]	Sequential trace start condition
[SE]	Sequential trace end condition
[Q]	Qualify trace condition

Menu Bar

(a) File

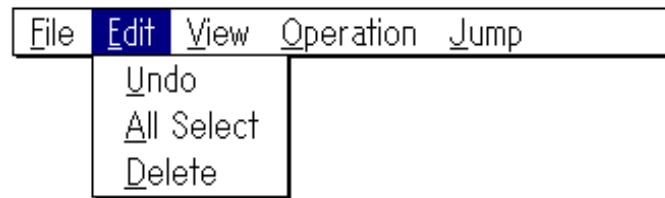


Open... The event setting file is loaded. The setting file selection dialog opens. (The event register/setting contents before loading are all lost.)

Save The current event settings are overwritten in the event setting file.

Save As... Saves the current event settings in the event setting file. The setting file selection dialog is opened.

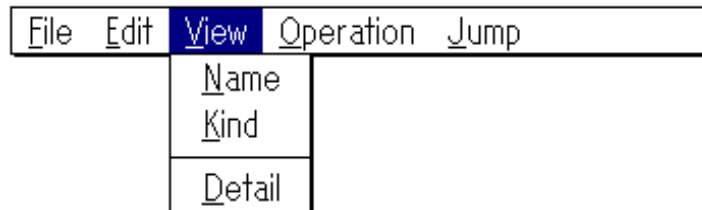
Close Closes the event manager.

(b) Edit

Undo Undoes the previous editing operation.

All Select Selects all of the icons.

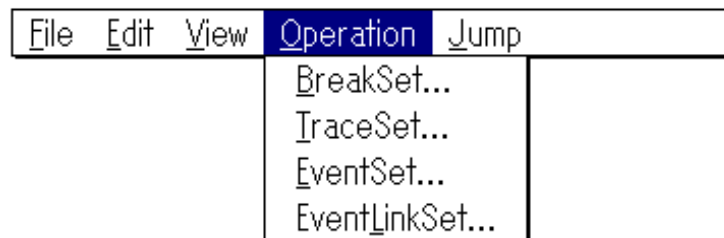
Delete Deletes the selected icon.

(c) View

Name The icons are arranged in order by the event name.

Kind The icons are arranged in order by the kind of event.

Detail Toggles between the normal display and the detailed display.

(d) Operation

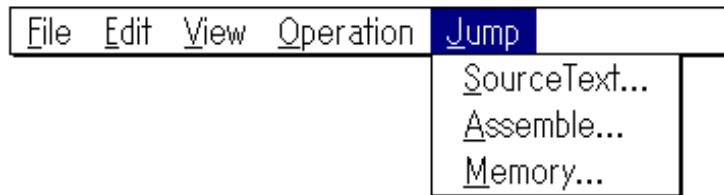
BreakSet... Opens the break dialog.

TraceSet... Opens the trace dialog.

EventSet... Opens the event dialog.

EventLinkSet... Opens the event link dialog.

(e) **J**ump



SourceText... With the address setting in the selected event as the jump destination address, the appropriate source text and source lines are displayed. The source text window opens.

Assemble... With the address setting in the selected event as the jump destination address, the disassemble from that address is displayed. The disassemble window opens.

Memory... With the address setting in the selected event as the jump destination address, the memory contents from that address are displayed. The memory window opens.

Memo

Various Event Setting Methods

By using the event manager, the breaks and traces can be easily set. Next, the event setting methods are described.


- (1) Open the event dialog.
(**B**rowse → **E**vent → **E**ventSet... in the menu bar)
- (2) Set the event conditions.
Two events are registered under the names of Event01 and Event02.
- (3) Open the event manager.

(**B**rowse → **E**vent → **E**ventManager... in the menu bar)



When opened, it is clear that the two events of Event01 and Event02 are registered in the event manager.







- (4) Open the window for setting events (trace dialog, break dialog, timer dialog, event link dialog). Here, the break dialog is opened.

(**B**rowse → **B**reakSet... in the menu bar or the  button)

- (5) Click the icon in the event manager.

By dragging the icon, the mouse cursor changes from  to . When the mouse moves in this state, the dragged icon moves simultaneously.

- (6) A mouse in the state in (5) moves to the break dialog .   t01

When moved to the break dialog, the mouse cursor changes from  to . By dropping the mouse in this state, the icon is copied in the break dialog.

- (7) By entering the break event name and clicking the **Make Brk** button, the break event can be registered.

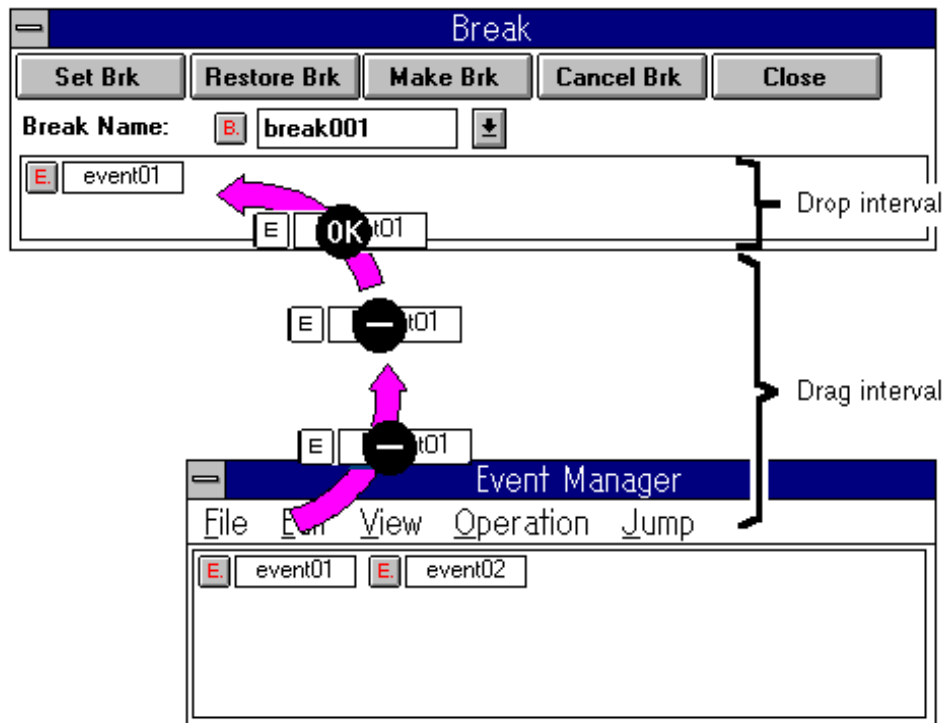


Figure 5-27. Event Setting Example

Event-related Windows

The event manager manages all of the events. The related images are shown below.

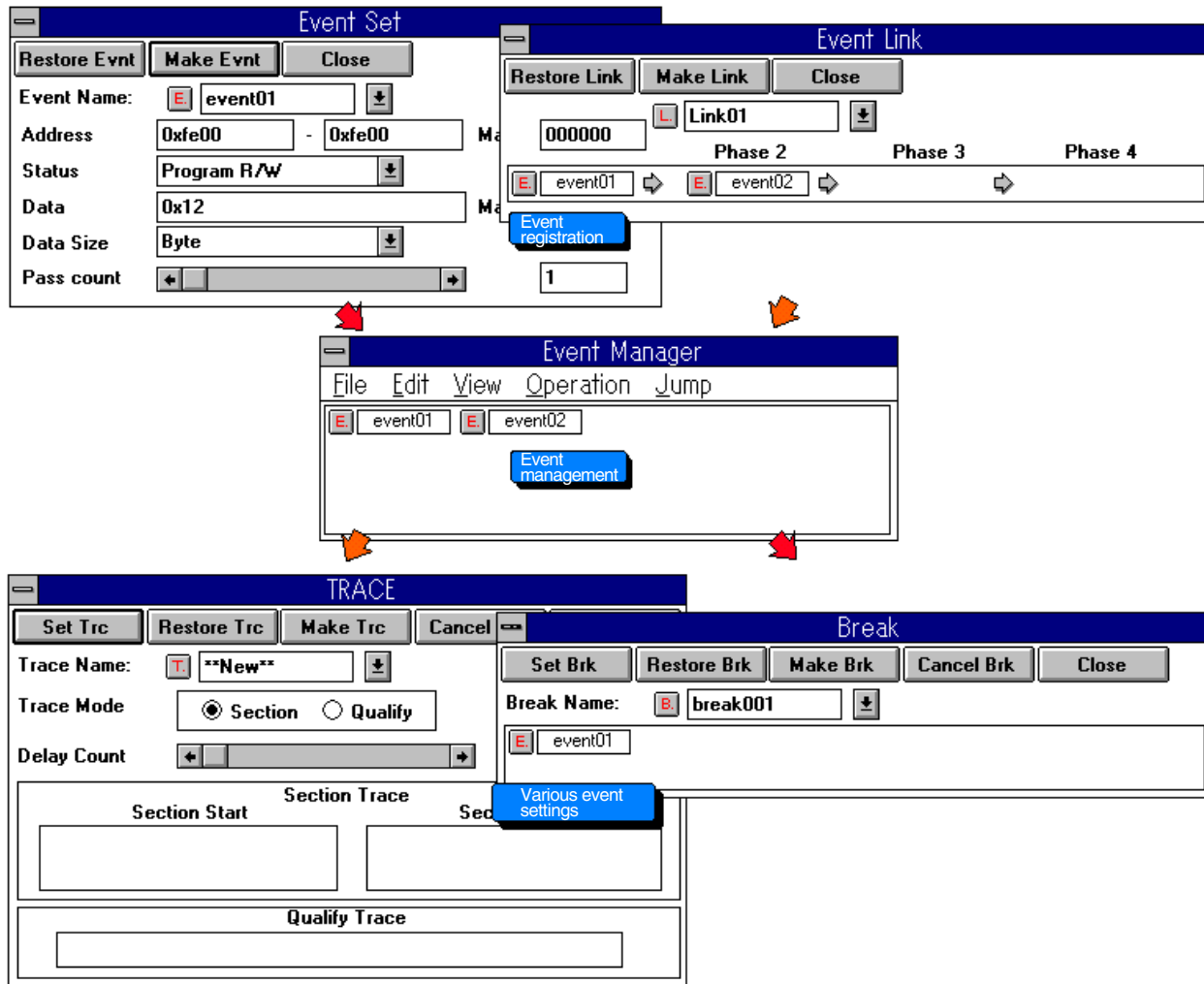


Figure 5-28. Event-related Images

Event Link Dialog	Setting Dialog (Modeless)
-------------------	---------------------------

Overview

The event link conditions are registered and displayed.

An event link condition registered in this dialog is automatically registered in the event manager.

This dialog can be opened by the following methods.

- In the main window
Select **B**rowse → **E**vent → **E**ventLinkSet... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, **E**, and **L** keys.
- In the event manager
Select **O**peration → **E**ventLinkSet... in the menu bar.
- In the event manager
Press in order the **GRPH**, **O**, and **L** keys.

Window

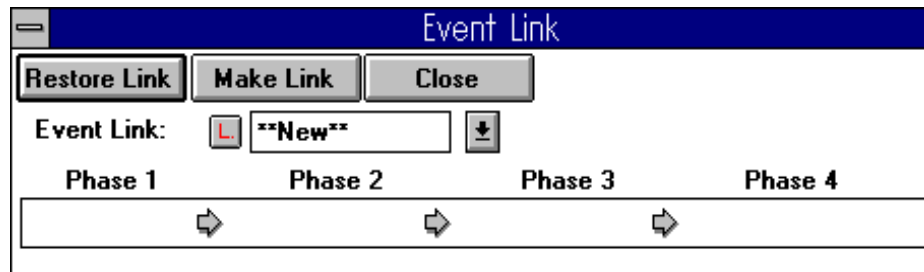


Figure 5-29. Event Link Dialog

Functions

Event link conditions are registered and displayed.

The registration of an event link condition is set by using the event conditions registered in the event manager. The event link condition can have up to four phases.

For an event condition used in an event link condition, use a pass count of one event condition. An event condition with a pass count other than 1 cannot be used.

Events are generated only when the user program is executed in accordance with the specification order of the set event conditions. However, if a disable condition is detected during operation, event conditions that have been satisfied are initialized, and the first event condition becomes the detection target.

The event link dialog contains the following items.

- Event link name setting area
- Link condition setting area
- Function buttons

Each function is described next.

(1) Event link name setting area

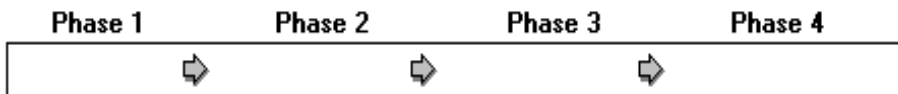
Event Link:  

The event link name is set and selected. "***NEW***" is displayed by default.

By pressing the  button when selecting an event link name, the selection is made from the drop-down list.

A maximum of eight characters can be set in an event link name.

(2) Link condition setting area



The settings are made in the order of the event condition and the event detection.

The order is set to **Phase 1 → Phase 2 → Phase 3 → Phase 4**. The setting does not have to include **Phase 4**.

If the setting does not include **Phase 4**, when an event condition set in the final phase is selected, the event is generated.

Function Buttons**Restore Link**

button

Restores an event link condition.

Make Link

button

Registers an event link condition.

By clicking this button, the event link condition is registered in the event manager.
(Red **L.** mark)

Close





button

Closes the event link dialog.



Notes**Setting Event Link Conditions**

An example showing how to set event link conditions is presented.

- (1) Open the event dialog.
(**B**rowse → **E**vent → **E**ventSet... in the menu bar)
- (2) Create the event conditions in the event dialog.
Five events are registered with the event names of E_INIT, E_SUB0, E_SUB1, E_SUB2, and E_SUB4.
- (3) Open the event manager.
(**B**rowse → **E**vent → **E**vent**M**anager... in the menu bar)
- (4) Open the event link dialog.
(**B**rowse → **E**vent → **E**vent**L**inkSet... in the menu bar)
- (5) Drag the icon in event manager.

Dragging the icon changes the mouse cursor from  to . When the mouse moves in this state, the dragged icon moves simultaneously.  

- (6) Move the mouse in the state in (5) to the event link dialog.

When moved to the event link dialog, the mouse cursor changes from  to . Dropping the mouse in this state copies the icon to the event link dialog.

- (7) Repeat the operations in (5) and (6) to make the following settings in the event link dialog from the event manager.

Setting Position	Set Event
Phase 1	E_INIT
Phase 2	E_SUB0
Phase 3	E_SUB1
Phase 4	E_SUB4

- (8) Enter the name of the event link.
Enter the name E_LINK.

- (9) Press the **Make Link** button to register the event link condition E_LINK in the event manager.

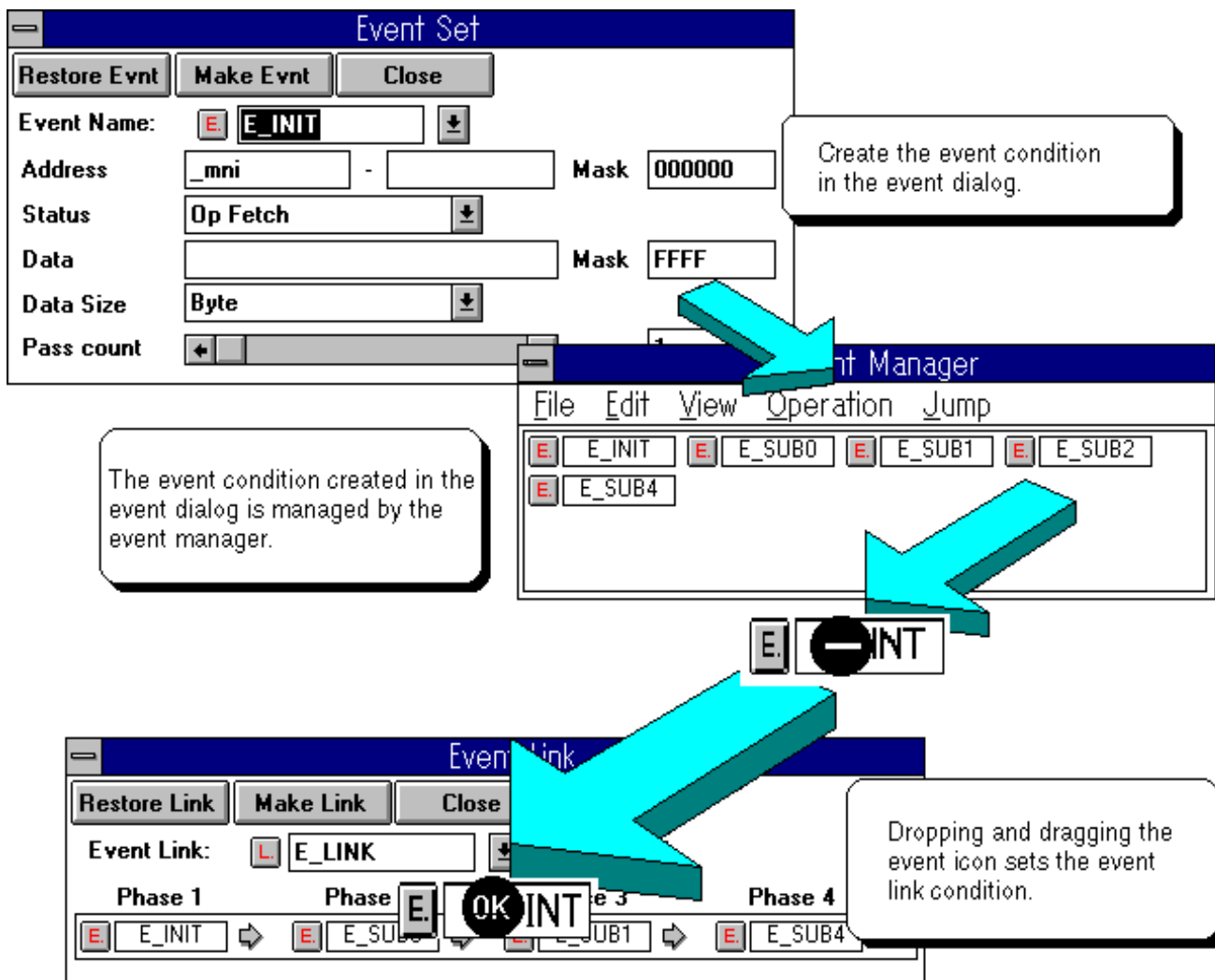


Figure 5-30. Setting Event Link Conditions

Example Using Event Link Conditions

An example showing how to use event link conditions is presented.

