

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 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 1<sup>st</sup>, 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**



# **ID78K0-LCE Integrated Debugger**

## **Preliminary**

---

Document No. U18152EU1V0UM00 (1st edition)  
©1999 - 2001 NEC Electronics Inc.  
All rights reserved. Printed in U.S.A.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics Inc. (NECEL). The information in this document is subject to change without notice. All devices sold by NECEL are covered by the provisions appearing in NECEL's Terms and Conditions of Sales only, including the limitation of liability, warranty, and patent provisions. NECEL makes no warranty, express, statutory, implied or by description, regarding information set forth herein or regarding the freedom of the described devices from patent infringement. NECEL assumes no responsibility for any errors that may appear in this document. NECEL makes no commitments to update or to keep current information contained in this document. The devices listed in this document are not suitable for use in applications such as, but not limited to, aircraft control systems, aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. "Standard" quality grade devices are recommended for computers, office equipment, communication equipment, test and measurement equipment, machine tools, industrial robots, audio and visual equipment, and other consumer products. For automotive and transportation equipment, traffic control systems, anti-disaster and anti-crime systems, it is recommended that the customer contact the responsible NECEL salesperson to determine the reliability requirements for any such application and any cost adder. NECEL does not recommend or approve use of any of its products in life support devices or systems or in any application where failure could result in injury or death. If customers wish to use NECEL devices in applications not intended by NECEL, customers must contact the responsible NECEL salespeople to determine NECEL's willingness to support a given application.

# Contents

<b>1. OVERVIEW</b> .....	<b>1</b>
1.1 Debugger .....	1
1.2 Functions .....	1
1.2.1 Operands.....	1
1.2.2 Source-Level Debugging Function .....	1
1.2.3 Instruction-Level Debugging Function .....	1
1.2.4 Low-Cost Emulator .....	1
1.2.5 Watch Function (Automatic Display Update Function When the Execution Pauses).....	1
1.2.6 Saving and Restoring the Debugging Environment.....	1
1.2.7 Displaying the Source Text in a Function .....	1
1.3 Input Conventions .....	2
1.3.1 Character Set .....	2
1.3.2 File Specification .....	3
1.3.3 Operands.....	4
<b>2. TERMINOLOGY</b> .....	<b>9</b>
2.1 Debugging Modes.....	9
2.2 Files .....	9
2.3 Current File .....	9
2.4 Functions .....	10
2.5 Current Function .....	10
2.6 Structures.....	10
2.7 Stack Frame Number .....	10
2.8 Line .....	10
2.9 Real-Time RAM Sampling.....	11
<b>3. WINDOW FUNCTIONS</b> .....	<b>13</b>
3.1 Basic Operations.....	13
3.1.1 Mouse.....	13
3.1.2 Push Button and Function Button.....	13

## CONTENTS

3.1.5	Scroll Bar .....	13
3.1.6	Menu Bar .....	14
3.1.7	Menu Command and Options.....	14
3.1.9	Status Bar.....	15
3.1.10	Drop-Down List.....	15
3.2	Active State and Static State.....	15
3.3	Errors and Warnings .....	15
3.3.1	Errors and Warnings During GUI Operation .....	15
3.3.2	Errors and Warnings Output by the Debugger .....	15
<b>4.</b>	<b>OPERATION.....</b>	<b>17</b>
4.1	Overview .....	17
4.1.1	Windows .....	17
4.1.2	Dialog Boxes .....	20
4.2	Functional Overview.....	21
4.3	Detailed Functional Descriptions.....	23
4.3.1	Main Window .....	23
4.3.2	Menus.....	26
4.3.3	Dialog Boxes .....	37
(1)	Configuration Dialog Box .....	37
(2)	Extended Option Dialog Box.....	40
(3)	Open Dialog Box.....	42
(4)	Save As Dialog Box.....	44
(5)	Download Dialog Box.....	46
(6)	Upload Dialog Box .....	48
(7)	Debugger Option Dialog Box .....	50
(8)	Open Dialog Box.....	53
(9)	Source Text Window.....	54
(10)	Source Search Dialog Box.....	57
(11)	Symbol to Address Dialog Box.....	58
(12)	Quick Watch Dialog Box .....	59
(13)	Watch Window .....	60
(14)	Add Watch Dialog Box.....	61
(15)	Local Variables in Watch Window.....	62
(17)	Assemble Window .....	63
(18)	Memory Window.....	66
(19)	Memory Fill Dialog Box.....	68
(20)	Memory Copy Dialog Box.....	69
(21)	Memory Compare Dialog Box.....	70
(22)	Memory Compare Result Dialog Box .....	71
(23)	Stack Trace Window.....	72
(25)	Event Manager .....	79

## CONTENTS

(26)	Event Link Dialog Box .....	85
(27)	Break Dialog Box .....	90
(28)	Trace Dialog Box.....	94
(29)	Trace View Window .....	97
(30)	Register Window.....	102
(31)	SFR Window.....	106
(32)	Open Dialog Box.....	108
(33)	Save As Dialog Box .....	110
(34)	Exit Debugger Dialog Box.....	115
(35)	Pass Count Dialog Box.....	116
(36)	Delay Count Dialog Box.....	117
(37)	Timer Dialog Box.....	117
(38)	Flash Programming Dialog Box.....	118
	Figure 4-70. Flash Programming Dialog Box .....	118
<b>5.</b>	<b>FUNCTIONAL OVERVIEW.....</b>	<b>121</b>
5.1	Operating Modes .....	121
5.1.1	Break Mode .....	121
5.1.2	Emulation Mode.....	121
5.1.3	Trace Mode .....	121
5.2	Basic Functions.....	122
5.2.1	Clock Selection Function.....	122
5.2.2	Mapping function .....	122
5.2.3	Stack area .....	122
5.2.4	Reset function.....	122
5.2.5	Load function.....	122
5.2.5	Emulation function.....	124
5.2.6	Break Function .....	128
5.2.7	Trace functions.....	130
5.2.8	Event setting and detection function.....	133
5.2.9	Register manipulation functions .....	135
5.2.10	Memory manipulation functions.....	136
5.2.11	Save function.....	136
5.2.12	Time measurement function .....	136

## CONTENTS

5.2.13	Source debugging .....	136
<b>Appendix A</b>	<b>Error Messages.....</b>	<b>137</b>
<b>Appendix B</b>	<b>Key Functions.....</b>	<b>147</b>
<b>Appendix C</b>	<b>INDEX.....</b>	<b>149</b>



---

# 1. OVERVIEW

## 1.1 Debugger

The LCE-K0 integrated debugger (known as the *ID* or the *debugger*) operates using a dedicated parallel board connected to an IBM® PC-compatible host running Microsoft® Windows®.

## 1.2 Functions

This section describes the functions and features of the ID.

### 1.2.1 Operands

Debugging takes place in the Windows environment using a mouse. Buttons and menus are arranged on each window. Related information is easily viewed from the display.

### 1.2.2 Source-Level Debugging Function

Referencing and setting variables and structures, displaying programs, and setting breakpoints is efficiently performed at the source text level of function names and line numbers.

### 1.2.3 Instruction-Level Debugging Function

Referencing and setting symbols and register values, displaying programs, and setting breakpoints is efficiently performed at the instruction level of labels and addresses.

### 1.2.4 Low-Cost Emulator

The detailed event setting functions of the low-cost emulator are used to set breaks and to trace programs.