The event link conditions are set for the program shown in Figure 5-29. This program has the following structure.

Main processing

1. Initialization process (INIT)
2. Sub program 0 (SUB0)
3. Condition decision
 - a. Sub program 1 (SUB1)
 - b. Sub program 2 (SUB2)
4. Sub program 4 (SUB4)

The execution route of this program is viewed as routes (1) and (2) shown in the figure.

This program execution follows route (1) and route (2). When a event is generated, the event condition shown in the figure is set. Setting the following event link condition can cause the desired events to be generated.

Setting Position	Set Event
Phase 1	E_INIT
Phase 2	E_SUB0
Phase 3	E_SUB1, E_SUB2
Phase 4	E_SUB4

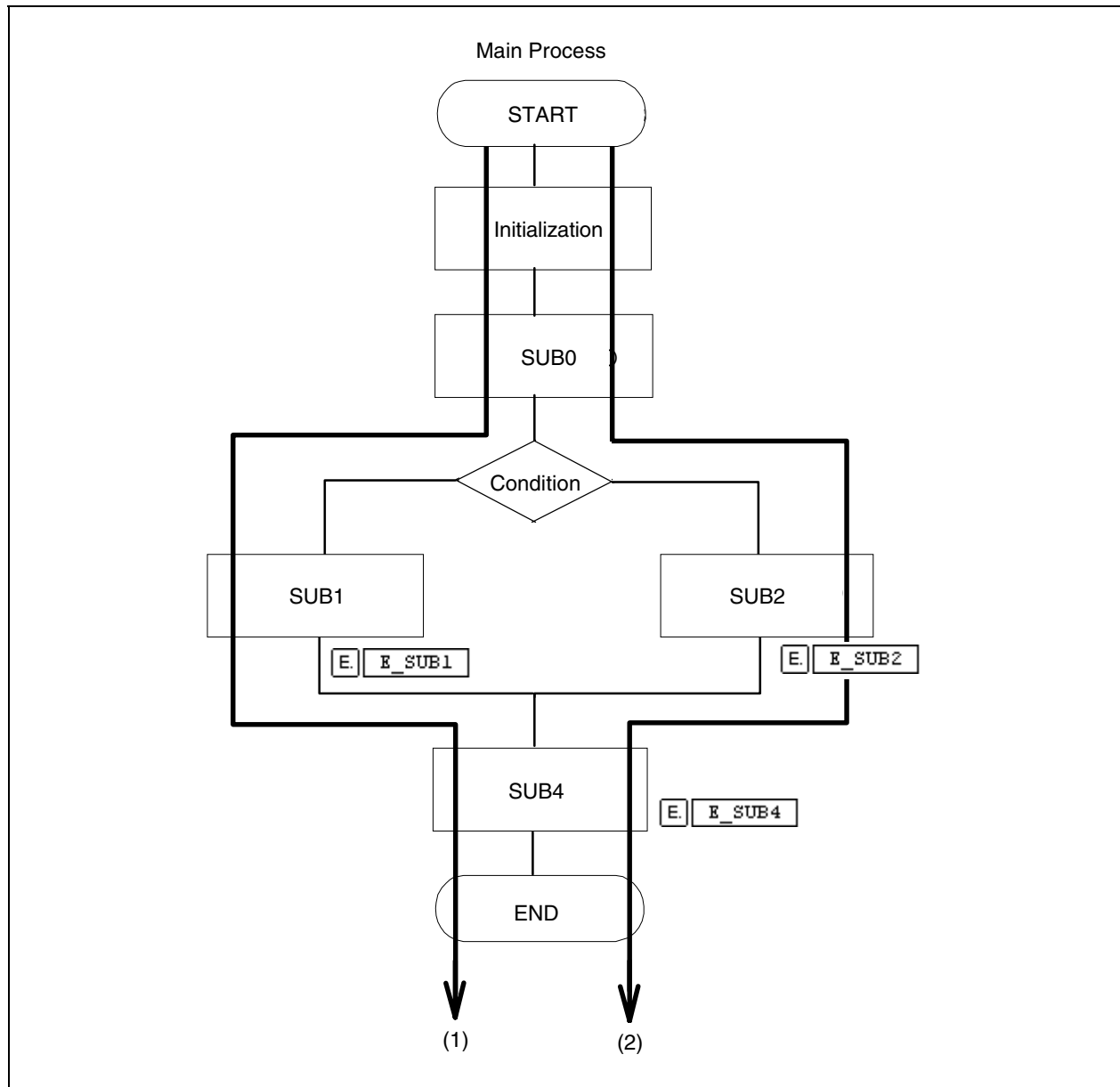


Figure 5-31. Example Using Event Link Conditions

Break Dialog**Setting Dialog (Modeless)**

Break event conditions are registered, set, and displayed.

The break event condition registered in this dialog is automatically registered in the event manager.

This dialog can be opened by the following methods.

- In the main window
Select **Browse** → **BreakSet...** in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **B** keys.
- Click the  button in the tool bar.
- In the event manager
Select **Operation** → **BreakSet...** in the menu bar.
- In the event manager
Press in order the **GRPH**, **O**, and **B** keys.

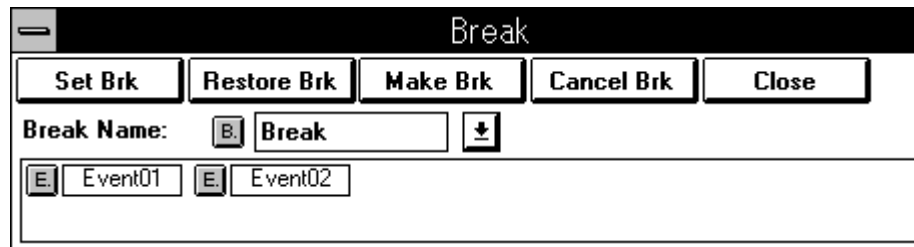
Window

Figure 5-32. Break Dialog

Functions

Break event conditions are registered, set, and displayed.

Break event conditions are registered by using the event conditions and the event link conditions registered in the event manager.

A maximum of 32,767 break event conditions can be registered. However, a maximum of ten break event conditions can be simultaneously used.

The break dialog contains the following items.

- Break event name setting area
- Break condition setting area
- Function buttons

Each function is described next.

(1) Break event name setting area

Break Name:  Break 

Break event name is set and selected. "***NEW***" is displayed by default.

To select the break event name, press the  button and select from the drop-down list.

A maximum of eight characters can be set in a break event name.








(2) Break condition setting area

 Event01  Event02

The break event condition is set. Settings are made by dropping and dragging the icons for the event conditions and the event link conditions registered in the event manager.

A maximum of 12 conditions can be set for the event conditions and event link conditions that can be set in the break condition setting area.





Function Buttons

	button	Enables the break conditions. The  mark changes to red. (Break events are generated.)
	button	Restores the break condition.
	button	Registers the break conditions. A registered break condition enters the disable state.
	button	Disables the break condition. The  mark changes to black. (Break events are not generated.)
	button	Closes the break dialog.



Notes**Setting Break Event Conditions**

An example showing how to set break event conditions is presented.

- (1) Open the event dialog.
(**B**rowse → **E**vent → **E**ventSet... in the menu bar)
- (2) Create the event conditions in the event dialog.
Two events are registered with the event names of Event01 and Event02.
- (3) Open the event manager.
(**B**rowse → **E**vent → **E**vent**M**anager... in the menu bar)
- (4) Open the break dialog.
(**B**rowse → **B**reakSet... in the menu bar)
- (5) Drag the icon in event manager.


Dragging the icon changes the mouse cursor from  to . When the mouse moves in this state, the dragged icon moves simultaneously.  

(6) Move the mouse in the state in (5) to the break dialog.

Moving to the break dialog changes the mouse cursor from  to . Dropping the mouse in this state copies the icon to the break dialog.

(7) Enter the break event name.

Enter the name of BREAK.

(8) Pressing the  button registers the BREAK break event condition in the event manager.

(9) Pressing the  button changes the mark of the BREAK break event condition from black to red.

This indicates that the break event is enabled.

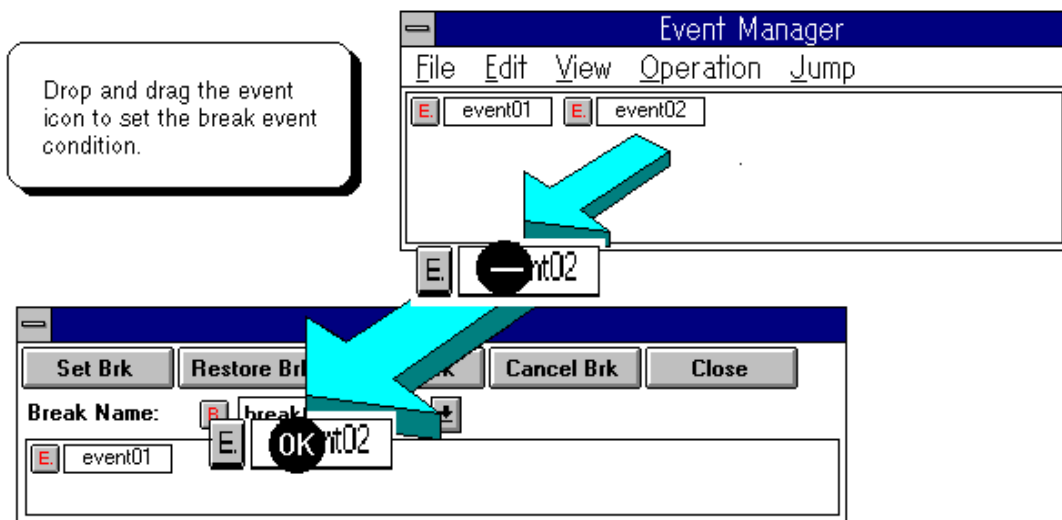



Figure 5-33. Setting Break Event Conditions

Trace Dialog**Setting Dialog (Modeless)**

Trace event conditions are registered, set, and displayed.

A trace event condition registered in this dialog is automatically registered in the event manager.

This dialog can be opened by the following methods.

- In the main window
Select **B**rowse → **T**race → **T**raceSet... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, **C**, and **T** keys.
- Click the  button in the tool bar.
- In the event manager
Select **O**peration → **T**raceSet... in the menu bar.
- In the event manager
Press in order the **GRPH**, **O**, and **T** keys.

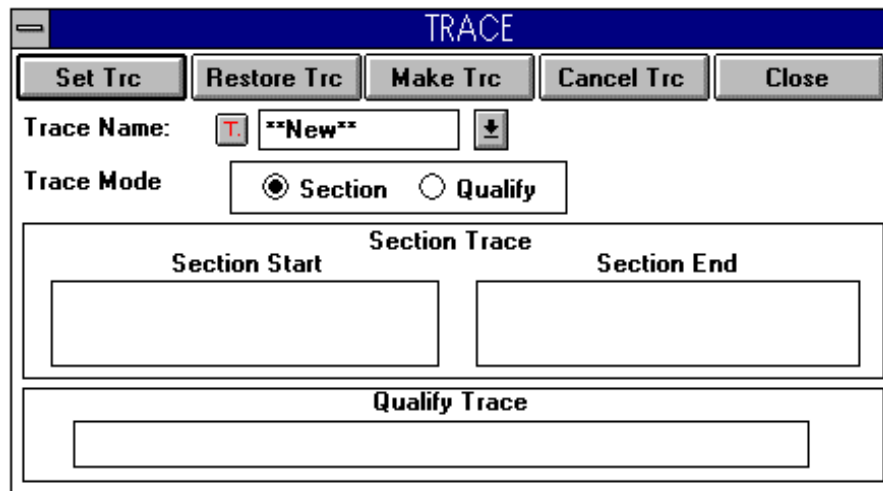
Overview

Figure 5-34. Trace Dialog

Functions

Trace event conditions are registered, set, and displayed.

Trace event conditions are registered by using the event conditions and event link conditions registered in the event manager.

A maximum of 32,767 trace event conditions can be registered. However, only one trace event condition is enabled.

If you wish to operate the tracer in accordance with the trace event conditions, always select **Execute → Cond. Trace ON** in the menu bar of the main window.

This trace dialog contains the following items.

- Trace event name setting area
- Trace mode setting area
- Delay count setting area
- Section trace condition setting area
- Qualify trace condition setting area
- Function buttons

Each function is described next.

(1) Trace event name setting area

Trace Name: 

The trace event name is set and selected. "***NEW***" is displayed by default.

To select the trace event name, press the  button and select from the drop-down list.

A maximum of eight characters can be set in a trace event name.

(2) Trace mode setting area

Trace Mode ☒ Section ☐ Qualify

The trace mode is set.

The three trace modes are **All Trace**, **Section Trace**, and **Qualify Trace**.

Trace Mode	Trace Description
All Trace	Trace all of the causes.
Section Trace	Trace only the section specified by the event conditions.
Qualify Trace	Trace only the locations with matched event conditions. Only access events can be set

When setting the trace mode, setting the menu bar in the main window and setting in this area are required. The trace modes and each setting are shown below.

Trace Mode	Execute (X) Setting in the Menu Bar of the Main Window	Trace Mode Setting in the Trace Dialog	Delay Conditions
All trace	Select Uncond. Trace ON.	—	None
Section trace	Select Cond. Trace ON.		
Qualify trace		Select qualify.	Yes

(3) Section trace condition setting area

Section Start	Section End

The event conditions for a section trace are set.

Section Start : Sets the trace start event condition.

Section End : Sets the trace end event condition.

The event conditions are set by dragging and dropping the icons of the event conditions and event link conditions registered in the event manager.

A maximum of ten event conditions and event link conditions can be set in the section trace condition setting area.

(4) Qualify trace condition setting area

Qualify Trace


The event conditions for a qualify trace are set.

Event conditions are set by dragging and dropping the icons of the event conditions and event link conditions registered in the event manager.

A maximum of ten event conditions and event link conditions can be set in the qualify trace condition setting area.

Function Buttons

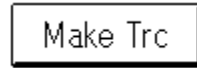
button

Enables the trace setting conditions. The  mark becomes red. (Trace events are generated.)



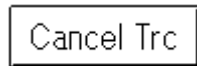
button

Restores the trace setting condition.




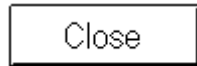
button

Registers a trace setting condition. The registered trace condition is disabled.



button

Disables the trace setting condition. The  mark becomes black. (Trace events are not generated.)



button

Closes the trace dialog.

Notes**Setting Trace Event Conditions**

An example showing how to set trace event conditions is presented.



- (1) Select the **Execute → Cond. Trace ON** in the menu bar in the main window.
- (2) Open the event dialog.
(**Browse → Event → EventSet...** in the menu bar)
- (3) Make the event condition in the event dialog.
The two events with the event names of Event01 and Event02 are registered.
- (4) Open the event manager.
(**Browse → Event → EventManager...** in the menu bar)
- (5) Open the trace dialog.
(**Browse → Trace → TraceSet...** in the menu bar)
- (6) Set the trace mode and the break condition.
- (7) Drag the icon in the event manager.

Dragging the icon changes the mouse cursor from  to . When the mouse moves in this state, the

dragged icon moves simultaneously.

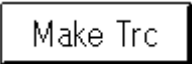



- (8) Move the mouse in the state in (7) to the trace dialog.

Moving to the trace dialog changes the mouse cursor from  to . Dropping the mouse in this state copies the icon to the trace dialog.

- (9) Enter the trace event name.

Enter the name of TRACE.

- (10) Press the  button to register the TRACE trace event condition in the event manager.

- (11) Press the  button to change the mark of the TRACE trace event condition from black to red.

This indicates that the trace event is enabled.

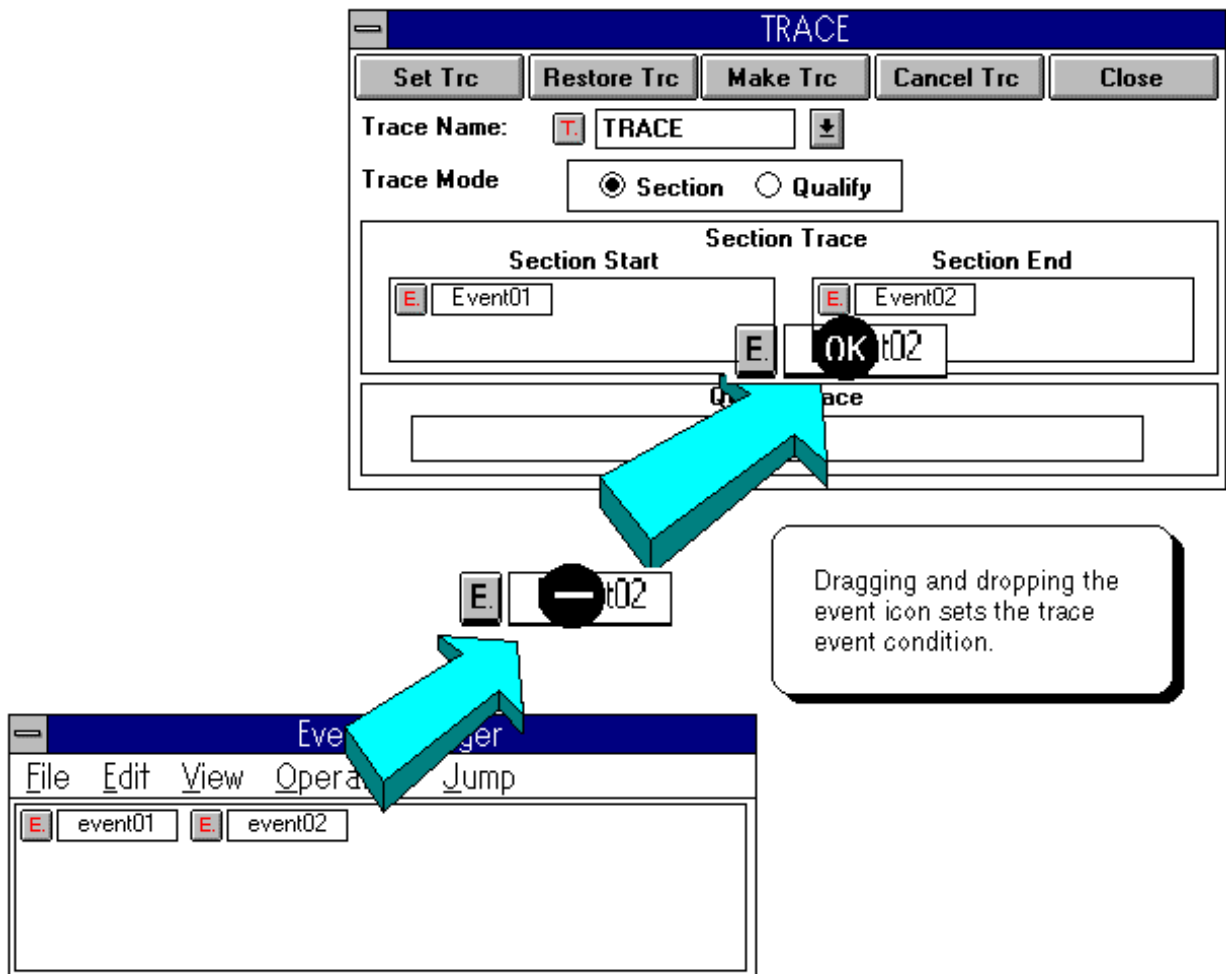



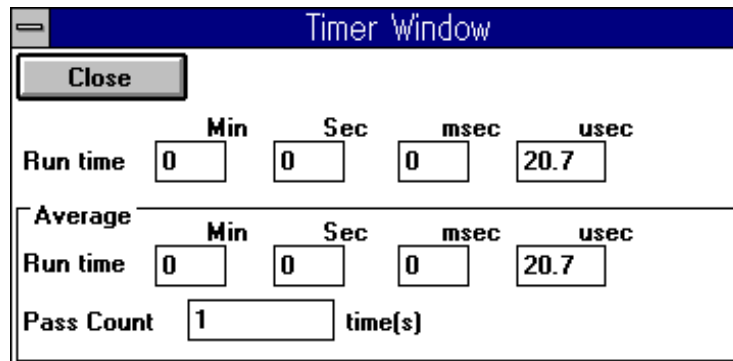
Figure 5-35. Setting Trace Event Conditions

Timer Window Dialog**Display/Setting Dialog (Modeless)**

The run time measurement is displayed.

This dialog can be opened by the following methods.

- In the main window
Select **B**rowse → **T**imer... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **I** keys.
- Click the  button in the tool bar.

Window


	Min	Sec	msec	usec
Run time	0	0	0	20.7
Average				
Run time	0	0	0	20.7
Pass Count	1 time(s)			

Figure 5-36. Timer Window Dialog

Functions

This dialog displays run time measurement from the trace start condition to the trace end when a section trace of the trace events is enabled.

The timer dialog contains the following items.

- Run time display area
- Average run time display area
- Function buttons

Each function is described next.

(1) Run time display area

Run time **Min** **Sec** **msec** **usec**
 0 0 0 20.7

The measurement of the program's run time is displayed. Time can be measured to a maximum around 14 minutes and 33 seconds at a resolution of 203.45 nanoseconds.

(2) Average run time display area

Average
 Min **Sec** **msec** **usec**
 Run time 0 0 275 733
 Pass Count 7 time(s)

The average run time and the execution count are displayed.

Item	Description
Run time	Displays the average run time.
Pass Count	Displays the pass count. Pass count: $0 \leq \text{Count} \leq 65,535$

Function Buttons

Close button Closes the timer dialog.

Cautions


The run time measurement displays the error message of "Result of Timer measurement is over." if the timer event name was selected when the maximum possible measurement time exceeded 14 minutes and 33 seconds or when the maximum measurement count exceeded 65,535 while measuring. This indicates that correct measurements are not possible.

Trace View Window	Display Window
-------------------	----------------

Overview

The trace result is displayed.

This window can be opened by the following methods.

- In the main window
Select **B**rowse → **T**race → **T**raceView... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, **C**, and **V** keys.
- Click the  button in the tool bar.

Window

Trace View									
	Frame	Address	Data	Statu	Address	Data	Statu	DisAsm	
B	A	00015	00008B 09	M1				CALL	!!_hdwinit
		00016							
	A	00017	00008C F0	OP					
	A	00018	00008D 5D	OP					
	A	00019	00008E 50	OP					
	A	00020	00008F 38	NON					
	A	00021			OFFE7F 70		WR		
	A	00022	000090 DB	NON					

Figure 5-37. Trace View Window

Functions

The trace result is displayed.

The tracer has a capacity of 32,768 frames and has a ring structure. Therefore, if more than 32,768 frames of data are written, the oldest data is overwritten. And the oldest data on this display is frame 0, and the frame numbers are displayed in order.

During pauses in the execution of the user program, the block data is written to the tracer. The block data display has one horizontal line in each display area. Block data is written in the following cases depending on the previous and next execution modes.

Previous Execution Mode	Next Execution Mode
In real-time execution	During real-time execution During step execution
In step execution	During real-time execution When the execution address was changed and the execution was in steps

The trace view window contains the following items.

- Point mark display area
- Trace mode display area
- Trace result display area

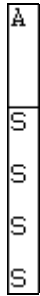
Each function is described next.

(1) Point mark display area



The setting states of the events are displayed. If the execution event or access fetch event is set at the corresponding trace address, the mark corresponding to the event type is displayed.

Mark	Mark Meaning
E	Indicates that an event condition is set.
L	Indicates that the last phase of event link is set.
B	Indicates that a break event is set.
T	Indicates that a trace event is set.

(2) Trace mode display area

The type of trace mode is displayed.

A: All trace or section trace mode

Q: Qualify trace mode

S: Step execution trace

(3) Trace result display area

Frame	Address	Data	Status	Address	Data	Status	DisAsm
00015	00008B	09	M1				CALL !!_hdwinit
00016							
00017	00008C	F0	OP				
00018	00008D	5D	OP				
00019	00008E	50	OP				
00020	00008F	38	NON				
00021				0FFE7F	70	WR	
00022	000090	DB	NON				

①
②
③
④

The trace result is displayed. Selecting this area allows the jump function and the window connect function to be used.

The window connect function can be used by using the following operations.

1. In the main window, select the window you wish to connect by using **Operation → WindowConnect** in the menu bar.

Items in Connect (<u>W</u>)	Connect Window
<u>S</u> ourceText	Source text window
<u>A</u> ssemble	Disassemble window
<u>M</u> emory	Memory window

2. Open the window selected in 1 and the trace view window.
3. Use the mouse to select the trace result display area of the trace view window.
4. With the address of the trace result selected in 3 as the pointer, the display areas of each window selected in 1 are highlighted.

The window connect function differs from the jump function. When the area selected in the trace view window is moved, the result is simultaneously highlighted in each window of the connect target.

The following contents are displayed in the trace result display area.

- (1) Trace frame number display
- (2) Fetch access display
- (3) Data access display
- (4) Mnemonic display

a. Trace frame number display (Frame)

The trace frame number is displayed.

Range: $0 \leq \text{Trace frame number} \leq 32,767$

The trace frame number display can be selected from the **View → TraceView → Frame** item in the menu bar of the main window.

b. Fetch access display (Address Data Statu)

The result of a program fetch is displayed. This field is displayed in the following way based on the status displayed in the status display field.

Statu	Display Contents	
BRM1	Program fetch display	First byte of the fetch in the first instruction after a branch
M1		Fetch of the first byte of the instruction
OP		Op code fetch
NON		Invalid fetch
Otherwise.	Nothing displayed	

For a program fetch display

The following is displayed.

Address	Fetch address display
Data	Fetch data display

The fetch access display can be selected from the items in the menu bar in the main window.

Item	Selection
Fetch address display	View → TraceView → InstructionFetchAddress item
Fetch data display	View → TraceView → InstructionFetchData item
Fetch status display	View → TraceView → InstructionFetchStatus item

c. Data access display (Address Data Statu)

The data access result is displayed.

Status	Display Contents
INTWR	Vector read
WR	Data read/write by user program
RD	Data read by user program
WR	Data write by user program
MSWR	Data read/write by macro service
MSRD	Data read by macro service
MSWR	Data write by macro service

Address	Address display
Data	Data display

The data access display can be selected from the items in the menu bar in the main window.

Item	Selection Method
Address display	<u>V</u> iew → T raceView → M emoryAccess A ddress item
Data display	<u>V</u> iew → T raceView → M emoryAccess D ata item
Status display	<u>V</u> iew → T raceView → M emoryAccess S tatus item


d. Mnemonic display (DisAsm)

The disassemble result is displayed.

The status is only displayed for BRM1 and M1.

The mnemonic display can be selected from the **V**iew → **T**raceView → **D**isAssemble item in the menu bar of the main window.

Icon

The trace view window can be displayed as an icon by clicking the  button in the title bar.



Trace View

Trace Window Dialog**Specification Dialog (Modal)****Overview**

The display conditions are specified for the trace result to be displayed in the trace view window.

This dialog can be opened by the following methods **when the current window is the trace view window.**

- In the main window
Select **V**iew → **T**raceView → **P**ickUp... in the menu bar.
- In the main window
Press in order the **GRPH**, **V**, **T**, and **P** keys.

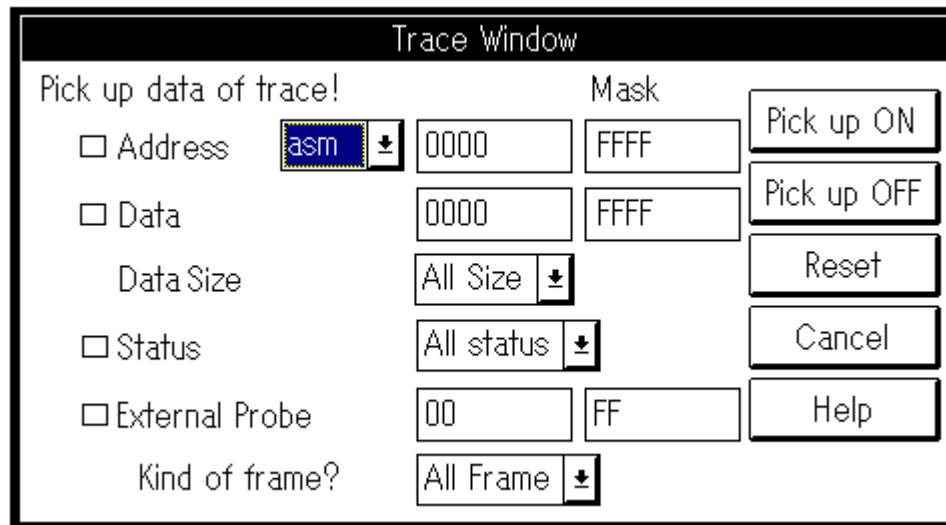
Window

Figure 5-38. Trace Window Dialog

Functions

The display conditions are specified for the trace result to be displayed in the trace view window.

The trace pick up dialog contains the following items.

- Address condition specification area
- Data condition specification area
- Status condition selection area
- Display frame condition selection area
- Function buttons

Each function is described next.

(1) Address condition specification area

☐ Address

asm

↓

0000

FFFF

Mask

The address value is specified when added to the pick up condition. The address specification mode, address value, and address mask value are specified.

The inputs for the address value and the address mask value are specified in the range from 0 to 0xffff.

The address value input differs with the mode being specified.

Mode	Input Data
Source mode	Variable name, function name, line number
Instruction mode	Immediate address, symbol name

(2) Data condition specification area

☐ Data

0000

FFFF

Data Size

All Size


↓

The data value is specified when added to the pick up condition. The data value, data mask value, and data size are specified.

The inputs of the data value and data mask value are specified in the range from 0 to 0xffff.

The data size can be selected from the following sizes.

Data Size	Meaning
All Size	All of the access sizes are search targets.
Byte	Only frames accessed in bytes are search targets.
Word	Only frames accessed in words are search targets.


(3) Status condition selection area☐ StatusAll status 

The status is selected when added to the pick up condition. The status conditions that can be selected are shown below.

Status Condition	Meaning
All status	All of the frames become pick up targets.
BRM1	Only the frames in the first M1 fetch after a program branch become pick up targets.
M1	Only the frames in an M1 fetch become pick up targets.
OP	Only the frames in a fetch become pick up targets.
R	Only the frames in a read become pick up targets.
RM	Only the frames in a read in macro service processing become pick up targets.
RP	Only the frames in a read by a user program become pick up targets.
RW	Only the frames in a read or write become pick up targets.
RWM	Only the frames in a read or write by a macro service process become pick up targets.
RWP	Only the frames in a read or write by a user program become pick up targets.
VECT	Only the frames in a vector read become pick up targets.
W	Only the frames in a write become pick up targets.
WM	Only the frames in a write by a macro service become pick up targets.
WP	Only the frames in a write by a user program become pick up targets.

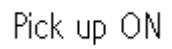
(4) Display frame condition selection area

Kind of frame?

All Frame 

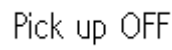
The type of the display frame is selected. The types of frames that can be selected are shown below.

Frame Type	Description
All Frame	All of the frames become pick up targets
Step	Only the step execution frames become pick up targets.
Next	Only the next step execution frames become pick up targets.
Real Time	Only the real-time execution frames become pick up targets.

Function ButtonsA rectangular button with a black border and the text "Pick up ON" inside.

button

Displays only the frames matching the condition.

A rectangular button with a black border and the text "Pick up OFF" inside.

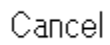
button

Displays all of the frames.

A rectangular button with a black border and the text "Reset" inside.

button

Initializes the settings.

A rectangular button with a black border and the text "Cancel" inside.

button

Closes the trace pick up dialog.


A rectangular button with a black border and the text "Help" inside.

button

Opens the help window.

Register Window**Display/Setting Window****Overview**

The registers (general-purpose registers, control registers) are displayed and changed. This window can be opened by the following methods.

- In the main window
Select **B**rowse → **R**egister... in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **R** keys.
- Click the  button in the tool bar.

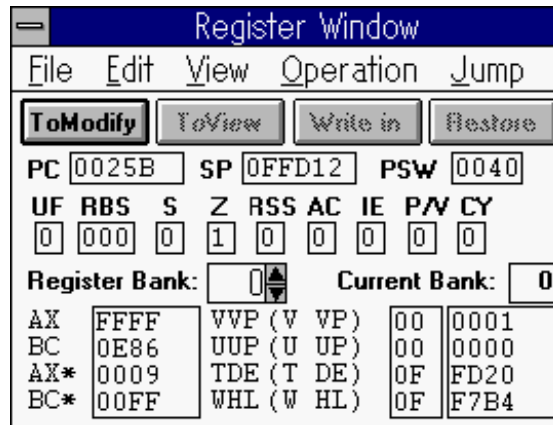
Window

Figure 5-39. Register Window

Functions

The registers (general-purpose registers, control registers) are displayed and changed. This window has a view mode and a modify mode.

This register window contains the following items.

- Control register display area
- Register bank setting area
- General-purpose display area
- Menu bar
- Function buttons

Each function is described next.

(1) Control register display area

PC 002B2 SP 0FFCD7 PSW 0048
 UF RBS S Z RSS AC IE P/V CY
 0 000 0 1 0 0 1 0 0

The control registers are displayed and changed. Changes can be made by pressing the **ToModify** button.

Pressing the **Write in** button after making the change writes the change to the target.

In addition to displaying and changing the control registers, this is also the jump pointer of the jump function.

The jump function uses the value of the selected control register as the jump pointer and jumps to the source text window, disassemble window, or memory window. The jump destination window displays from the jump pointer.


This function is executed by the following operations.

1. Select the control register.
2. In the register window, select **Jump** → **Source Text...** in the menu bar, or press in order the **GRPH**, **J**, and **S** keys, or press the **CTRL** + **U** shortcut keys. (When the jump destination is the source text window)

(2) Register bank setting area

Register Bank:  Current Bank: 0

The bank numbers of the general-purpose registers are displayed and set.

Item	Description
Register Bank:	The register bank shown in the general-purpose register display area is displayed and set. The bank number is changed by the  button.
Current Bank:	The register bank number currently set in the target (current bank) is displayed.

(3) General-purpose display area

AX 0000 VVP (V VP) 00 0001
 BC 0000 UUP (U UP) 0F FCD7
 AX* 0000 TDE (T DE) 0F FDB6
 BC* 0000 WHL (W HL) 0F FCD7

The register with the bank number displayed in Register Bank: in the register bank setting area is displayed and changed.

Changes can be made by pressing the **ToModify** button. Press the **Write in** button after making the change to write the change to the target.