### 1.2.5 Watch Function (Automatic Display Update Function When the Execution Pauses)

When the user program pauses, the values in the display window and display/setting window are automatically updated.

### 1.2.6 Saving and Restoring the Debugging Environment

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

### 1.2.7 Displaying the Source Text in a Function

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

## 1.3 Input Conventions

### 1.3.1 Character Set

This character set can be used in the integrated debugger.

**Table 1-1. Character Set**

<b>English Letters</b>	<b>Uppercase Letters</b>	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
	<b>Lowercase Letters</b>	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	
<b>Numbers</b>		0	1	2	3	4	5	6	7	8	9			
<b>Equivalent English Characters</b>		@	?	_										
<b>Special Characters</b>		.	,	:	;	*	/	+	-	'	<	>	(	)
		\$	=	!	#	[	]							

**Table 1-2. Character Descriptions**

<b>Character</b>	<b>Name</b>	<b>Main Use</b>
.	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
'	Apostrophe	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 line feed LF (0DH).

### 1.3.2 File Specification

A file is specified in the format ***primary-name[file-type]***

A directory is specified in the format ***[drive-name:][[directory-name]...]***

**Table 1-3. File/Directory Naming**

String	Description
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

**Figure 1-1. File Specification**

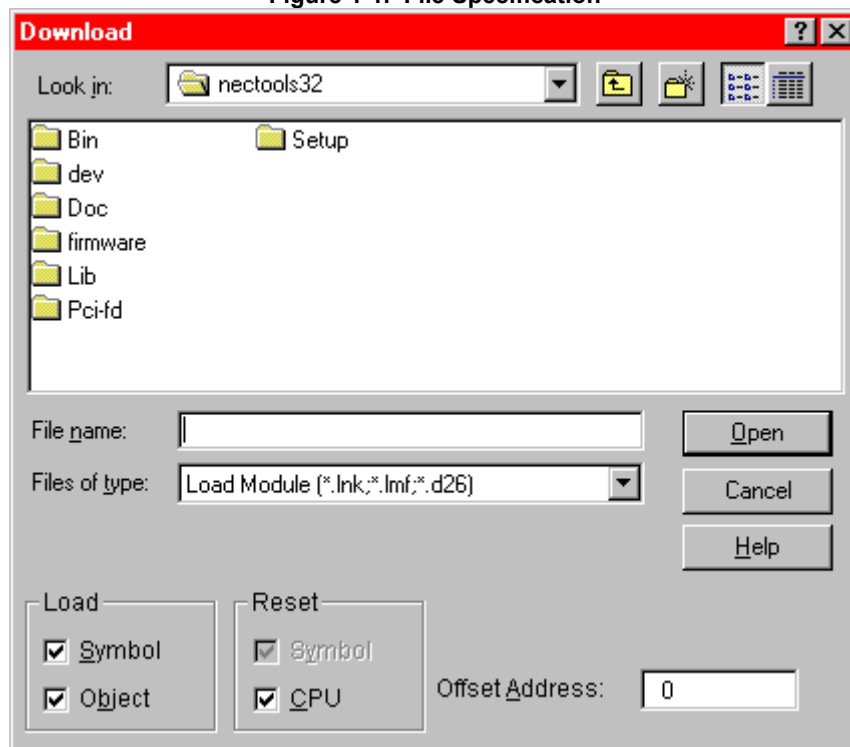


Table 1-4. Wild Cards

Character	Description
* and ?	Can be used in a path name or file name as wild cards
*	Denotes any character string
?	Denotes any one character (blank is also considered one character)

**Notes:**

1. If a wild card is specified, then the corresponding directory names under the directory and all of the file names are displayed.
2. If a file name is directly specified, an error occurs when a wild card is used. For example, if these eight files are saved in a directory, then the file names corresponding to the wild cards would be:

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

Table 1-5. File Name Corresponding to Wild Card

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

**1.3.3 Operands**

There are five types of operands:

- Numerical values
- Addresses
- Registers
- Symbols
- Expressions and operators

(1) Numerical Values

Table 1-6. Numerical Values

Number	Input Format	Example(s)		
Binary	NY <sup>(Note 1)</sup>	n...nY <sup>(Note 1)</sup> (n = 0, 1)		
Octal	NO <sup>(Note 1)</sup>	n...nO <sup>(Note 1)</sup> (n = 0, 1, 2, 3, 4, 5, 6, 7)		
Decimal	n	n...n	nT <sup>(Note 1)</sup>	n...nT <sup>(Note 1)</sup> (n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
Hexadecimal	nH <sup>(Notes 1, 2)</sup>	n...nH <sup>(Notes 1, 2)</sup>	0xn <sup>(Note 1)</sup>	0xn...n <sup>(Note 1)</sup> (n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F <sup>(Note 1)</sup> )

Notes:

1. The suffixes (Y, O, T, H, 0x) and the hexadecimal letters can be either uppercase or lowercase.
2. If the first character is A to F, a 0 must be added at the beginning (for example, FFH → 0FFH).

(2) Addresses

An address is specified by a numerical value, symbol, or expression. If an address is specified by a numerical value, a hexadecimal, decimal, octal, or binary number can be used.

(3) Registers

A general-purpose register is specified by an absolute name or function name. The PSW register has a name assigned to each bit.

Table 1-7. Registers

Type	Name
Control registers	PC
	SP
	PSW

Type	Name
PSW	Z
	AC
	IE
	CY

Type	Name	
	Absolute Name	Function Name
General-purpose registers	R0	X
	R1	A
	R2	C
	R3	B
	R4	E
	R5	D
	R6	L
	R7	H
	RP0	AX
	RP1	BC
	RP2	DE
	RP3	HL

#### (4) Symbols

A symbol is any of the characters A to Z, a to z, @, ?, \_ (underline), and 0 to 9. A symbol name must begin with a character other than 0 to 9 and be no longer than 31 characters (maximum). If a symbol longer than 31 characters is defined, then only the first 31 characters are valid.

- ❖ Uppercase letters A to Z are distinguished from lowercase letters a to z (case-sensitive).
- ❖ A symbol is defined by loading a load module file. It can be described instead of an address or numerical value, but its valid range is determined by the source debugging data when assembled or compiled.
- ❖ Each valid range has several types of symbols.
  - Public symbols (assembler, structured assembler, C) described by a symbol name
  - Local symbols described by a file name or module name
    - Local symbols in a module (assembler and structured assembler languages)
    - Local symbols in a file (C language)
    - Local symbols in a function (C language)
- ❖ For each language used, the following are available:
  - Assembler language and structured assembler language: label names, constant names, bit symbol names
  - C language
    - Variable names (including point variable names, enumeration variable names, array names, structure names, union names)
    - Function names and label names (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)
    - Array elements, structure elements, union elements, bit fields (when the symbol is an array, structure, or union)

**(5) Expressions and Operators****(a) 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, then the address is operated on as a symbol value. Elements other than operators forming an expression are called *terms* (constant, label) and are the first term, second term, and so on from the left in the description.

**(b) Operators**

The types of operators available are listed in Tables 1-8, 1-9, and 1-10.

**Table 1-8. 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; 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.