By using the items in **View** in the menu bar in the register window, the display method for general-purpose registers can toggle between

Absolute name display ↔ Function name display

Register display ↔ Register pair display

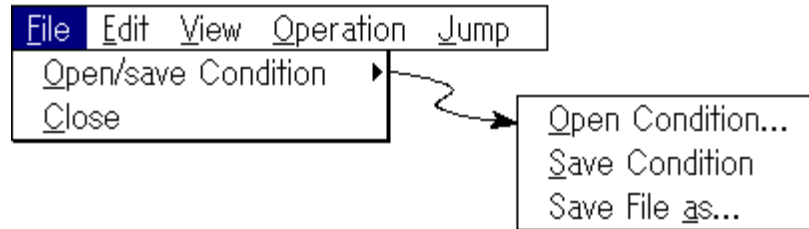
If **View** → **Functional Name** in the menu bar in the register window was selected, the views of the **A**, **X**, **B**, **C**, **AX**, and **BC** registers change as shown below based on the contents of the RSS bit.

Register Display	When RSS = 0	When RSS = 1
X	X	R0
A	A	R1
C	C	R2
B	B	R3
X *	R4	X
A *	R5	A
C *	R6	C
B *	R7	B
AX	AX	RP0
BC	BC	RP1
AX*	RP2	AX
BC*	RP3	BC

Menu Bar

Click the mouse in the menu bar to display the pull-down menu.

(a) File



Open/Save Condition ►

Open Condition

Save Condition

Save File As...

Loads and saves general-purpose registers.

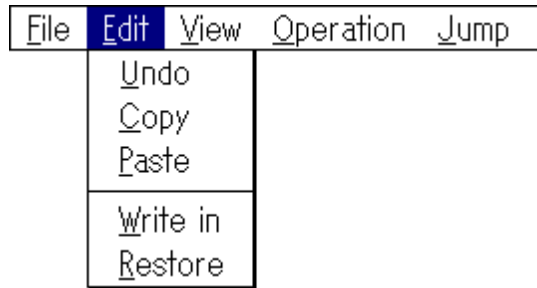
Opens the selected file for reference. Opens the view file selection dialog.

Saves the window contents in the view file.

Saves the window contents in the view file. Opens the view file selection dialog.

Close

Closes the register window.

(b) Edit**Undo**

Undoes the previous editing operation.

Copy

Copies the selected string to the clipboard buffer.

Paste

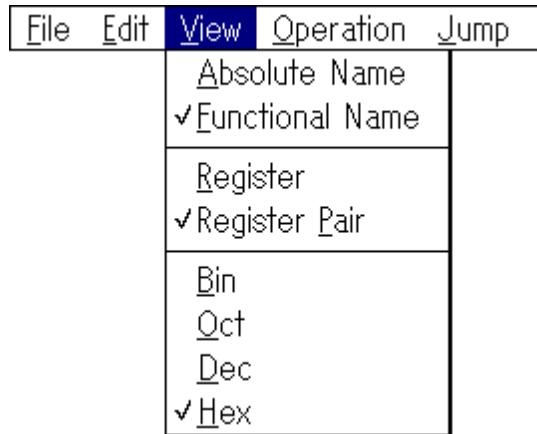
Pastes the contents of the clipboard buffer.

Write in

Writes the changes.

Restore

Cancels the changes.

(c) View**Absolute Name**

View register names as absolute names.

Functional Name

View the register names as functional names.

Register

View by symbol register.

Register Pair

View by register pair.

Bin

Displays in binary.

Oct

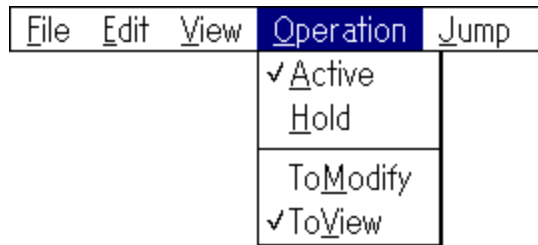
Displays in octal.

Dec

Displays in decimal.

Hex Displays in hexadecimal.

(d) **Operation**



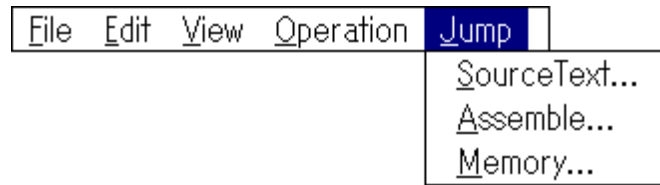
Active Switches the register window to the active state.

Hold Switches the register window to the hold state.

ToModify Switches the register window to the modify mode.

ToView Switches the register window to the view mode.

(e) **Jump**



Source Text With the selected register value as the jump destination address, the applicable source text and source lines are displayed. The source text window opens.

Assemble With the selected register value as the jump destination address, the disassemble window is displayed from that address. The disassemble window opens.

Memory With the selected register value as the jump destination address, the memory contents are displayed from that address. The memory window opens.


Function Buttons

button

Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the register can be changed.

When the modify mode is entered, the window's background color changes, and this button can no longer be selected.

To change the register, click the register you want to change. After the text cursor is displayed, changes can be typed in from the keyboard. The changes are fixed by clicking the  button.



button

Switches to the view mode.

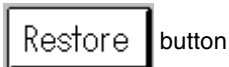
This button can be selected only when the window is in the modify mode. By clicking this button, the view mode can be moved to.

When the view mode is entered, the window's background color changes, and this button can no longer be selected.



button


Writes the changes.



button

Cancels the changes.

All of the values changed in the modify mode are restored to their original values.


However, if the  button has already been clicked, the values in a later state are restored.

SFR Window	Display/Setting Window
-------------------	-------------------------------

Overview

The SFR is displayed and changed.

This window can be opened by the following methods.

- In the main window
Select **Browse** → **Sfr...** in the menu bar.
- In the main window
Press in order the **GRPH**, **B**, and **F** keys.
- Click the  button in the tool bar.

Window

SFR Window			
<div> <div>ToModify</div> <div>ToView</div> <div>Write in</div> <div>Restore</div> <div>Close</div> </div>			
SFR Name	Atr.		
P6	R/W	1,8	0FFF06 00
P7	R/W	1,8	0FFF07 00
P0L	R/W	1,8	0FFF0E 00
P0H	R/W	1,8	0FFF0F 00
CR00	R/W	16	0FFF10 0000
CR01	R/W	16	0FFF12 0000
CR10W	R/W	16	0FFF14 FF00

Figure 5-40. SFR Window

Functions

The SFR contents are displayed and changed.

A read-only SFR is displayed in gray, and ones that cannot be changed are highlighted.

The SFR display can specify the display and reading method by the View @e Sfr item in the menu bar in the main window.

SFR Items	Description
<u>A</u> ddress Sort	The display order is specified. No check ' ' : Display in alphabetical order. Check '✓' : Display in the address order.
<u>P</u> ick Up	Displays only the changed SFR.
<u>A</u> tttribute	Specifies display or do not display the SFR attribute.
<u>C</u> ompulsion Read	Forcibly reads the read-protected SFR.
<u>S</u> ynchroniz	Reads the SFR again.

This window has a view mode and a modify mode.

The register window contains the following items.

- SFR name display area
- Attribute display area
- SFR contents display area
- Function buttons

Each function is described next.

(1) SFR name display area

SFR Name

```
P6
P7
P0L
P0H
CR00
CR01
CR10W
```

The SFR names are displayed.

(2) Attribute display area

Atr.

```
R/W 1,8 0FFF06
R/W 1,8 0FFF07
R/W 1,8 0FFF0E
R/W 1,8 0FFF0F
R/W 16 0FFF10
R/W 16 0FFF12
R/W 16 0FFF14
```

The SFR read/write attributes, access type, and address are displayed.

The attribute display can be selected by the View → **Sfr** → **A**tttribute item in the menu bar in the main window.

The types of read/write attributes are shown below.

Attribute	Description
R	Read-only SFR. This is displayed in gray.
W	Write-only SFR
R/W	Readable/writable SFR

The access types are shown below.

Access Type	Description
1	Bit-accessible SFR
8	Byte-accessible SFR
16	Word-accessible SFR

(3) SFR contents display area

```
00
00
00
00
0000
0000
FF00
```

The SFR contents are displayed and changed.

The display methods differ in the manner shown below based on the SFR attribute.

Read-only SFR: Displayed in gray.

Write-only SFR: "--" is displayed.

Readable/writable SFR: Displayed in black.

SFR whose value changes when read: "*" is displayed.

Changes can be made by pressing the **ToModify** button. Press the

Write in button after making the change to write the changes to the target.

Function Buttons

ToModify

button Switches to the modify mode.

This button can be selected only when the window is in the view mode. By clicking this button, the SFR contents can be changed.

When the modify mode is entered, the window's background color changes, and this button can no longer be selected.

To change the SFR contents, click the SFR display position you want to change. After the text cursor is displayed, changes can be typed in from the keyboard. The changes are fixed by clicking the **Write in** button.



button

Switches to the view mode.

This button can be selected only when the window is in the modify mode. By clicking this button, the view mode can be moved to.

When the view mode is entered, the window's background color changes, and this button can no longer be selected.



button


Writes in the changes.

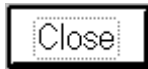


button

Cancels the changes.

All of the values changed in the modify mode are restored to their original values.

However, if the  button has already been clicked, the values in a later state are restored.



button

Closes the SFR window.

Icon

The SFR window can be displayed as an icon by clicking the



button in the title bar.

**SFR Window**

Save Dialog	Selection Dialog (Modal)
-------------	--------------------------

Overview

The contents of the current window when this dialog was opened are saved to the view file.

This dialog can be opened by the following methods.

When the window to be saved is a local variable window, disassemble window, memory window, stack trace window, SFR window, or trace view window

- In the main window
 1. The window to be saved becomes the current window.
 2. Select **File** → **Save As...** in the menu bar.
- In the main window
 1. The window you want to reference becomes the current window.
 2. Press in order the **GRPH**, **F**, and **A** keys.

When the window to be saved is a variable window

- In the variable window

Select **File** → **Open/save Condition** → **Save File As...** in the menu bar.
- In the variable window

Press in order the **GRPH**, **F**, **O**, and **A** keys.

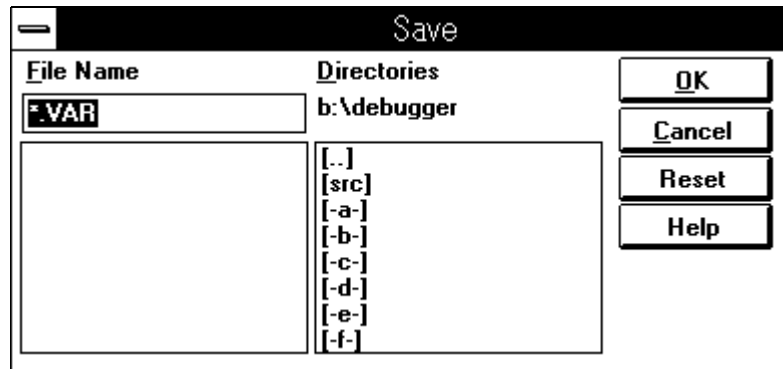
When the window to be saved is the register window

- In the register window

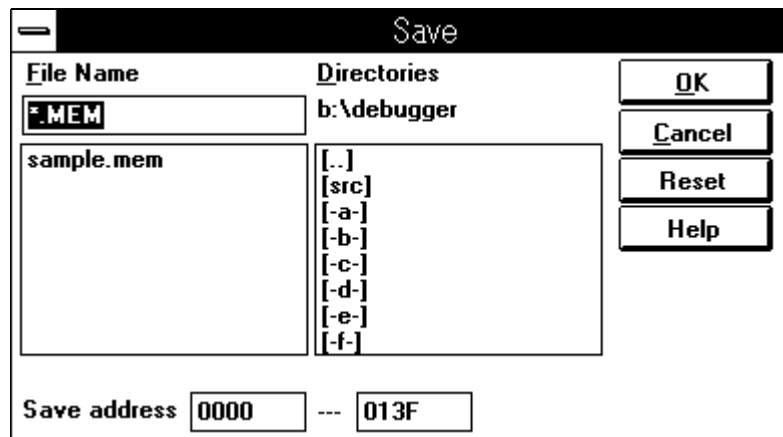
Select **File** → **Open/save Condition** → **Save File As...** in the menu bar.
- In the register window

Press in order the **GRPH**, **F**, **O**, and **A** keys.

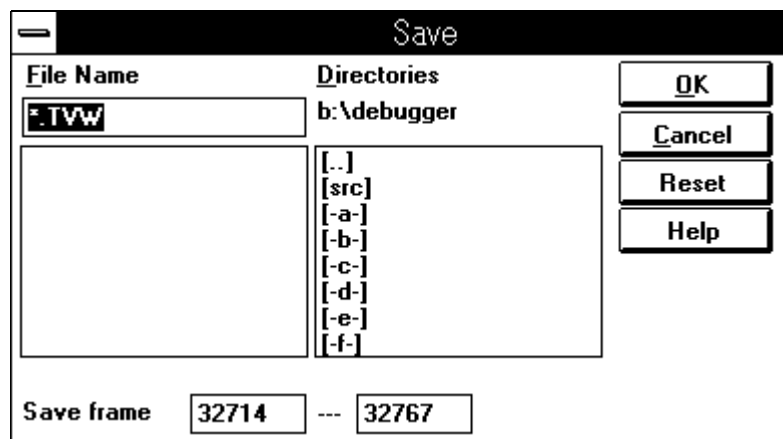
Window



When the window to be saved is a local variable window, disassemble window, variable window, stack trace window, SFR window, register window, or a window in the hold state



When the window to be saved is a memory window in the active state



When the window to be saved is a trace view window in the active state

Figure 5-42. View File Save Dialog

Functions

The display contents of the current window are saved in the view file.

The view file save dialog contains the following items.

- File selection area
- Path setting area
- Save range setting area
- Function buttons

Each function is described next.

(1) File selection area

File Name

*VAR

The file name of the file to be saved is specified.

The view file is selected by clicking the desired view file in the view file list.

The selected file name is highlighted and displayed in the selected view file display area.

In the view file list, double clicking the file name or clicking the

OK

button have the same effect.

The default extensions are listed below.

Window	Default Extension
Variable window	VAR
Local variable window	LOC
Disassemble window	DIS
Memory window	MEM
Register window	REG
Stack trace window	STK
SFR window	SFR
Trace view window	TVW
Event manager	EVN

(2) Path setting area

Directories

b:\debugger

[..]
[src]
[-a-]
[-b-]
[-c-]
[-d-]
[-e-]
[-f-]

The path of the view file to be saved is specified.

By double clicking the desired path name, the view files in the path are displayed in the view file display area.

The display formats are

[xxx] : Indicates a directory name

[-x-] : Indicates a drive name

(3) Save range setting area

This area is displayed only when the current window to be saved is a memory window or a trace view window.

a. When the current window is a memory window

Save address ~

The address range to be saved is specified. The address specification can be specified by symbols. The specification method is shown below.

Function and Variable	<u>_</u> fnc file#_fnc (for a static function and variable)
Line number in the source text	file:no

fnc: function, variable name; file: file name; no: line number

If a function or variable name is specified, an underline (_) is specified at the beginning. The sharp (#) is used as the separator between a file name and a function or variable name. The colon (:) is the separator between a file name and a line number.

b. When the current window is a trace view window

Save frame ---

The range of the trace frames to be saved is specified.

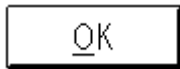
Specification range: $0 \leq \text{Frame number} \leq 32,767$

If a range above 100 frames was specified, the following message dialog appears, and the save state can be gradually understood. If you want to stop the save while it's operating, it can be stopped by pressing the



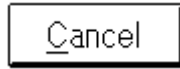
button in the message dialog.



Function Buttons

button

Writes to the view file to be selected.



button

Closes the dialog.



button

Returns to the initial state.



button

Opens the help window.

Error/Warning Dialog	Confirmation Dialog (Modal)
----------------------	-----------------------------

Overview

If an error or a warning occurs, the message is confirmed.

Window

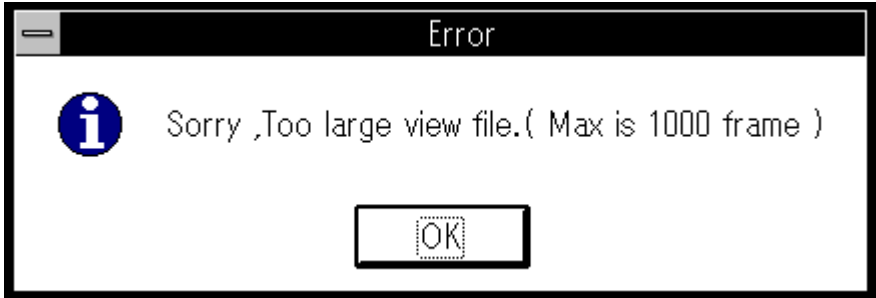
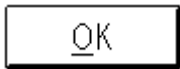


Figure 5-43. Error/Warning Dialog

Functions

The error or warning is displayed in the message display area.

Function Buttons

 button Closes the dialog.

Reset Debugger Dialog**Confirmation Dialog (Modal)****Overview**

The debugger itself and the emulation CPU are initialized.

This dialog can be opened by the following methods.

- In the main window
Select **E**xecute → **C**PU Reset... in the menu bar.
- In the main window
Press in order the **G**RPH, **X**, and U**U** keys.

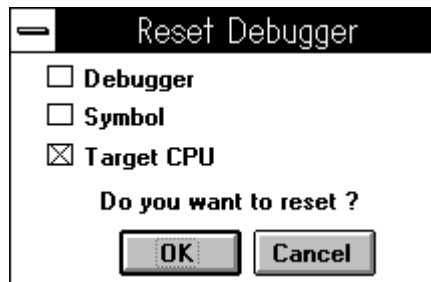
Window

Figure 5-44. Reset Confirmation Dialog

Functions

The initialization target is specified by a radio button. By default, only the emulation CPU is set to be initialized. The reset confirmation dialog contains the following items.

- Reset target selection area
- Function buttons

Each function is described next.

(1) Reset target selection area

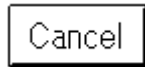
- ☐ **Debugger**
- ☐ **Symbol**
- ☒ **Target CPU**

The initialization target is selected.

Selection	Description
Debugger	Restarts the debugger.
Symbol	Initializes all of the loaded and registered symbol data.
Target CPU	Resets the emulation CPU.

Function Buttons

button Initializes according to the selected item.



button Closes the dialog.

About Dialog**Display Dialog (Modal)****Overview**

Version information about the debugger is displayed.

This dialog can be opened by the following methods.

- In the main window
Select **H**elp → **A**bout... in the menu bar.
- In the main window
Press in order **G**RPH, **H**, and **A**.

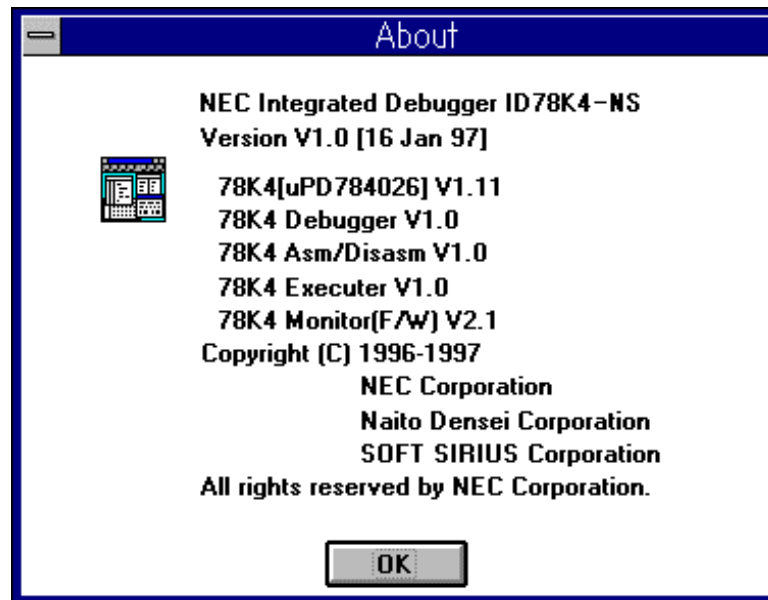
Window

Figure 5-45. Version Display Dialog

Functions

Version information about the debugger is displayed.

The version information displays the versions of the debugger and the device files.

Function Buttons

button Closes the version display dialog.

Exit Debugger Dialog	Confirmation Dialog (Modal)
----------------------	-----------------------------

Overview

The debugger exits.

When the debugger exits, the debugging environment can be saved in a project file.

This dialog can be opened by the following methods.

- In the main window
Select **F**ile → **E**xit in the menu bar.
- In the main window
Press in order **G**RPH, **F**, and **X**.


Window



Figure 5-46. Exit Confirmation Dialog

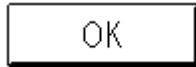
Functions

When the debugger exits, the current debugging environment can be exited after saving in a project file or without saving. This can be selected by a toggle button. The default is not to save.

If save is selected and the  button is pressed, the project file save dialog opens. After the current debugging environment is saved in a project file, all of the windows are closed and the debugger exits.

If not saving is selected and the  button is pressed, all of the windows are closed and the debugger exits.

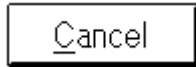
Function Buttons



button

If saving is selected, the project file save dialog opens. After the current debugging environment is saved in a project file, all of the windows are closed and the debugger exits.

If saving is not selected, all of the windows are closed and the debugger exits.



button

Closes the dialog without doing anything.

[MEMO]

CHAPTER 6 DEBUGGER FUNCTION OVERVIEW

This chapter describes in detail each function of the integrated debugger.

6.1 System Operating Modes

The system operating mode indicates whether the user program execution (emulation) function and the analyzer function are running and the operating states of the system.

6.1.1 Types of operating modes

The three operating modes for the system operating mode are given below. Each command is restricted by the system operating mode.

Break mode

In this state, both the user program execution (emulation) function and the analyzer function stop.

Emulation mode

In this state, the user program execution (emulation) function runs, but the trace function stops. This mode is used when you do not want to stop the execution of the user program. However, analyzer functions (timer measurement) other than the tracer run.

Trace mode

In this state, both the user program execution (emulation) function and the analyzer function run.

6.1.2 System Operating Modes

The system operating modes can be distinguished by the status bar in the main window.

System Operating Mode	CPU Operation	Tracer Operation
Break mode	Stop	Stop
Emulation mode	Run	Stop
Trace mode	Run	Run

6.1.3 System operating states

Refer to the following figure for the correlation relationships of the functions of the emulation CPU and the analyzer. An example of this correlation relationship is presented.

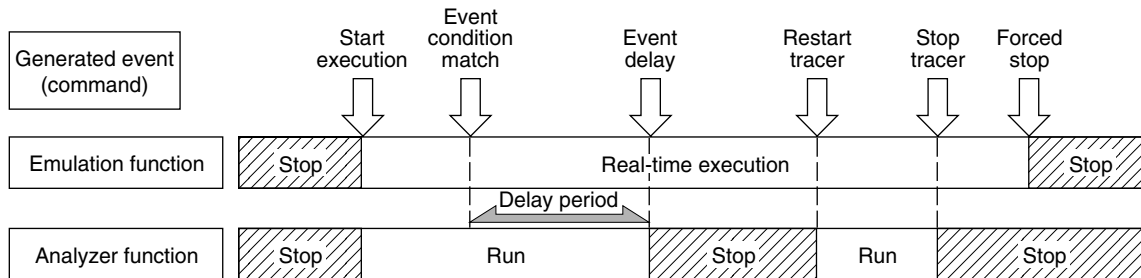


Figure 6-1. Example of the System Operating States

6.2 Using the Basic Functions

The basic functions of the abundant debugging functions used in the debugger are described.

6.2.1 Clock selection function

The clock selection function specifies the clock source to be supplied to the emulation CPU (target device). The following two types can be selected as the source clock.

Socket mounted clock in the in-circuit emulator (Internal)

Clock of the user target supplied by the probe (External)

This setting can be made when the debugger starts or in the configuration dialog.

The two methods for supplying any clock of the user setting are

- the clock is generated by the target system, passed through the emulation probe, and supplied to the emulation CPU. External is set in the clock selection.
- the oscillator in the clock installation socket on the I/O board in the in-circuit emulator unit is replaced to supply the clock to the emulation CPU. Internal is set in the clock selection.

If the clock source is changed, the emulation CPU is reset.

6.2.2 Mapping functions

The six mappings are described below. The following settings can be made for the address region other than the internal ROM and the SFR. These settings can be made when the debugger starts or in the configuration dialog.

Internal ROM

The memory range specified in the internal ROM becomes the memory space equivalent to the internal ROM of the target device.

In this case, the target device accesses the memory in the in-circuit emulator.

When this memory space is written, the target device generates the write protect break.

Internal RAM

The memory space specified in the internal RAM becomes the memory space equivalent to the internal RAM in the target device.

In this case, the target device accesses the memory in the in-circuit emulator.

User area mapping (Target)

The memory space specified in user area mapping accesses the memory in the target system.

In this case, the target device accesses the memory in the target system.

Substitute ROM (Emulation ROM)

The memory space specified in the substitute ROM becomes the same memory space as when ROM is connected to the target device.

In this case, the target device accesses the memory in the in-circuit emulator.

When this memory space is written, the target device generates a write protect break.

Substitute RAM (Emulation RAM)

The memory space specified in the substitute RAM becomes the same memory space as when the RAM is connected to the target device.

In this case, the target device accesses the memory in the in-circuit emulator.

6.2.3 Reset function

The reset function resets the entire system of the in-circuit emulator or the emulation device.

Reset the entire system (Debugger)

Reset only the emulation device (Target CPU)

This function can be specified in the reset confirmation dialog.

6.2.4 Load function

The load function separately loads the specified file contents, such as the debugging environment, object files, load module files, and symbol files.

The two types of files to be loaded are the view file for screen reference and the data file that updates the data in the debugger.

The view file records saved screen data. By loading the view file, the reference window opens.

The view files are as follows.

File	Window	Description
Variable view file (File name: XXXXXXXX.VAR)	Variable window	Stores the variable data.
Disassemble view file (File name: XXXXXXXX.DIS)	Disassemble window	Stores the disassemble data.
Memory view file (File name: XXXXXXXX.MEM)	Memory window	Stores the memory data.
Register view file (File name: XXXXXXXX.REG)	Register window	Stores the register data.
Stack trace view file (File name: XXXXXXXX.STK)	Stack trace window	Stores the stack trace data.
SFR view file (File name: XXXXXXXX.SFR)	SFR window	Stores SFR data.
Local variable view file (File name: XXXXXXXX.LOC)	Local variable window	Stores the local variable data.
Trace view file (File name: XXXXXXXX.TVW)	Trace view window	Stores trace data.

The data files are as follows.



File	Window	Description
Object file (File name: XXXXXXXX.HEX)	Load module selection dialog	Stores the object code (Motorola, Intel) of the user program.
Symbol table file (File name: XXXXXXXX.SYM)	Load module selection dialog	Stores the symbols defined in the source by the user for the user program.
Load module file (File name: XXXXXXXX.LNK)	Load module selection dialog	Stores the object code and symbols of the user program and the source data.
Project file (File name: XXXXXXXX.PRJ)	Project file load dialog	Stores the debugging environment. The data in the following windows are set by this file. <ul style="list-style-type: none"> • Configuration dialog • Extended option setting dialog • Load module selection dialog • Source text window • Source path specification dialog • Disassemble window • Memory window • Stack trace window • SFR window • Local variable window • Trace view window • Event manager • Event link dialog • Break dialog • Trace dialog • Timer dialog • Register window • Variable window
Event setting file (File name: XXXXXXXX.EVN)	Event manager	Stores the event setting data.

6.2.5 Emulation execution function



The emulation execution function starts the user program execution (emulation) by the emulation CPU and the analyzer.

The functions are classified in the following way by the execution state of the emulation.

Real-time execution functions

Go ( button)	Executes in real time. Breaks when a break event is generated.
Return ( button)	Executes in real time until returning to the calling function.
Go & Go	Executes in real time. When a break event was generated, repeats the execution in real time after the break and the window is updated.
Come	Executes in real time until the target address or source line. A break event is not generated during execution.
CPU Reset & Go	Executes in real time after the emulation CPU is reset.

Non-real-time execution functions

Step ( button)	When the source mode is selected Step executes at the source level. When the instruction mode is selected Step executes at the instruction level.
Next ( button)	When the source mode is selected Next step executes in the source level. When the instruction mode is selected Next step executes in the instruction level.
Slowmotion	Continuously step executes.

Real-time execution function

In real-time execution, there are "Go" that executes the user program until a break event is generated, "Go & Go" that updates each window even if a break event is generated and executes the user program again, "Come" that executes until the specified point and breaks, and "Return" that executes the user program until returning to the calling function.

Go command (button)

Real-time execution by the Go function executes the user program from the specified address and stops the execution of the user program when a break event is generated. Each analyzer executes the program and enters the operation enabled state, and executes or enters the stop state based on each event.

The following figure shows the correlation relationship between the CPU and the tracer in real-time execution by the Go command.

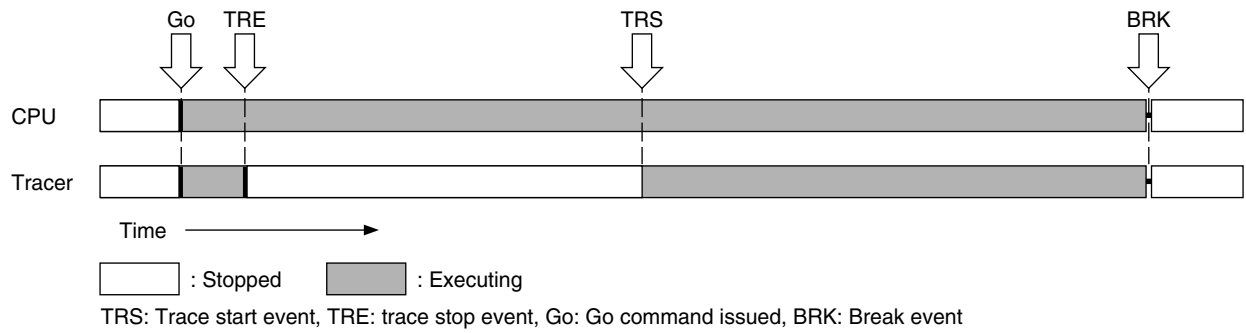


Figure 6-2. System Operation State (Go)

Return command

Real-time execution by the Return command executes in real time until returning to the calling function. If there is no calling function, nothing happens.

The following figure shows the concept of real-time execution by the Return command.

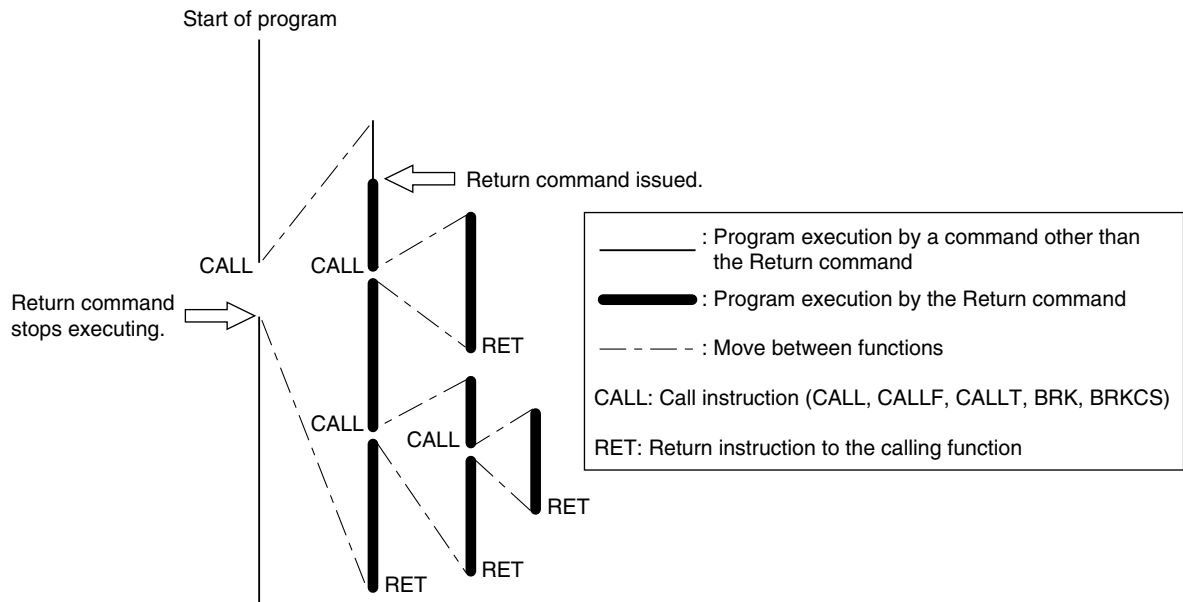


Figure 6-3. Conceptual Diagram of Return Command Execution

The Return command sets an execution break at the return address of the function and executes in real time. The following figure shows the correlation relationship between the CPU and the analyzer when executing in real time by the Return command.

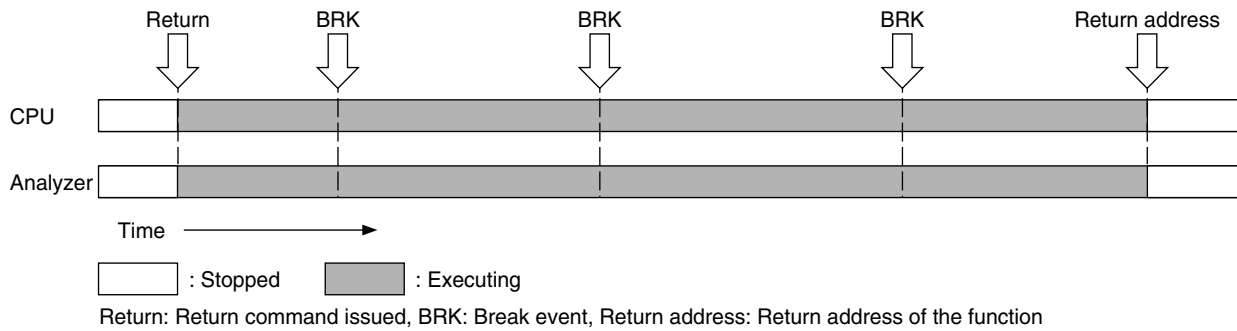


Figure 6-4. Example of the System Operating State (Return)

Go & Go command

Real-time execution by the Go & Go command

- (1) The user program is executed from the specified address.
- (2) When a break event is generated, the program stops executing.
- (3) The screen of each window is updated.
- (4) Execution begins again from the address where the program stopped.
- (5) Repeat (2), (3), and (4) until the Stop command is issued.

Each analyzer executes a program and enters the operation enabled state, and executes due to each event or enters the stop state.

The following figure shows the correlation relationship between the CPU and the analyzer in real-time execution by the Go & Go command.