**Table 1-9. 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

**Table 1-10. Other Operators**

Symbol	Meaning	Description
(	Left parenthesis	An operation enclosed by parentheses () has priority over those outside the parentheses
)	Right parenthesis	

**Remarks:**

1. The left parenthesis and the right parenthesis are always used in pairs.
2. A character string can be described in the term in a comparison operation.
3. Operations are performed according to the following conventions:
  - 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 those outside of parentheses.
  - Each term in an operation is treated as unsigned 32-bit data.

- Operation results are handled as unsigned 32-bit data.
  - If an overflow occurs, the low-order 32 bits are valid and the overflow is not detected.
4. The operator precedence is as follows.

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

**(6) Terms**

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

- For binary numbers:  $0Y \leq \text{numerical value} \leq 11111111111111111111111111111111Y$  (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)
- For hexadecimal numbers:  $0H \leq \text{numerical value} \leq 0FFFFFFFH$



---

## 2. TERMINOLOGY

This chapter describes the terminology related to the 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**

### 2.1 Debugging Modes

Three debugging modes are available from the Main window.

- In source mode, a program is executed in one-line units of the source text.
- In instruction mode, step execution is performed at the instruction level.
- In auto mode, the mode is automatically detected based on the active window.

### 2.2 Files

The ID handles the following types of files:

- \*.C, \*.ASM, \*.S source files
- \*.LNK, \*.LMF load module files
- \*.HEX hexadecimal files
- \*.PRJ project files
- \*.\* display files

### 2.3 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.

#### File Specification Format

- |  |
|--|
| a. path name/file name<br>b. file name |
|--|

- In case a (when the path is specified), the file is read from or written to the directory given by the path.
- In case b (when no path is specified), the file is read from or written to the current directory.

## 2.4 Functions

These functions form a C source program.

#### Function Display and Specification Format

- |                           |
|---------------------------|
| a. file#_func<br>b. _func |
|---------------------------|

(file: file name; func: function name)

- In case a (when the file is specified), *func* is interpreted as a valid static function in the specified file.
- In case b (when no file is specified), search for the corresponding function name first among the valid static functions and then among 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

## 2.5 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.

## 2.6 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.

## 2.7 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.

## 2.8 Line

The line specifies a particular line in the source file. The line display and specification format is

file:line

(file: file name; line: line number)

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

### Line Specification Example

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

## 2.9 Real-Time RAM Sampling

Even while a user program is executed, if the variables are allocated to a space where the memory contents can be read or the memory displayed, the 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*, which is anything other than the unmapped area and SFR area.



## 3. WINDOW FUNCTIONS

### 3.1 Basic Operations

The window interface is used to perform debugging operations. In other words, after selecting the debugging target (variable, line, task, and so forth), you can select a corresponding debugging function using a function button.

Some menus are functionally equivalent to the function buttons, and debugging can also be performed using shortcut keys from the keyboard.

These objects are used to manipulate the ID78K0-LCE.

#### 3.1.1 Mouse



Operation of the integrated debugger uses the left mouse button, unless otherwise specified. There are three basic mouse operations.

Click	Press the mouse button once and release.
Double click	Consecutively press the mouse button twice and release.
Drag and drop	While continuing to press the button, move to drag, and then release the button to drop the element in place.

#### 3.1.2 Push Button and Function Button

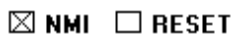


A push button is a thick, rectangular button used to display a bit map or character string. Click the rectangular shape to start the corresponding process.



A function button starts a debugging function.

#### 3.1.3 Check Box



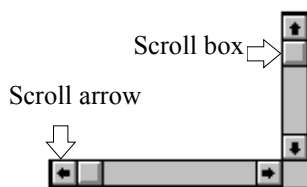
A check box is used to turn an option on or off. Click the box to clear  or select  the option. Multiple selections are possible.

#### 3.1.4 Option Button



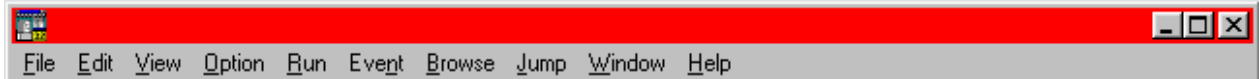
An option button indicates choices in a dialog box. Click the button to clear  or select  the option. If two or more option buttons are grouped together, only one can be selected.

#### 3.1.5 Scroll Bar



The vertical and horizontal scroll bars are used to move through the contents of the display vertically and horizontally, respectively. The scroll box shows the current location of the display in proportion to the entire contents. Clicking the vertical scroll arrow moves the display one line in the vertical direction. Clicking the horizontal scroll arrow moves the display one line in the horizontal direction. Dragging and dropping the scroll box moves the display to that relative position.

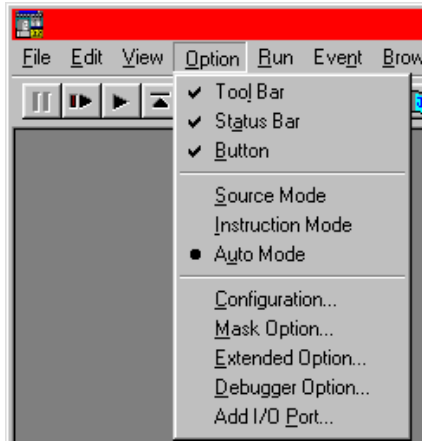
### 3.1.6 Menu Bar



The menu bar appears at the top of each window. Click a menu name to display the corresponding commands.

Alternatively, you can press the **Alt** key and the underlined letter in each name to perform the same operation.

### 3.1.7 Menu Command and Options



Each menu contains a series of commands and/or options. Click a name to invoke the corresponding command. The same command can be invoked using shortcut keys:

**CTRL** + **letter** (where *CTRL* refers to the CONTROL

key and *letter* refers to the letter underlined in each command and/or option). An operation that can be invoked using shortcut keys has “CTRL + letter” displayed to its right. Menu operations are described by the following terminology.

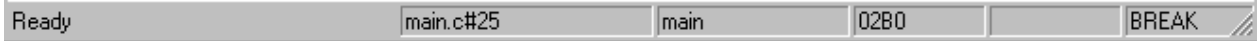
1. **“Item”**  
Indicates a command and/or option that initiates when the item is selected.
2. **“Item...”**  
Indicates a command and/or option that displays a dialog box requiring a response from the user.
3. **“Item ▶”**  
Indicates a command and/or option that displays a submenu with additional choices.

### 3.1.8 Toolbar



The toolbar contains buttons that invoke frequently used commands. Each button is identified by a symbol depicting the operation. Click the button to initiate the operation.

### 3.1.9 Status Bar



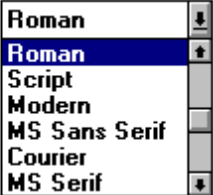
The status bar at the bottom of the window shows several pieces of information about the debugger and emulator:

- Source file name and line number indicated by the program counter (PC)
- Function name indicated by the PC
- PC value
- CPU ( $\mu$ PD789xxx) status
- Status of the low-cost emulator
- Break cause

### 3.1.10 Drop-Down List

Font:  : Before clicking the **arrow**

A drop-down list is a closed version of a list box with an arrow next to it. Clicking the arrow opens the list. Highlight and click an item in the list to select it.