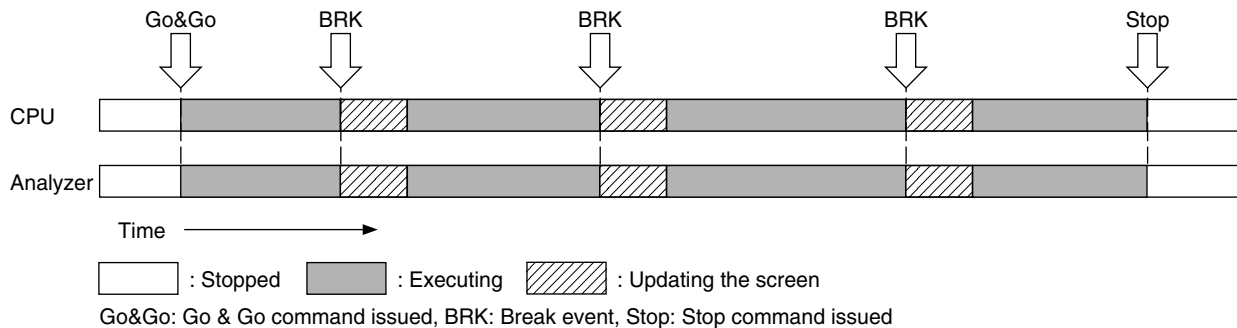


Figure 6-5. Example of the System Operating State (Go & Go)

Come command

Real-time execution by the Come command

- (1) Move the cursor to the location where you want to stop the program execution in the source text window or the disassemble window.
- (2) Issuing the Come command executes the user program from the address in the PC register.
- (3) The program executes until the address specified by the cursor and breaks.

A break is not caused by a break event during program execution.

The following figure shows the correlation relationship between the CPU and the analyzer during real-time execution by the Come command.

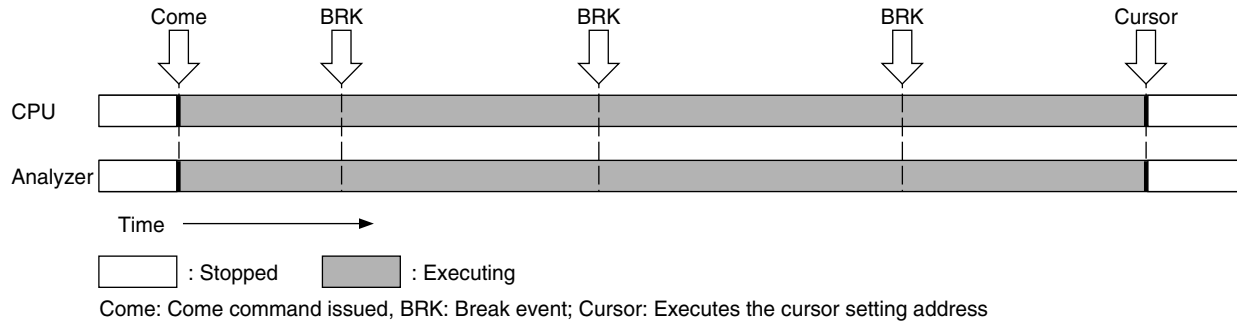


Figure 6-6. System Operating State (Come)

CPU Reset & Go command

CPU Reset & Go execution

- (1) The emulation CPU is reset.
- (2) The program is executed by the reset vector.

The operation before a program is executed and after the emulation CPU is reset is the same operation as the Go command.

The following figure shows the correlation relationship between the CPU and the tracer in the CPU Reset & Go execution.

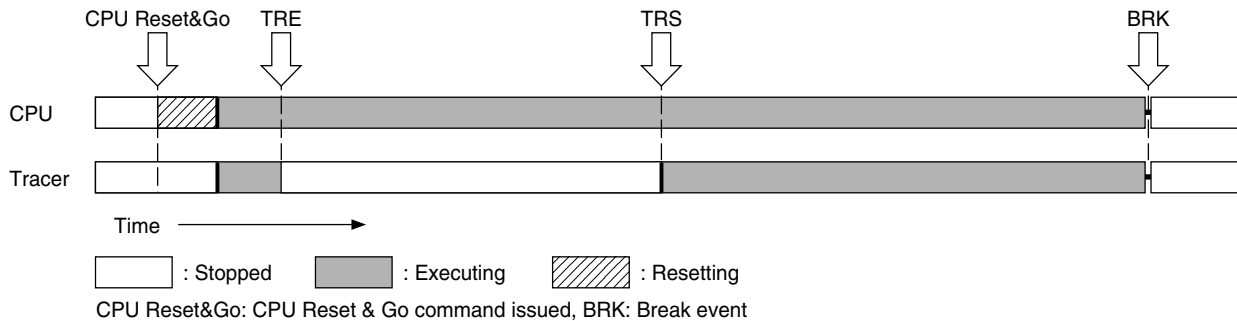


Figure 6-7. Example of the System Operation State (CPU Reset & Go)

Non-real-time execution functions

The non-real-time execution functions are broadly classified into "Step" that executes in steps, "Next" that executes the next step, and "Slowmotion" that continuously executes in steps.

Step command

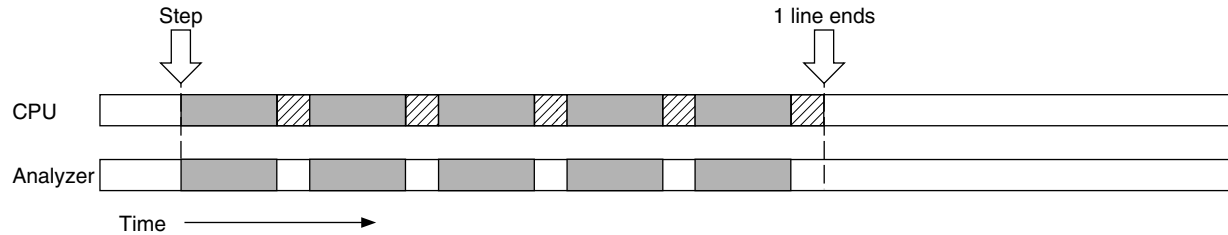
Step execution by the Step command

- In the source mode
Step executes in one line segments starting from the specified source line.
- In the instruction mode
Executes one instruction at the specified address.

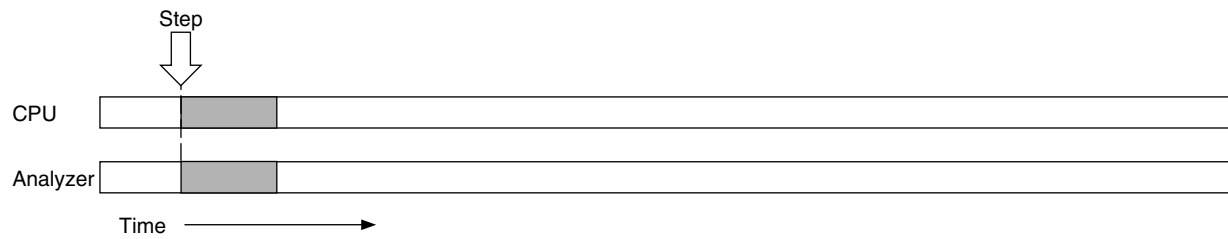
After execution, each window is updated.

The following figure shows the correlation relationship between the CPU and the analyzer in step execution by the Step command.

In the source mode



In the instruction mode



□ : Stopped ■ : Executing 1 instruction ▨ : End of step execution confirmed

Step: Step command issued, 1 line ends: End of instruction execution of one line

Figure 6-8. Example of the System Operation State (Step)

Next command

Next step execution by the Next command operates differently when a call statement is executed and when a statement other than a call statement is executed.

The call statement becomes the following instructions depending on the debugging mode.

- In the source mode
 - Line calling the function
- In the instruction mode
 - CALL, CALLF, CALLT, BRK, BRKCS instructions

The operation of the Next command is shown below.

- When executing the call statement
 - The execution break is set in the "line" or "instruction" following the call statement, and executes in real time.
- When executing a statement other than the call statement
 - The same process as the Step command is performed.

The concept of next step execution by the Next command is shown below.

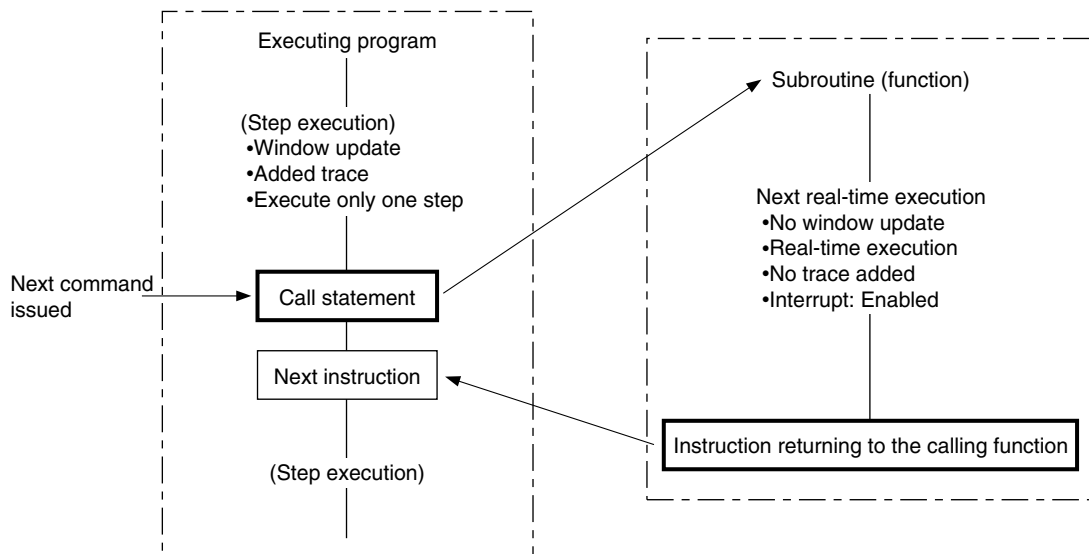


Figure 6-9. Conceptual Diagram of Next Step Execution

Slowmotion command

Step execution by the Slowmotion command

- (1) The debugging mode from the specified address executes in steps of one line units if in the source mode or in one instruction units if in the instruction mode.
- (2) Each window is updated.
- (3) Repeat (1) and (2) until the Stop command is issued.

6.2.6 Break function

The break function stops the execution (emulation) of the user program by the emulation CPU and stops the analyzer (tracer).

The four types of break functions are broadly divided into classes listed below.

- Event detected break
- Break caused by satisfying a condition during step execution
- Forced break
- Fail-safe break

The relationship between these break functions and the emulation execution functions are given below.

	Event detected break	Break caused by satisfying a condition during step execution	Forced break	Fail-safe break
Real-time execution by the Go command	○	×	○	○
Real-time execution by the Go & Go command	○	×	○	○
Real-time execution by the Come command	×	×	○	○
Real-time execution by the CPU Reset & Go command	○	×	○	○
Non-real-time execution by the Step command	×	○	○	○
Non-real-time execution by the Return command	×	○	○	○
Non-real-time execution by the Next command	×	○	○	○
Non-real-time execution by the Slowmotion command	×	×	○	○

Event detected break

An event detected break is a function that stops the execution of a user program by detecting the specified event condition.

This break is valid for the Go command, Go & Go command, and CPU Reset & Go command.

However, after an event detected break in the Go & Go command, each window is redrawn and the program is executed again.

The event detected conditions must set the break events in the event dialog, event manager, and break dialog.

Break caused by satisfying a condition during step execution

A break by satisfying a condition during step execution is a function that stops program execution by satisfying the stop condition of the each command (Step, Next, Slowmotion). In order to repeat the execution, stopping, and condition confirmation for each instruction, the processing time is delayed compared to real-time execution.

Forced break

A forced break is a function that forcibly stops the execution of a user program. This is valid for all of the commands executed in the program.

The two types of forced breaks are

1. Stop command

Forcibly stops the execution of the user program.

2. Reset command

After the execution of the user program is forcibly stopped, the device is reset.

If you want to temporarily stop a program, the Stop command is effective. If you want to execute a program from the beginning, the Reset command is effective.

Fail-safe break

The fail-safe break is a function that forcibly stops the execution of a program when the user program is prohibited from using the memory and the registers.

The three types of fail-safe breaks are

1. Nonmapping break

Generated when a nonmapping region is accessed

2. Write protect break

Generated when writing to a memory that cannot be written, such as ROM

3. Illegal SFR access break

Generated when an illegal access to the SFR region occurred

When a fail-safe break occurred, the two possibilities are a problem in the user program or a mistake in the environment settings of the debugger.

Caution When a program is written near the boundary between a mapping region and a nonmapping region, a nonmapping break is generated.

Danger of generating a nonmapping break

Maximum address of the mapping region - 5 ≤ Program address ≤ Maximum address of the mapping region

Example: Mapping region: 0x00000 to 0x03FFF, Nonmapping region: Above 0x04000

Sometimes a nonmapped break is generated when the program was written to addresses 0x3FFA to 0x3FFF.

All of these are related to the prefetch and queue buffer and generated.

6.2.7 Trace functions

A trace function accesses the memory during the execution of the user program and writes in real time data such as external sense clip values to the trace memory.

With the data written in the trace memory, the execution process of the target program can be examined by opening the trace view window.

The main functions related to trace execution and trace display are given below. The trace conditions can be set in the trace dialog. The settings for the trace data display can be specified in the trace pick up dialog or by **View → Trace View** in the menu bar in the main window.

Trace operation

- Operation during real-time execution
- Operation during step execution
- Operation during next step execution

Trace condition setting function (trace dialog)

- Trace mode specification
- Qualify trace setting
- Section trace setting

Trace data display, format, and search condition settings

- Trace data display specification
- Trace data search condition setting

The relationship between trace execution and the trace memory

The trace is divided into trace blocks according to the periods shown below.

- (1) Block from real-time execution to a break by an event
- (2) Block from emulation execution until the generation of a fail-safe break
- (3) Block from emulation execution until a forced break
- (4) Step execution block

The trace memory is a 32k frame ring buffer. Therefore, if 32k frames are exceeded during a trace, the latest trace data is overwritten in the oldest frame.

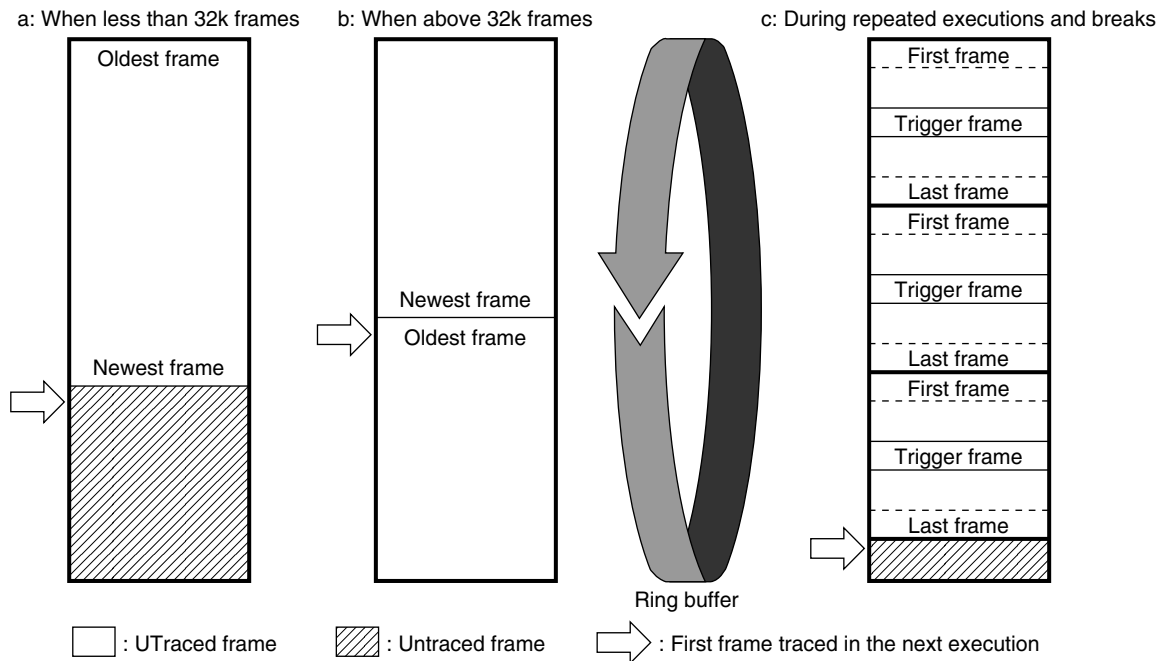


Figure 6-10. Trace Memory Concept

Trace operation

The tracer operates differently depending on the execution state.

Operation during real-time execution

The tracer starts the trace at the specification of the real-time execution. When the event conditions, including delay conditions, specified in the break conditions of the trace dialog are set up, the trace operation ends.

Operation during step execution

The tracer runs for each step execution. The trace data in one step is added to the tracer for each step execution.

Operation during next step execution

When executing an instruction other than a call instruction (CALL, CALLF, CALLT, BRK, BRKCS)

The operation is identical to the operation during step execution.

When executing a call instruction (CALL, CALLF, CALLT, BRK, BRKCS)

The operation is identical to the operation during real-time execution.

Real-time execution is stopped by returning to the calling function.

Trace condition setting function (trace dialog)

The following specifications can specify the trace conditions. If these specifications are not made, "All trace" is performed. In other words, the trace data are recorded for each instruction in the user program.

Specifying the trace mode

A complete trace or which trace of a conditional trace is specified. The two types in a conditional trace are the qualify trace and the section trace.

Finally, the specified trace mode becomes valid.

Qualify trace setting

This specifies a trace only when the specified address was executed or the specified address was accessed. The specified conditions are created in the event setting window.

Section trace setting

This specifies starting the trace with the specified trace start condition and stopping the trace with the specified trace end condition. Specifically, this specifies a trace with a range specification. The condition to the specified is created in the event setting window.

Since this also serves as the setting of measurement range of the run time measurement, enable this event when the run time measurement function is used.