Font:  : After clicking the **arrow**

### 3.2 c State

In the active state, execution of a user program or command automatically updates values displayed in the window. The static state maintains the values regardless of user program or command execution. When displayed contents sometimes change because of user program execution, the display window or display/setting window can switch to the active or static state. An active window can only display one type of window, but a static window can simultaneously display multiple windows of the same type. The following procedures explain how to change a window from active to static state and vice versa.

*Active State* → *Static State*

1. Click **Window** → **Static**.

*Static State* → *Active State*

1. Click **Window** → **Active**.

### 3.3 Errors and Warnings

The LCE-K0 handles errors and warnings differently. Errors are generated by the debugger.

#### 3.3.1 Errors and Warnings During GUI Operation

An error in GUI operation is regarded as a warning. If a warning occurs, the warning tone sounds or the error/warning dialog box appears.

#### 3.3.2 Errors and Warnings Output by the Debugger

If an error occurs, the Error/Warning dialog box appears.





## 4. OPERATION

### 4.1 Overview

The LCE-K0 is composed of windows and dialog boxes. A dialog box contains command buttons and options that enable you to invoke a command or specify settings. Windows can be minimized; dialog boxes cannot.

#### 4.1.1 Windows

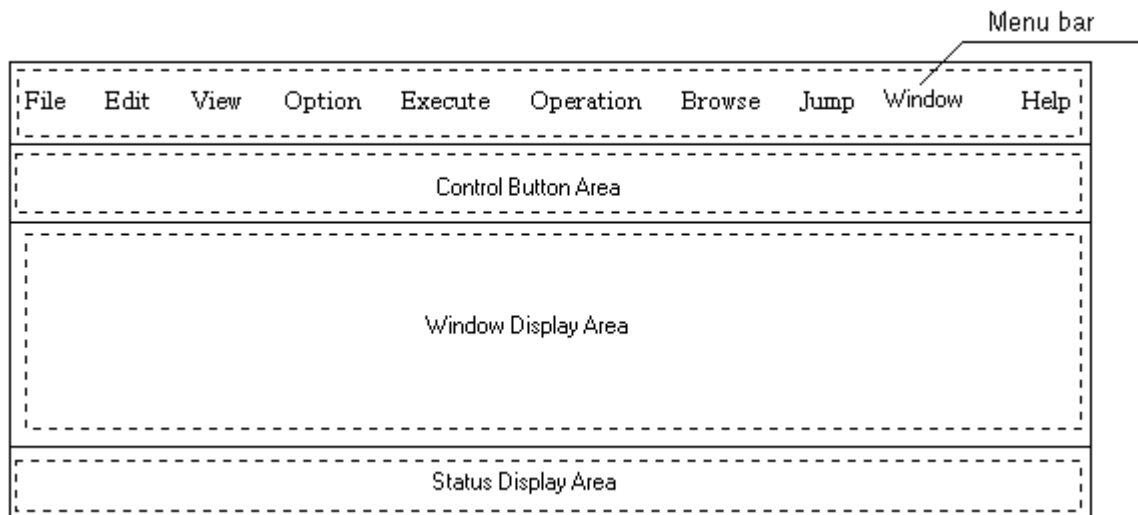
Windows are broadly classified by function into the following types.

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

##### (1) Execute-Type Window

The Main window is an execute-type window used to control the other windows and program execution. It consists of a menu bar, control buttons, window display area, and status display area (Figure 4-1).

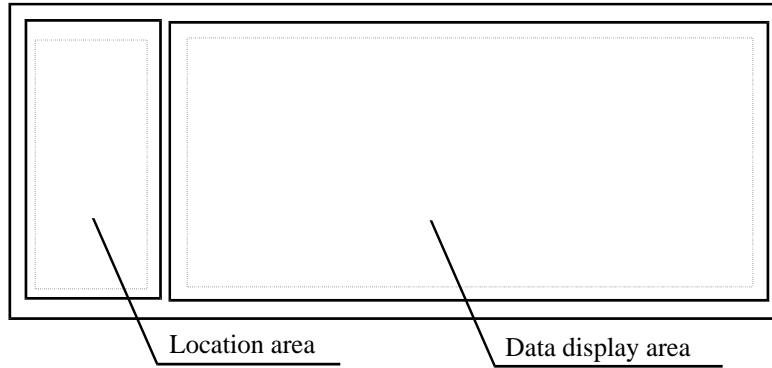
Figure 4-1. Execute-Type Window



##### (2) Display-Type Window

A display-type window has a location area and data display area where the contents of the target are displayed and values cannot be changed (Figure 4-2). The Source Text, Stack Trace, and Trace View windows are display-type windows.

Figure 4-2. Display-Type Window



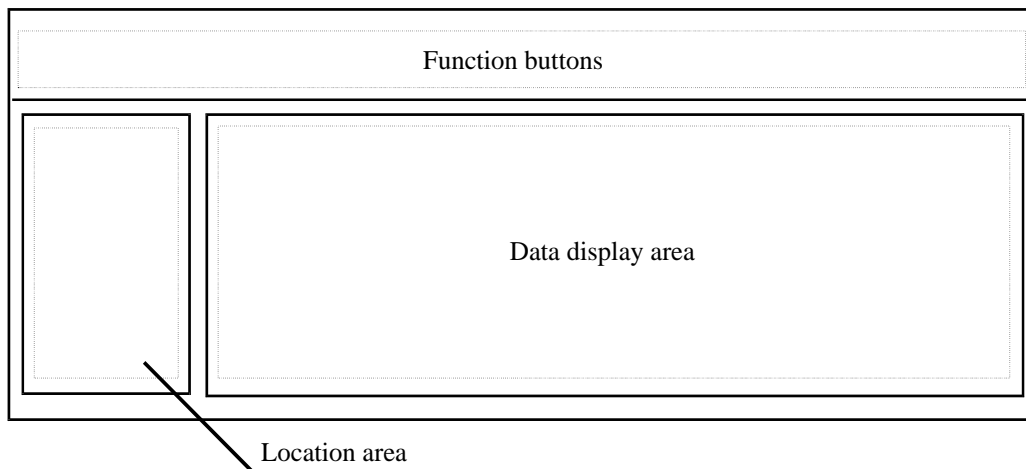
### (3) Display/Setting-Type Window

A display/setting-type 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 in Modify mode. There are two types of display/setting-type windows: those opened in the Main window and those opened outside the Main window.

#### (a) Opened in the Main Window

This type of display/setting window consists of function buttons, a location area, and a data display area. The Local Variable, Memory, SFR, and Disassemble windows are display/setting-type windows opened from the Main window.