Setting the trace data display, format, and search conditions

The data can be displayed or hidden in the trace view window, and the display conditions can be set.

Trace data display setting

The display screen can be effectively used by specifying the display of the trace data. Trace data display can specify displaying or hiding the data by setting **View** → **Trace View** in the menu bar in the main window.

Table 6-1. Description of the Trace Data Display

Items in the Menu Bar	Item Display in the Trace View Window	Description
Frame number (E)	Frame	Temporal order written to the trace memory by the frame number in the trace memory (range from 00000 to 32767)
Instruction fetch address (A)	Addr	Fetch address
Instruction fetch data (D)	Data	Fetch data
Instruction fetch status (U)	Statu	Fetch status <ul style="list-style-type: none"> • M1 Fetch the first byte of the instruction • BRM1 Fetch the first byte of the first instruction after a branch • OP Opcode fetch • NON Illegal fetch
Memory access address (R)	Addr	Access address
Memory access data (M)	Data	Access data
Memory access status (S)	Statu	Access status <ul style="list-style-type: none"> • VECT nterrupt process • RWP Data read or write by a user program • RP Data read by a user program • WP Data write by a user program • RWM Data read or write by a macro service • RM Data read by a macro service • WM Data write by a macro service
Disassemble (I)	DisAsm	Disassemble result

Setting the search conditions for trace data

The search conditions for trace data can be specified.

The search conditions can be selected and specified by all of the items or any of the items in the following table in the trace pick up dialog. The specified data are enabled by the **Pick up ON** button.

Table 6-2. Trace Search Items

Specification Item	Description	Specified Range	Default
Address	Search address	0-0FFFFFFFH	0XXXXXXH
Data	Search data	0-0FFFFFFFHH	0XXXXXXXXH
Status	Search status <ul style="list-style-type: none"> • All status : All of the status • M1 : Fetch the first byte of the instruction • BRM1 : Fetch the first byte of the first instruction after a branch • OP : Opcode fetch • NON : Invalid fetch • VECT : Interrupt process • RWP : Data read or write by a user program • RP : Data read by a user program • WP : Data write by a user program • RWM : Data read or write by a macro service • RM : Data read by a macro service • WM : Data write by a macro service 	Same as on left.	All status
Kind of frame ?	Search data type <ul style="list-style-type: none"> • All Frame : All of the frames • Step : Step execution frames • Next : Frames other than step execution frames 	Same as on left.	All Frame

6.2.8 Event setting and detection function

The event setting and detection function sets the conditions for stopping the execution of the user program by the emulation CPU and for starting and stopping the trace operation by the analyzer.

The five types of event condition setting and detection functions are

- Event detected condition setting function

 - Bus event condition setting function

 - Execution event condition setting function

 - Event condition link setting function

- Integrated function of the event detection function

 - Break event setting

 - Trace event setting

Event condition setting function

The event condition setting function sets the event condition register with the condition to stop the execution of a user program and the condition to start or stop a trace by the analyzer. The event detected condition specified by the setting function for the event detected condition (event dialog, event link dialog) does not finally become valid (state for event generation) when not set in the event mode register by the event detected condition integrated function (event manager, break dialog, trace dialog).

The three types of functions set by the event detected condition are as follows.

Bus event condition setting function

The user program accessing the specified memory or inputting data to an external sense clip can be set in the bus event condition register as the event detected condition.

(a) Bus event condition register

A maximum of four conditions can be set in the bus event condition register (BRA) in the event dialog.

(b) Event condition

The following items can be set in the event detected condition.

Item	Status	Description
Address	Address	Address(address range)
	Mask	Address mask
Status	Fetch	Program fetch
	Program Read	Read by a program
	Program Write	Write by a program
	Program R/W	Read or write by a program
	Macro Read	Read by a macro service
	Macro Write	Write by a macro service
	Macro Read/Write	Read or write by a macro service
	Program/Macro Read	All reads
	Program/Macro Write	All writes
	Program/Macro R/W	All reads and writes
	BRM1	First instruction after a branch
	VECT	Vector read by an interrupt
	ALL (No Condition)	All accesses
Data	Data	Data value
	Mask	Data mask value
Data Size	Byte	Byte data size
	Word	Word data size
	ALL (No Condition)	Byte or word data size
Pass count	Pass count	Pass count

Caution When Fetch was selected as the Status condition, an event is generated after executing the fetch. Even if a program is not executed, an event is generated by the prefetch. If you want to generate an event by program execution, use the execution event condition setting function.

Execution event condition setting function

The user program executing the instruction at the specified address and inputting the data for the external sense clip at that time can be set in the execution event condition register as the event detected condition.

(a) Execution event condition register

A maximum of four conditions can be set in the execution event condition register (BRS) in the event dialog.

(b) Event condition

The following items can be set in the event detected condition.

Item	Status	Description
Address	Address	Address (address range)
	Mask	Address mask
Status	Run	Program execution
Data	Data	Data value
	Mask	Data mask value
Pass count	Pass count	Pass count

Event condition link setting function

The event conditions registered in the event dialog can register the event connect conditions in the event link dialog.

Concepts Behind Event Detection

The concepts from setting the event conditions to event detection are shown below.

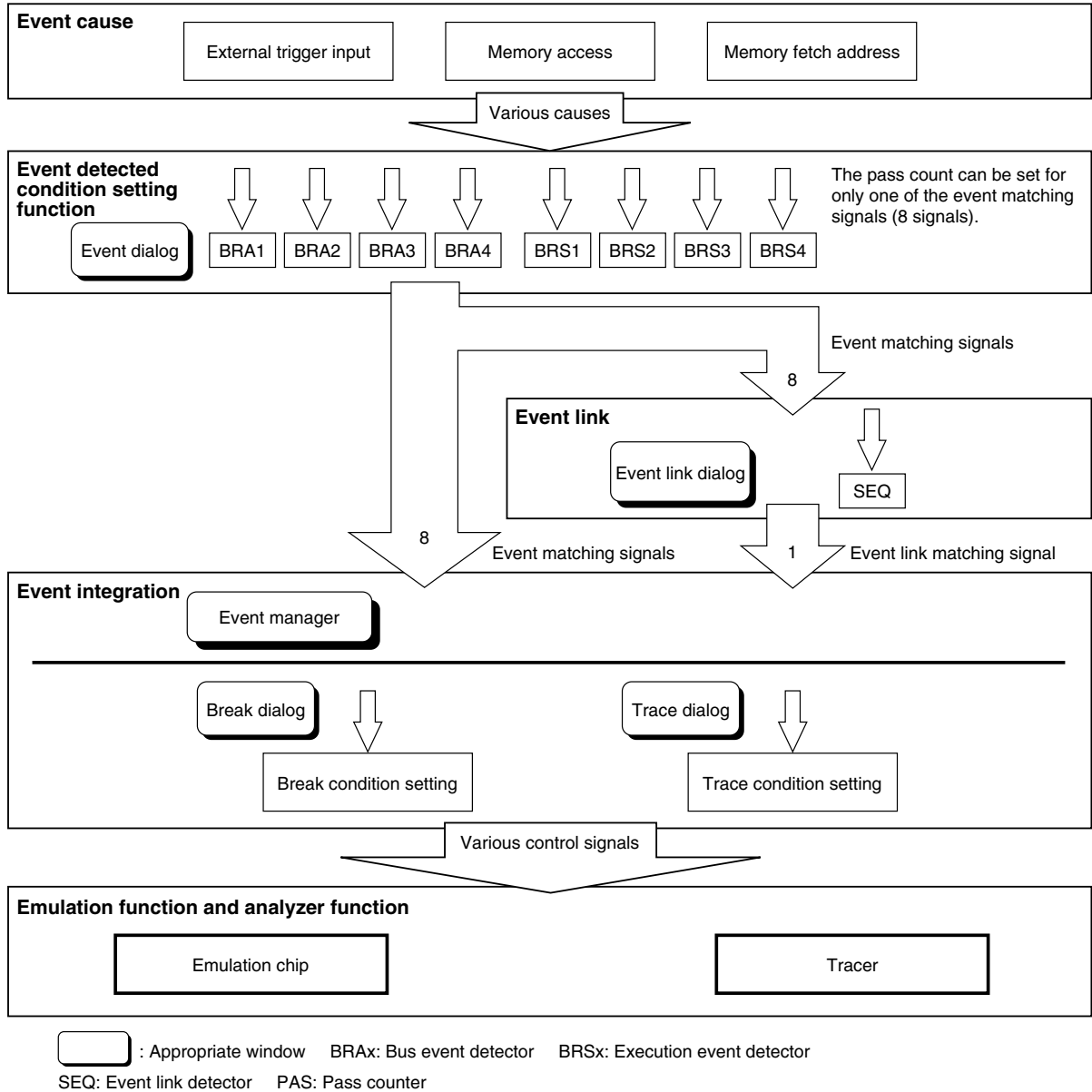


Figure 6-11. Concepts Behind Event Detection

6.2.9 Register manipulation functions

The register manipulation functions display the contents of the general-purpose registers and the SFR, and change the contents. The major functions are shown below.

(1) General-purpose manipulation function (register window)

This function displays and changes the contents of the control registers and general-purpose registers.

- Control registers: PC, SP, PSW
- General-purpose registers: RP0, RP1, RP2, RP3, RG4, RG5, RG6, RG7, AX, BC, VVP, UUP, TDE, WHL

Even the PSW flag names listed below are displayed or changed for PSW.

- PSW flag name: UF, RSB, S, Z, RSS, AC, IE, P/V, CY

(2) Special register manipulation function (SFR window)

This function displays and changes the contents of the special register (SFR). In addition, the SFR can be manipulated by bits.

6.2.10 Memory manipulation functions

The memory manipulation functions uses mnemonic codes, hexadecimal codes, and ASCII characters to change the memory contents. These functions can be used in the assemble window and memory window.

6.2.11 Save function

The save function stores the object codes in the in-circuit emulator and the debugging environment in a file on a disk drive connected to the host machine.

6.2.12 Time measurement function

The entire run time until a break after execution begins and the interval measurement from event to event can be measured.

The time measurement can measure the accumulated run time, average run time, and time measurement count. The timer specifications are shown below.

Item	Contents
Accumulated run time	203.45 nanosecond resolution Maximum 14 minutes, 33 seconds
Time measurement count	Maximum 65,535 times

6.2.13 Source debugging

If ID is used for debugging, not only object programs, but source programs can be debugged. Consequently, debugging which uses the source program is called source debugging.

Debugging which uses source programs has the following advantages over debugging which uses object programs.

- Debugging is possible while examining the C or structured assembler source actually written using the editor.
- Breakpoints can be set in the source and step execution can be performed.

For example, if a breakpoint is set, generally, the real address of the breakpoint is specified. However, in source debugging, the position where a breakpoint will be set is specified in the source program by the mouse.

In step execution, the line currently being executed in the source program is indicated by the ">" mark. Therefore, program operation can be understood more accurately.

Source debugging is particularly effective when debugging programs written in the C language or in structured assembler.

In source debugging, note the following points.

- (1) If assembling or compiling, the options must be specified to include the source debugging data in the object.

Type of Source for Source Debugging	Required Action
C program	Specify the -G option when compiling.
Structured assembler program	Specify the -GS option in structured assembler
Assembler program	Specify -GA option when assembling

Link	Specify the -G option when linking
------	------------------------------------

- (2) Specify the path data for storing source program in the source path specification dialog.
- (3) In source debugging, always load the load module file created when linking. Even if the object file created by the object converter is loaded, source debugging is not possible.

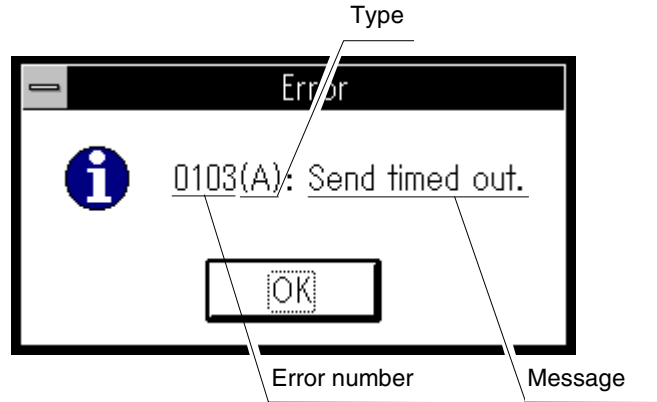
This section describes how to install the debugger (for both the PC-9801, 9821 and the IBM-PC/AT).

[MEMO]

APPENDIX A ERROR MESSAGES

This appendix lists the errors and warnings output by ID.

An error message consists of Error number + Type + Message.



The type is represented by one letter and has the following three types.

Type	Meaning
A	Indicates a fatal error. (<u>A</u> bort error) The processing is stopped and debugging ends. If this error was generated, debugging cannot be continued.
F	Indicates a syntax error. (<u>F</u> atal error) The processing stops. The opened windows and dialogs are closed.
W	Indicates a warning. (<u>W</u> arning) The processing stops. The opened windows and dialogs are not closed.

In these messages, the file names, variable names, and device names that are the target errors are listed below.

Message	Meaning
xxx	Inputs 3 digits in the device name.
yyy	File name
zzz	Function name

Error Message List (1/9)

Error Number	Type	Message	Meaning
–	–	Can't open this file. Please make sure, not Active Window.	Illegal format of the project file or corrupted file contents. Loading the project file stopped.
–	–	Cannot find "string"	Did not find the search string. The search stops. Or if the specified file did not contain data, the file open stopped.
–	–	Event Name is not set.	No event name After setting the event name, register the event.
–	–	Even number already exist.	An event with the same number cannot be registered. Change the number of the event being registered, or change the number of the event that was already registered with the same number.
–	–	Not enough memory.	There is insufficient memory to display or change the window, or save the changes. After freeing more memory, execute again.
–	–	Other view mode window exist.	Two or more active windows having the same type cannot be opened simultaneously. Other active windows were closed.
–	–	Sorry, Too large view file. (Max is 1000 frames)	The content of the specified view file (.MEM, .TVW, .DIS) is longer than 1,000 lines. The display was stopped.
–	–	"Event-name" already exist.	An event with the same name cannot be registered. Change the name of the event to be registered or change the name of the event already registered with the same name.
0103	A	Send times out	Data cannot be sent to the in-circuit emulator. Check for possible causes such as the setting of the interface board, or no power being applied to the IE. After rechecking, restart the debugger.
0104	A	Receive timed out	No response from the IE The error may be in the IE. After checking the IE, restart the debugger.
0105	A	Invalid D4xxx. 78K	The device file (D4xxx.78K) cannot be properly read. The device file is not in the specified directory, or the device file is corrupted. Reinstall the device file and start.
01a0	A	Monitor timed out	Data communication with the IE is not possible. The clock is not supplied to the target CPU, or the power is not applied. After checking, restart the debugger.
01a3	A	Unconnected Emulation-board	The emulation board is not properly connected. Correctly connect the emulation board to the IE.
01a4	A	Contradictory Board-set	The board configuration in the IE has conflicts. Correct the board configuration and restart.
01a5	A	Unconnected I/O emulation-board	Emulation board 1 is not connected correctly. Correctly connect emulation board 1 to the IE.
01a8	A	Invalid EXPC.INI	The initialization file (EXPC.INI) cannot be properly read. The initialization file does not exist or may be corrupted. After reinstalling the initialization file, start.
02a0	F	Bus hold error	Bus hold. The user program cannot execute.

Error Message List (2/9)

Error Number	Type	Message	Meaning
0300	F	User program is running	The user program is running. This command cannot be executed.
0301	F	User program is stopped	The user program had a break. This command cannot be executed.
0302	F	User program is tracing	The tracer is running. This command cannot be executed.
0303	F	No tracing	There are no trace measurements.
0304	F	Now, trace memory is off	The tracer is off.
0305	F	Cannot over trace block	The trace block is exceeded and cannot move.
0306	F	There is no trace block	There is no trace block.
0307	F	There is no event-No	There is no event condition.
0308	F	Not doing Timer measurement	The timer measurement is not made.
0309	F	There is no trigger frame	There is no trigger frame.
030a	F	Traces off	The tracer stopped.
030e	F	Illegal memory range	The memory copy range overlapped.
030f	F	Already specified mode	Tracer is already in the on state.
0310	F	Illegal event number	The event condition is not set.
0313	F	Mapping range over.	The mapping setting is incorrect. A mapping that cannot be set is specified.
03a0	W	Target power off	The power to the target is off.
03a1	F	Now stepping	This command cannot be used while stepping.
03a2	F	Tracer is running	The tracer is running. This command cannot be used.
0400	F	Illegal parameter	The parameter is illegal.
0401	F	Result of Timer measurement is over	The timer measurement overflowed.
0402	F	Pass count conditions overflow	The event condition setting the pass count cannot be simultaneously used.
0403	F	Specified address range is over	Tried to set more than the maximum number of settings for the address range specification condition.
0404	F	Event conditions overflow	Set more than the number of event conditions that can be simultaneously used. There are a maximum of four bus event conditions and a maximum of four execution event conditions.
0407	F	Initialized data overflow	The number of initialized data exceeds the initialization range
0408	F	Search data number over	The search data becomes string data that exceeds 16 bytes. The maximum size of search data is 16 bytes.
0409	F	Search range over	The size of the search data exceeds the size of the search range.
04a0	F	Number of Trigger condition overflow	The number of software break settings exceeds 100.
04a1	F	Emulation memory is not enough	Tried to map the substitute memory to a region larger than 1 Mbyte.