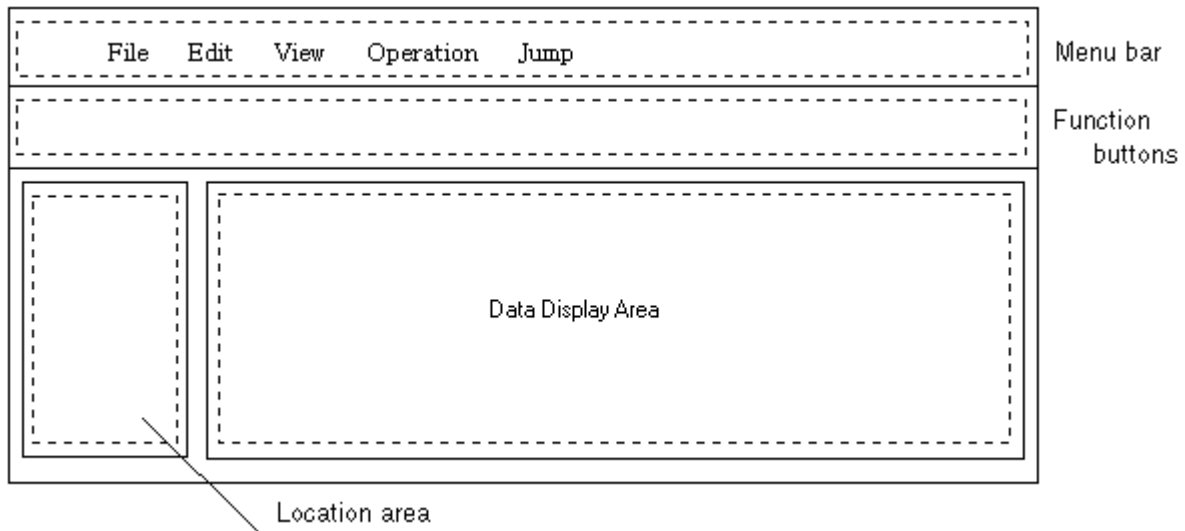
**Figure 4-3. Display/Setting-Type Window Opened in Main Window**



## (b) Opened Outside the Main Window

This type of display/setting window can be located anywhere outside the Main window, but it is always displayed in front of the Main window and cannot be minimized. The window has a menu bar, function buttons, a location area, and a data display area. The Register and Variable windows are both display/setting-type windows opened outside the Main window.

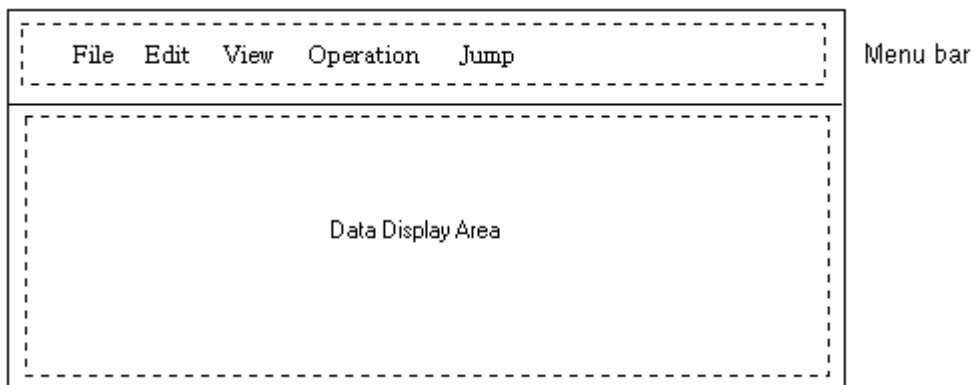
**Figure 4-4. Display/Setting-Type Window Opened Outside Main Window**



## (4) Management-Type Window

A management-type window manages the debugging settings and has a menu bar and data display area. The Event Manager window is a management-type window.

**Figure 4-5. Management-Type Window**




### 4.1.2 Dialog Boxes

Dialog boxes are classified into two types.

- Modal dialog boxes
- Modeless dialog boxes

#### (1) Modal Dialog Boxes

From a modal dialog box, you cannot access other windows or dialog boxes. To do so, you must first close the dialog box, either by waiting for the operation to finish or by clicking the  button to cancel it.

#### (2) Modeless Dialog Boxes

From the modeless dialog box, you can access other windows or dialog boxes, even if the current operation is still in progress.

There are seven types of dialog boxes:

- Selection-type dialog box
- Specification-type dialog box
- Setting-type dialog box
- Confirmation-type dialog box
- Auxiliary-type dialog box
- Display-type dialog box
- Display/setting-type dialog box

##### (a) Selection-type dialog box

A selection dialog box allows you to select conditions. The Configuration, Project File Load, Upload, View File Save, Load Module Selection, Project File Save, View File Load, and Source File Selection dialog boxes are selection-type dialog boxes.

##### (b) Specification-type dialog box

A specification-type dialog box allows you to specify the conditions, usually in a text area. The Address Specification, Source Path Specification, and Trace Window dialog boxes are specification-type dialog boxes.

##### (c) Setting-type dialog box

A setting-type dialog box allows you to set conditions. The Extended Option Setting, Event Link, Trace, Event Set, and Break dialog boxes are setting-type dialog boxes.

##### (d) Confirmation-type dialog box

A confirmation-type dialog box prompts for confirmation of a selected action. The Reset Debugger, Error/Warning, and Exit Debugger dialog boxes are confirmation-type dialog boxes.

##### (e) Auxiliary-type dialog box

An auxiliary-type dialog box is used for the auxiliary operations in each window. The Variable View, Memory Copy, Memory Compare, Add Variable, Memory Fill, and Search dialog boxes are auxiliary-type dialog boxes.

**(f) Display-type dialog box**

A display-type dialog box temporarily displays data. The Memory Comparison Result and About dialog boxes are display-type dialog boxes.

**(g) Display/setting-type dialog box**

A display/setting-type box has an area for setting conditions and displaying data. The Timer dialog box is a display/setting-type dialog box.

**4.2 Functional Overview**

The table below lists and briefly describes the various windows and dialog boxes.

**Table 4-1. List of Windows and Dialog Boxes (1/2)**

Name	Description
Main window	The first window displayed after the debugger starts
Configuration dialog box	Sets the debugger operation environment
Extended Option Setting dialog box	Sets various extended options
Project File Load dialog box	Reads in the debugging environment
Project File Save dialog box	Saves the debugging environment
Load Module selection dialog box	Reads in an object file or a symbol file
Upload dialog box	Uploads the memory contents to a file
Source Path Specification dialog box	Specifies the source path
Source File Selection dialog box	Selects the source file displayed in the Source Text window
Source Text window	Displays the source text
Search dialog box	Searches for a character string in the current window
Symbol to Address dialog box	Displays the address allocated to the symbol
Variable View dialog box	Temporarily displays the variable values
Variable window	Displays and changes variables
Add Variable dialog box	Adds the displayed variables to the Variable window
Local Variable window	Displays and changes local variables in the current function
Address Specification dialog box	Specifies the display starting address
Disassemble window	Displays disassembly of the program and assembles on-line
Memory window	Displays and changes the memory contents
Memory Fill dialog box	Initializes the memory
Memory Copy dialog box	Copies the memory
Memory Compare dialog box	Compares the memory
Memory Comparison Result dialog box	Displays the result of the memory comparison
Stack Trace window	Displays the contents of the function's stack
Event Set dialog box	Registers the event conditions
Event manager	Manages each registered event condition

Table 4-1. List of Windows and Dialog Boxes (2/2)

Name	Description
Event Link dialog box	Registers the event link conditions
Break dialog box	Registers and sets the break event conditions
Trace dialog box	Registers and sets the trace event conditions
Timer dialog box	Displays the result of run time measurements
Trace View window	Displays the trace result
Trace Window dialog box	Sets the trace display conditions
Register window	Displays and changes the registers
SFR window	Displays and changes the SFR
View File Load dialog box	Opens the window for referencing the current window
View File Save dialog box	Saves the display contents of the current window in a file
Error/Warning dialog box	Displays errors and warnings
Reset Debugger dialog box	Resets the debugger and target CPU
About dialog box	Displays the debugger version
Exit Debugger dialog box	Exits the debugger
Mask Option setting dialog box	Sets the mask option
Pass Count setting dialog box	Sets the pass count
Delay Count setting dialog box	Sets the delay count
Flash Programming dialog box	Displays the flash programmer graphical user interface