Error Message List (3/9)

Error Number	Type	Message	Meaning
04a2	F	Bus size conditions overflow	The divisions of the bus size exceeded 8. Sometimes events cannot be properly set.
04a3	F	BRS event conditions overflow	More than 5 execution event conditions are set. (The maximum number of execution event conditions is 4.)
04a4	F	BRA event conditions overflow	More than 5 bus event conditions are set. (The maximum number of bus event conditions is 4.)
05a0	A	Evade runaway hardware	The IE is unstable. Reset the IE and forcibly break the user program.
0600	A	Communication buffer error	The region of the buffer for the communication data with the IE cannot be guaranteed. Exit other Windows applications, or change the setting the swap file used by Windows to increase the main memory of the host machine.
1000	A	Failure in initialization	The IE initialization failed. Make sure the IE is functioning properly.
1003	F	Illegal relocation address	Cannot locate to the specified address.
1004	F	Illegal parameter	The parameter is illegal.
1006	F	Illegal address	The address is illegal.
1007	A	Not enough substitute memory	Tried to map the substitute memory to a region larger than 1 Mbyte.
100b	F	Program Is Running	This command cannot be used while a user program is running.
100c	F	Different Bussize	A setting duplicated a region with a different bus size.
100d	F	Total Maximum Over	Tried to register above the maximum number (8) of bus size divisions.
100e	F	Enable Maximum Over	The divisions of the bus size exceeded 8.
100f	W	Wrong Target Status (Power Off)	The target state is unstable.
10ff	A	Communication Error	Cannot communicate with the IE. Check that the IE is functioning properly.
2000	F	Illegal sfr name	The SFR name is illegal.
2002	F	User program is running	The user program is running. This command cannot be executed.
2003	F	Illegal SFR number	Tried to access a nonexistent SFR.
2004	F	Illegal bit number	The bit SFR is not at the specified bit position.
2005	W	Redraw sfr name	A redraw-protected SFR was specified.
2006	F	This SFR is hidden SFR	The SFR is not usually open. The data cannot be displayed and changed.
2007	F	Can't Read/Write	Tried to write to a write-protected SFR. Or tried to read a read-protected SFR.
2008	F	Too big number	The specified SFR does not exist.
200a	F	Illegal Bit Pattern	Tried to set an illegal value in the SFR.

Error Message List (4/9)

Error Number	Type	Message	Meaning
20ff	A	Communication Error	Cannot communicate with the IE. Check that the IE is functioning properly.
3000	F	Illegal address	The address is illegal.
3001	F	Different data	The memory contents do not match.
3002	F	Illegal source address	The source address specification range exceeds the mapping range. (In a memory search, memory compare, memory copy)
3003	F	Illegal destination address	The destination address specification range exceeds the mapping range. (In a memory search, memory compare, memory copy)
3004	F	Illegal address (source & destination)	The address specification range exceeds the mapping range. (In a memory search, memory compare, memory copy)
3005	F	Illegal parameter	The parameter is illegal.
3006	F	User program is running	The user program is running. This command cannot be executed.
3008	F	No Parameter	There are no parameters.
3009	F	Parameter Size Alignment Error	The parameter size is illegal. Change the parameter to conform to the access size of the memory.
300a	F	Memory Alignment Error	The address is illegal. Change the address to conform to the access size of the memory.
300b	F	Source Start Address Alignment Error	The source address is illegal. Change the source address to conform to the access size of the memory
300c	F	Error, Destination Start Address Alignment Error	A memory range with a different access size was specified in the destination address range
300d	F	End Address Alignment Error	The end address is illegal. Change the end address to conform to the access size of the memory.
300e	F	Different Access Size in This Area	A memory range with a different access size was specified in the address range.
300f	F	Different Access Size in Source Area	A memory range with a different access size was specified in the source address range.
3010	F	Different Access Size in Destination Area	A memory range with a different access size was specified in the destination address range.
3011	F	Different Access Size, Source & Destination	The access sizes differ in the source address range and the destination address range.
30ff	A	Communication Error	Cannot communicate with the IE. Check that the IE is functioning properly.
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.

Error Message List (5/9)

Error Number	Type	Message	Meaning
4005	F	Illegal data	The data is illegal.
4006	F	Can't action number	Tried to use an event number that was already used.
4007	F	Can't empty number	Tried to register more than 32,767 events of the same type.
4008	F	Table not found	The specified event is not 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	Tried to register the same event more than 32,767 times.
400e	F	No entry event number	The specified event condition does not exist.
400f	F	Illegal Elink data	The event conditions setting the range condition and path condition were used in an event link condition. Or only one event condition is set.
4010	F	Function not found	The specified function is not found.
4011	A	No free memory	The memory is insufficient. Exit unused applications, or close the debugger window.
4013	W	Data access size mismatch at the bus size	The mapped bus size and the access size of the event condition differ.
4014	F	Can't use software break	The current software break cannot be used. Set a software break in the extended option setting dialog.
4015	F	Not point-address	In an address condition, the event condition setting the range cannot be used.
4016	F	Not renew event condition.	This event condition is used in another event. The address range condition and the pass count condition cannot be changed.
4017	F	Specified odd-address by word-access.	The data value cannot be detected in the word data that starts at an odd address. Delete the data specification and set.
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	The file seek failed.
5004	A	Cannot file close	The file close failed.
5005	A	Illegal device format	The format of the device file differs.
5006	A	Cannot device initialize	The IE initialization failed.
5007	A	Illegal device information	The device information does not exist.
5008	F	Cannot open device file	The specified device file 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	There is no function to move the internal RAM in the currently selected device.
6001	F	Illegal entry symbol name	The symbol name is illegal.

Error Message List (6/9)

Error Number	Type	Message	Meaning
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	The function display in the stack trace window is incomplete. One line has a maximum of 512 characters.
6005	F	Illegal expression	The expression is illegal.
7001	F	User program is running	The user program is running. This command cannot be executed.
7002	F	User program is stopped	The user program had 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 cannot be found. Stepping by the return command is not executed.
7010	W	Warning, No Source Line Information	Since there is no source information, instruction level stepping was executed.
7012	A	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
70fe	A	Bus Hold Error	There is a bus hold. The user program cannot be executed.
70ff	A	Communication Error	Cannot communicate with the IE. Check that the IE is functioning properly.
7801	F	Step wait canceled	The step execution was stopped. Since the Step execution is not finished, communication with the IE may no longer be possible.
7802	F	Step aborted	An illegal access break was generated during stepping. Check the user program.
7f00	F	Interrupted step	The step execution process was forcibly ended.
7f02	F	Suspended step	The stepping was suspended.
7f03	A	Run/Step cancel failed. CPU reseted	The user program break failed. Since the CPU was reset, the IE is unstable. Check that the IE is okay and restart.
7f04	F	Illegal address	Tried to execute from an unmapped region.
8000	F	File not found	This file is not found.
8001	F	Illegal line number	The line number is illegal.
8002	F	Current data is not set	The current data is not set.
8003	F	Illegal address	The address is illegal.
9002	F	Illegal set value	The specified value cannot be set in the register. Input 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 larger than the end address (start address > end address). Check the addresses.

Error Message List (7/9)

Error Number	Type	Message	Meaning
a003	F	Source path not found	The specified source path data is illegal. Set valid source path data.
a004	F	Expression is too big	The expression exceeded 127 characters.
a005	A	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
a006	F	Illegal argument	The argument is illegal.
a008	F	Source path not set	The source path is not set.
a009	F	File not found	The file is not found.
a00a	F	File not open	This file cannot be opened.
a00b	A	File not close	The file close failed.
a00c	A	File not read	The file read failed. The file is corrupted.
a00d	F	Not source file of LM	The specified source file is not registered in the load module file. A file not registered in the load module file cannot be displayed in the source display window.
a00e	F	Illegal line number	The line number is illegal.
a00f	F	Illegal variable	The variable does not exist.
a010	A	Communication failed	Cannot communicate with the IE. Check that the IE is functioning properly.
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 the mapping setting.
b000	F	Command line error	The parameter is illegal.
b001	F	Task type not found	The program data is not in the load module file.
b002	F	File not found	The file is not found.
b003	F	Function not found	The specified function is not found.
b004	F	Illegal magic number	The magic number of the load module file is illegal.
b005	F	Symbol not found	The symbol is not found.
b008	F	Illegal value	The expression is illegal.
b009	A	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
b00a	F	Illegal symbol entry	An illegal symbol is in the load module file. This may be a language related bug.
b00b	F	Current type nothing	There is no debugging information. Load the load module file.
b00c	F	Current file nothing	The current source file is not found. Since the load module file is not loaded, the source cannot be opened.
b012	F	Line number too large	The line number is illegal.
b015	A	Read error	The file read failed. The file may be corrupted.
b016	A	Open error	The file cannot be opened.
b017	A	Write error	The file cannot be written.

Error Message List (8/9)

Error Number	Type	Message	Meaning
b019	A	Seek error	The file seek failed.
b01a	A	Close error	The file close failed.
b01d	F	Address not found	The source line corresponding to the current PC does not exist.
b01e	F	No line information (not compile with -g)	There is no information in the source line in the load module file. Add the debugging option, and then recompile, assemble, and link.
b01f	F	Cannot find member	The member of the specified structure is not found.
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	An array with more than four dimensions cannot be displayed.
b027	F	End of file	The file is not at the end.
b029	F	Illegal address	The address is illegal.
b02a	A	Communication failed	Cannot communicate with the IE. Check that the IE is functioning properly.
b02b	F	No stack frame point	A stack trace is not possible for the current PC.
b02c	F	Max block overflow	The maximum number of blocks in one function is exceeded. The function cannot be displayed. (Maximum number of blocks per function: 256 blocks)
b02d	F	Illegal argument	The argument is illegal.
c001	F	Cannot open file	The file cannot be opened.
c002	A	Cannot close file	The file close failed.
c003	A	Cannot read file	The file read failed. The file may be corrupted.
c004	A	Cannot seek file	The file seek failed.
c005	F	Illegal file type	The file format is different. This file is not handled.
c006	F	Illegal magic number	The magic number of the load module file is illegal.
c007	F	This file is not load-module file	The specified file is not in the load module file.
c008	F	Old coff version	The version of the load module file is different.
c009	A	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
c00a	F	Illegal address	The address is illegal.
c00b	F	LM not load	The load module file is not loaded.
c00c	F	Illegal argument	Internal error
c00d	F	User program is emulating	The user program is running. This command cannot be executed.
c00e	F	User program is tracing	The tracer is operating. This command cannot be executed.




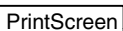








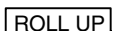

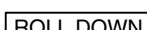
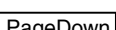














Error Message List (9/9)

Error Number	Type	Message	Meaning
c010	A	Communication failed	Cannot communicate with the IE. Check that the IE is functioning properly.
c011	F	Illegal file format	The file format in the load module file (LNK) is illegal.
c012	F	Check sum error	A checksum error occurred while reading the load module file. Check the load module file.
c013	F	Too large size	The address range to be uploaded exceeds 1 Mbyte.
c014	F	Cannot write file	Cannot write to the file.
c100	F	Not support	The Tektronix format is not supported.
d001	F	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
e000	F	Illegal argument	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 is illegal.
e004	F	Can't open file	The specified file cannot be opened.
e005	F	Can't read file	The file read failed. The file may be corrupted.
e006	F	Can't seek file	The file seek failed.
e007	F	Can't write file	The file write failed.
e008	F	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugging window.
e009	F	Illegal file format	The file format is illegal.






APPENDIX B LIST OF KEY FUNCTIONS

Special function keys can be used to effectively debug with ID operations. In the key descriptions, because the key expression differs with the type of keyboard in the IBM-PC/AT series, common key characters (generic key characters) are adopted.

B.1 Special Function Key Function List

Key		Function
PC-9801, 9821 Series	IBM-PC/AT Series	
		Deletes the character before the cursor. The cursor moves to the position of the deleted character. The string after the cursor is moved forward.
		The entire display screen is written to the clipboard as a bit image. (Windows function)
		(1) Closes the pull-down menu. (2) Closes the modal dialog.
		Moves the cursor to the menu bar.
		The last line is displayed. The cursor simultaneously moves to the last line.
		The first line is displayed. The cursor simultaneously moves to the first line.
		The screen scrolls up one screen. The cursor simultaneously moves to the top of the screen.
		The screen scrolls down one screen. The cursor simultaneously moves to the top of the screen.
		Inserts one space.
		The cursor moves to the next item.
		The cursor moves up. If the cursor is at the top of the screen, the screen scrolls down by one line each time.
		The cursor moves down. If the cursor is at the bottom of the screen, the screen scrolls up by one line each time.
		The cursor moves left. If the cursor is at the left of the screen, the screen scrolls to one item to the right.
		The cursor moves right. If the cursor is at the right of the screen, the screen scrolls to one item to the left.
		Confirms the input data.

B.2 Special Function Key Function List (**CTRL** + Key)

Key (Common to the PC-9801, 9821 Series and the IBM-PC/AT Series)	Function
A	The data value selected in the current window is the jump destination address. The disassemble is displayed from that address. The disassemble window opens.
B	Sets a breakpoint at the selected line.
C	Copies the selected string to the clipboard buffer.
F	The window switches to the modify mode. The operation is identical to the ToModify button.
G	Runs the program. The operation is identical to the  button.
H	Switches the window to the hold state.
I	Switches the window to the active state.
M	The data value selected in the current window is the jump destination address. The memory contents from that address are displayed. The memory window opens.
O	When the source text window is current The source view file is selected. The source file selection dialog is opened. Otherwise: The appropriate view file for the current window is displayed. The view file save dialog opens.:
P	Program execution pauses. The operation is identical to the  button.
R	Step executes until returning to the calling function. The operation is identical to the  button.
S	The displayed contents of the current window are saved in the view file.
T	Executes in steps. The operation is identical to the  button.
U	The data value selected in the current window is the jump destination address. The appropriate source text and source lines are displayed. The source text window opens.
V	The contents of the clipboard buffer are pasted at the text cursor position.
W	Switches the window to the view mode. The operation is identical to the ToView button.
X	Executes the next step. The operation is identical to the  button.
Z	The previous editing operation is undone.

APPENDIX C INDEX

[A]

Active State	43, 63
Address Specification	116
Addresses	19
ASCII display	126, 127
Auxiliary dialogs	49

[B]

Break event	95, 122
Break event conditions	161
break function	213
Break mode	74
Breakpoint set and delete functions	95
Breakpoint set/delete function	121

[C]

Character Set	16
Check box	41
Click	41
Confirmation dialogs	48
CPU clock source selection	69
CPU status	55
Current File	37

[D]

Debugger File List	24
Debugging Modes	37
Delay count	166
Dialogs	48
Disassemble	119
Display dialogs	49
Display window	46
Display/setting dialogs	49
Display/setting window	46
Double click	41
Drag & drop	41
Drive voltage display	69
Drop-down	43

[E]

Emulation CPU selection area	69
Emulation execution function	208
environment	24
Equipment connections	23
Error/Warning	196

Errors and Warnings	44, 227
event conditions	140
Event display	147
Event display function	95, 121
event link conditions	155
event manager	147
event setting and detection function	220
Execute window	45
Execution control	53
Exit	200
Exiting	33, 35
EXPC.INI	23
Expressions	21

[F]

file specification	17
Files	37
Find	98
font	94
Function specification	39

[G]

GUI function	15
--------------	----

[H]

Hold State	43, 63
------------	--------

[I]

icon	66, 97, 115, 176, 190, 124, 128, 139
Installation	23
instruction mode	37
Internal ROM/RAM display	69

[J]

Jump function	96, 122, 127, 138, 149, 182
---------------	-----------------------------

[L]

Line	37
Line number	95
Line specification	40
LIST OF KEY FUNCTIONS	237
List of Debugging Windows	50
load	76, 82
Loaded emulation/trace memory display	71
Location setting	70

lock selection function -----204

[M]

Management window -----47
 Mapping -----68
 Mapping setting -----70
 Mapping specification-----71
 Mark -----148
 Mask setting-----70
 memory compare -----133
 memory copy-----131
 Memory Initialization -----129
 Memory manipulation-----224
 Menu bar -----42, 53, 103, 147, 181
 Modal dialogs -----48
 Modeless dialogs -----48
 Modify Mode -----44
 Mouse-----41

[N]

Numerical Values -----18
 Non-real-time execution functions-----208

[O]

on-line assembly -----119
 Operands -----18, 21

[P]

Pass count-----144
 Pin mask -----70
 Point mark -----94, 120, 173
 Program counter -----96, 122
 Project File -----76, 79
 Pull-down menu -----42
 Push button and function button -----41

[R]

Radio button -----41
 Real-time execution -----208
 Real-time execution functions -----208
 Real-time internal RAM sampling-----74
 Real-time RAM Sampling-----37, 40
 register display -----181
 Register manipulation -----224
 Registers-----19
 run time measurement -----170, 217

[S]

Save -----85, 191
 save function -----224
 Scroll bar -----42
 Selection dialogs -----48
 Setting dialogs -----48
 SFR Window-----187
 Software break -----73, 74
 Source debugging-----225
 Source File-----90
 Source level debugging function -----15
 source mode-----37
 source path -----88
 source text displayed-----93
 Specification dialogs-----48
 Stack Frame Number-----37, 138
 Starting-----33
 Status bar -----43, 55
 Status display -----53
 step execution-----54, 61
 Structures -----37
 symbol-----20
 Symbol to Address-----101
 system operating mode-----203
 System operating states-----204

[T]

Terms-----22
 Time measurement function-----224
 Tool bar -----42, 53, 54
 Trace event conditions-----165
 trace function -----215
 Trace mode -----166, 215
 Trace View -----172

[V]

Variable display-----101, 104, 106
 Version-----199
 View Mode -----44
 Variable specification-----101, 104, 110

[W]

Watch function -----15
 Wild Cards -----17
 Window connect function-----96, 122, 127, 174
 Window display -----53
 Write mode-----75

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 Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

South America

NEC do Brasil S.A.
Fax: +55-11-889-1689

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	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>