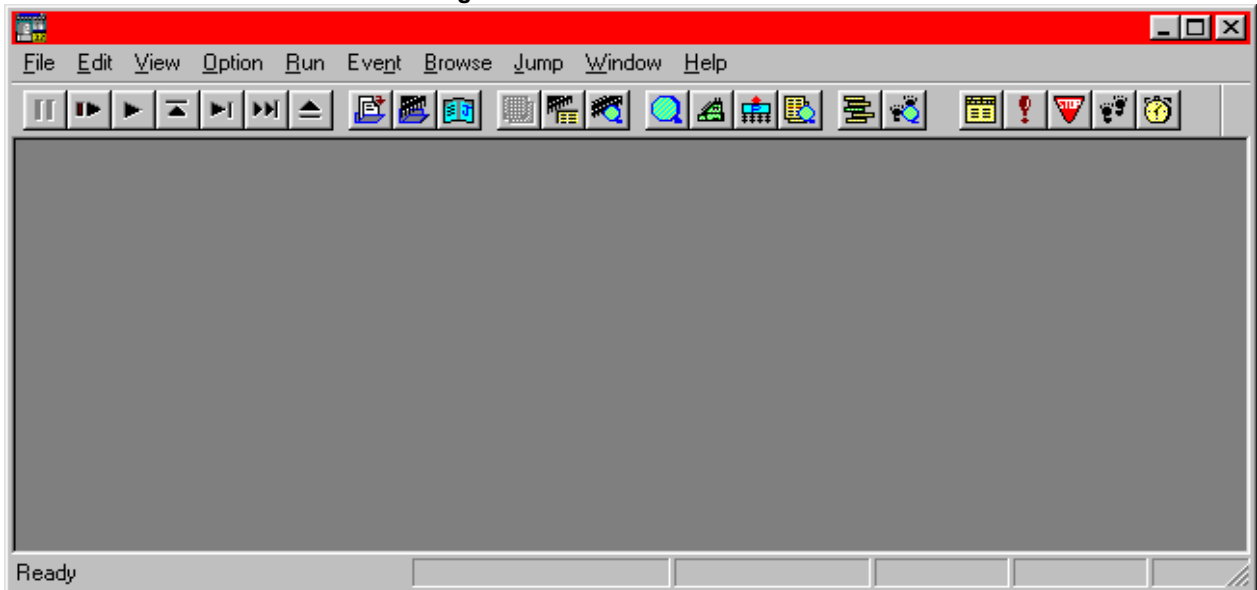
### 4.3 Detailed Functional Descriptions

This section provides an in-depth description of each window or dialog box.

#### 4.3.1 Main Window

The Main window automatically opens after completion of the initialization sequence and remains open until the program is exited. The Main window has three modes: *source mode* controls debugging at the source level; *instruction mode* controls debugging at the instruction level; and *auto mode* automatically detects the mode based on the active window (default mode after initialization).

Figure 4-6. Main Window



The Main window is composed of a toolbar, window display area, and status display area.

#### (1) Toolbar

The toolbar has buttons for popular commands to be executed in one action. Each button has an easy-to-understand graphical image that makes it readily identifiable. Commands can be invoked from the tool bar or the menu bar. To hide the toolbar, click **Option → Tool Bar → Hide**.



Stops or pauses execution of the user program



Performs the Reset and Go commands sequentially



Executes the user program; remains depressed until after completion of program execution



Executes a function in real time before returning to the CALL function



Executes a program in single steps. Pressing the button once executes one instruction. Successively pressing the button executes the corresponding number of instructions. Debugging at the source level is in line units. Debugging at the instruction level is in instruction units.



Executes the next step. Steps over a CALL instruction by executing the called function in real time. If the debugging takes place at the source level, step execution is in line units. If debugging takes place at the instruction level, step execution is in instruction units.



Initializes the debugger, emulation CPU, and symbol data; opens the Reset Confirmation dialog box



Opens the source code to be displayed in the Source window



Downloads the load module file (LMF) to the debugger



Opens a previously saved project file



Displays the source text



Displays the assembly results



Displays the memory contents



Displays the selected variable



Displays the CPU registers and general-purpose registers and their contents



Displays the SFRs and their contents



Displays the local variables and values



Displays the stack contents



Displays the Trace windows



Opens the Event Manager



Opens the Event dialog box





Opens the Break dialog box



Opens the Trace dialog box



Displays the Timer dialog box



Launches the flash programmer

## (2) Window Display Area

This area displays the following program windows, all of which can be resized or minimized.

- Source Text window
- Disassemble window
- Local Variable window
- Trace View window
- Memory window
- SFR window
- Stack Trace window

## (3) Status Display Area

		Function name	CPU Status	
Ready	main.c#25	main	02B0	BREAK
Source file name: line number			PC value	Status

**Table 4-2. Program and LCE Status**

Field	Description
Source file name	Displays the source file name and the source line number referenced by the program counter (PC) value. If there is no file data, "---" is displayed.
Function name	Displays the function name referenced 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 modes, and so forth)
Status	Displays the low-cost emulator state (RUN, BREAK modes, and so forth)

## 4.3.2 Menus

## (1) File Menu

Figure 4-7. File Commands

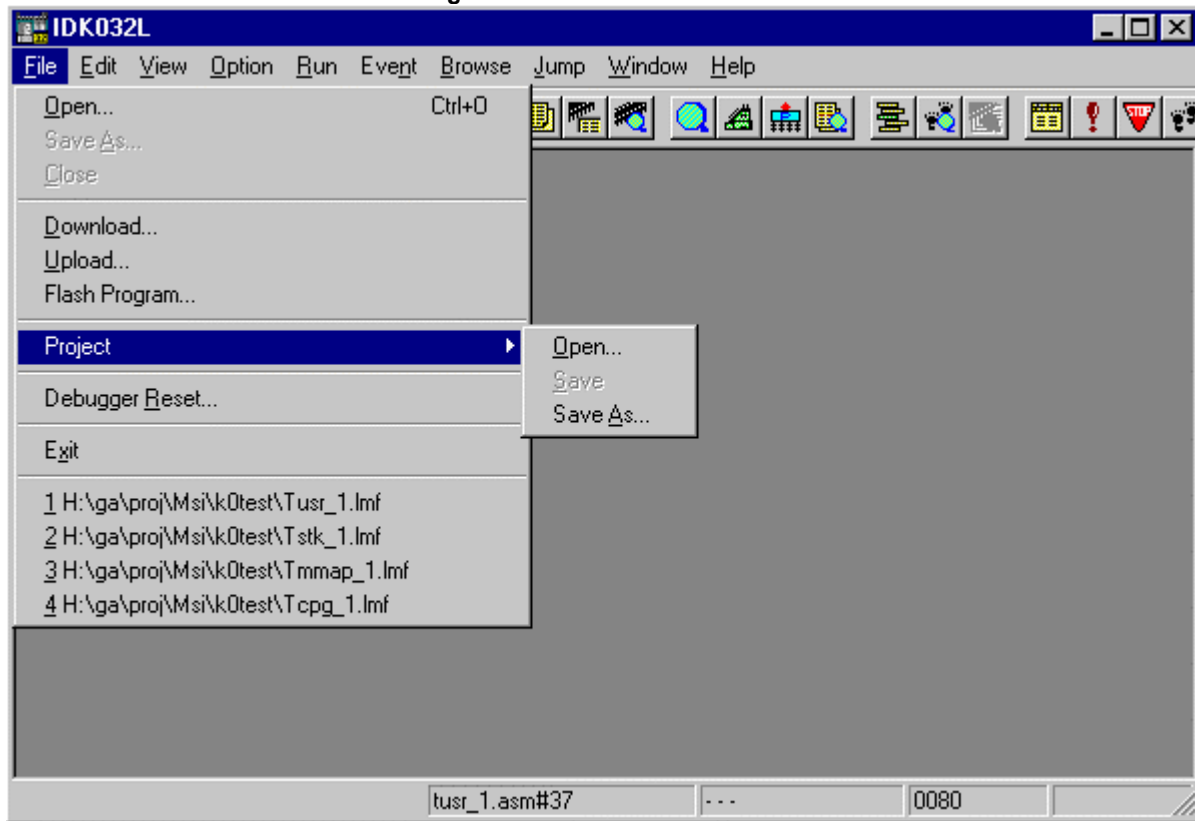


Table 4-3. File Command Descriptions

Command	Description
Open ▶	If the Source Text window is active, opens the Source File Selection dialog box where you can select a source file and click OK to open and view it. Otherwise, opens the View File Save dialog box where you can click OK to view the current file.
Close	Closes the current active window
Save As...	Opens the File Save dialog box and allows you to save the current window in a new file
Download...	Opens the Load Module Selection dialog box and allows you to select and download a LMF
Upload	Uploads the current load module file to a hex file
Flash Program...	Launches the flash programmer graphical user interface
Project	
Open...	Opens the Project File Load dialog box where you can select and open a project file
Save	Writes the current version to the project file opened previously
Save As...	Opens the Project File Save dialog box where you can click OK to save the current version
Debugger Reset...	Opens the Reset Debugger dialog box where you can click OK to reset the debugger
Exit	Opens the Exit Debugger dialog box where you can click OK to exit the debugger

## (2) Edit Menu

Figure 4-8. Edit Commands

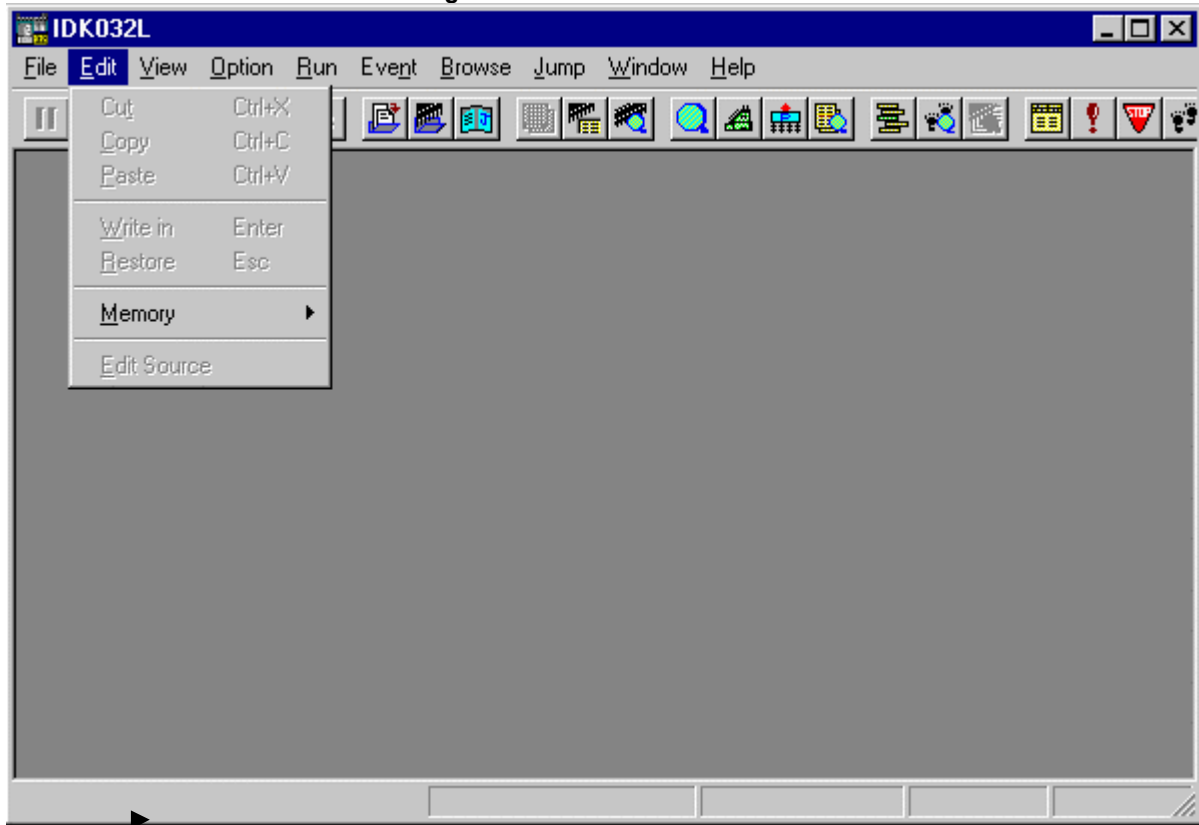


Table 4-4. Edit Command Descriptions

Command	Description						
Cut	Cuts a selected string						
Copy	Copies the selected string to the clipboard						
Paste	Pastes the contents of the clipboard buffer to the position of the text cursor						
Write in	Latches the changes made in the window						
Restore	Cancels the changes to the window						
Memory	<table border="1"> <tr> <td>Fill...</td> <td>Opens the Memory Fill dialog box where you can Initialize the memory</td> </tr> <tr> <td>Copy...</td> <td>Opens the Memory Copy dialog box where you can copy the memory</td> </tr> <tr> <td>Compare...</td> <td>Opens the Memory Compare dialog box where you can compare the memory</td> </tr> </table>	Fill...	Opens the Memory Fill dialog box where you can Initialize the memory	Copy...	Opens the Memory Copy dialog box where you can copy the memory	Compare...	Opens the Memory Compare dialog box where you can compare the memory
Fill...	Opens the Memory Fill dialog box where you can Initialize the memory						
Copy...	Opens the Memory Copy dialog box where you can copy the memory						
Compare...	Opens the Memory Compare dialog box where you can compare the memory						
Edit Source	Allows you to edit source code in the Source window, provided the debugger is invoked from the Project Manager, available with the RA78K0 package						

## (3) View Menu

Figure 4-9. View Commands

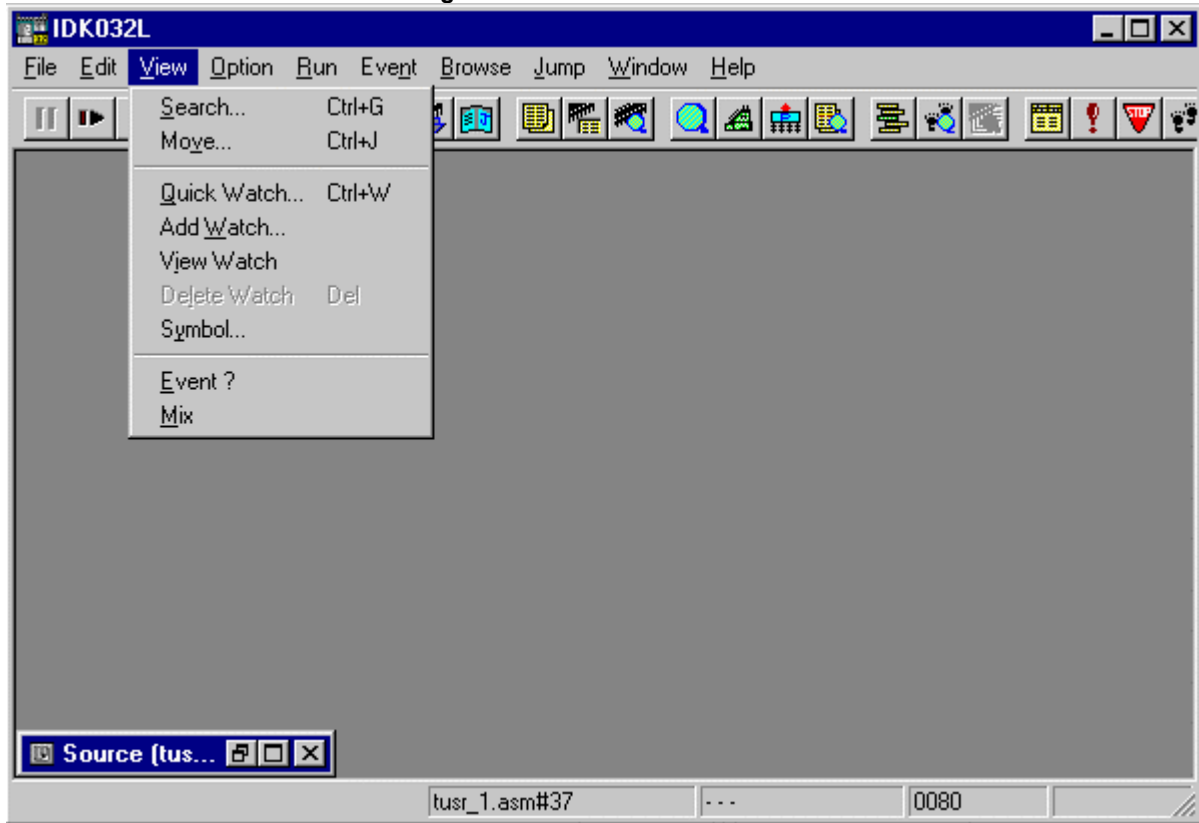


Table 4-5. View Command Descriptions

Command	Description
Search...	Opens the Search dialog box where you can search for strings or numerical values; same as <input type="button" value="Search..."/> button.
Move...	Moves cursor to specified address in the Memory or Assembly window or line number in Source Text window
Quick Watch...	Views the highlighted variable from the source text in the Quick Watch window
Add Watch...	Opens the Add Variable window where you can add a variable from the Source window to the Watch window
View Watch	Displays the Watch window
Delete Watch	Deletes the variable highlighted in the Watch window
Symbol...	Displays the absolute address of a specific symbol
Event?	Opens the Event Manager and displays event data
Mix	Simultaneously displays C language and assembly language code in the Source Text window

## (4) Option Menu

Figure 4-10. Option Commands

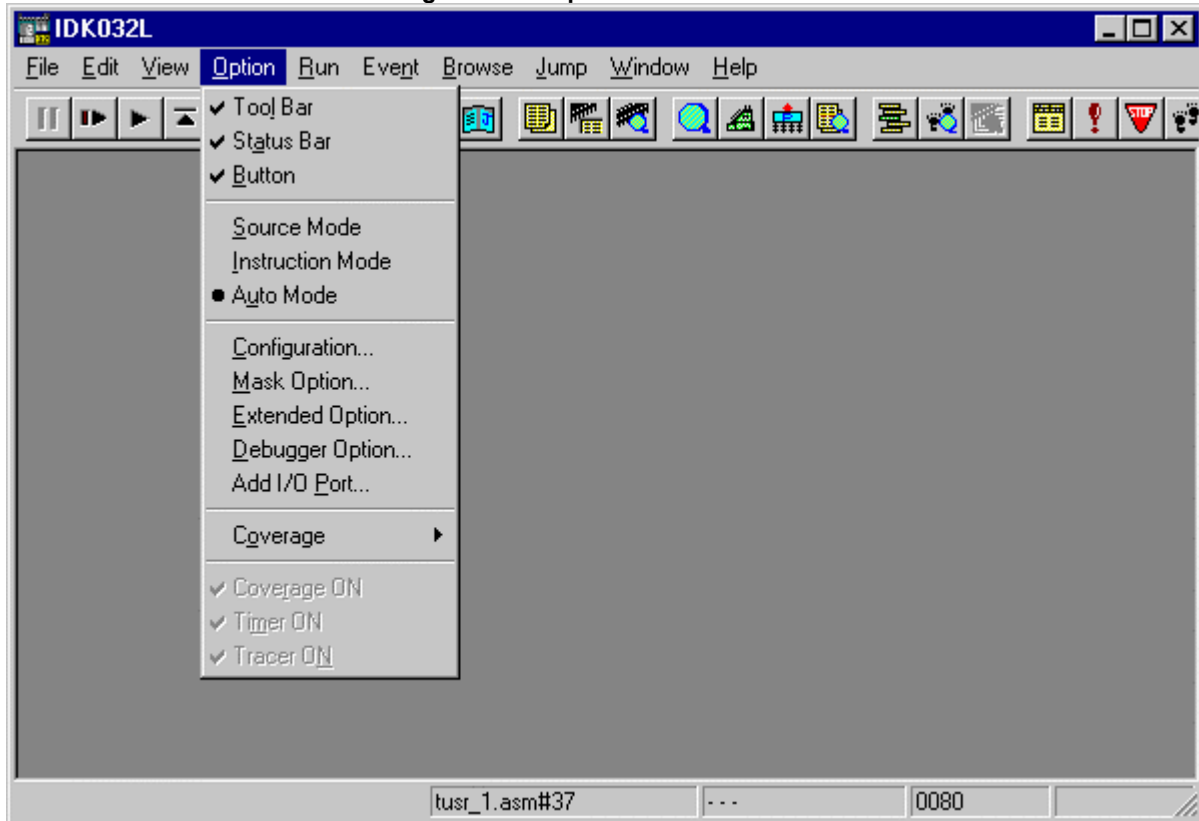


Table 4-6. Option Command Descriptions

Command	Description
Tool Bar	Selects to display or hide the toolbar
Status Bar	Selects to display or hide the status bar
Button	Selects to display or hide the buttons in each window
Source Mode	Step executes at the source level
Instruction Mode	Step executes at the instruction level
Auto Mode	Automatically selects step execution mode
Configuration...	Opens the Configuration dialog box, where you can set the environment
Mask Option...	Opens the Mask Option Setting dialog box
Extended Option...	Opens the Option Setting dialog box, where you can set the extended functions
Debugger Option...	Opens the Debugger Option window, where you can set parameters for the debugger
Add I/O Port...	Defines additional I/O ports
Coverage	Not supported

## (5) Run Menu

Figure 4-11. Run Commands

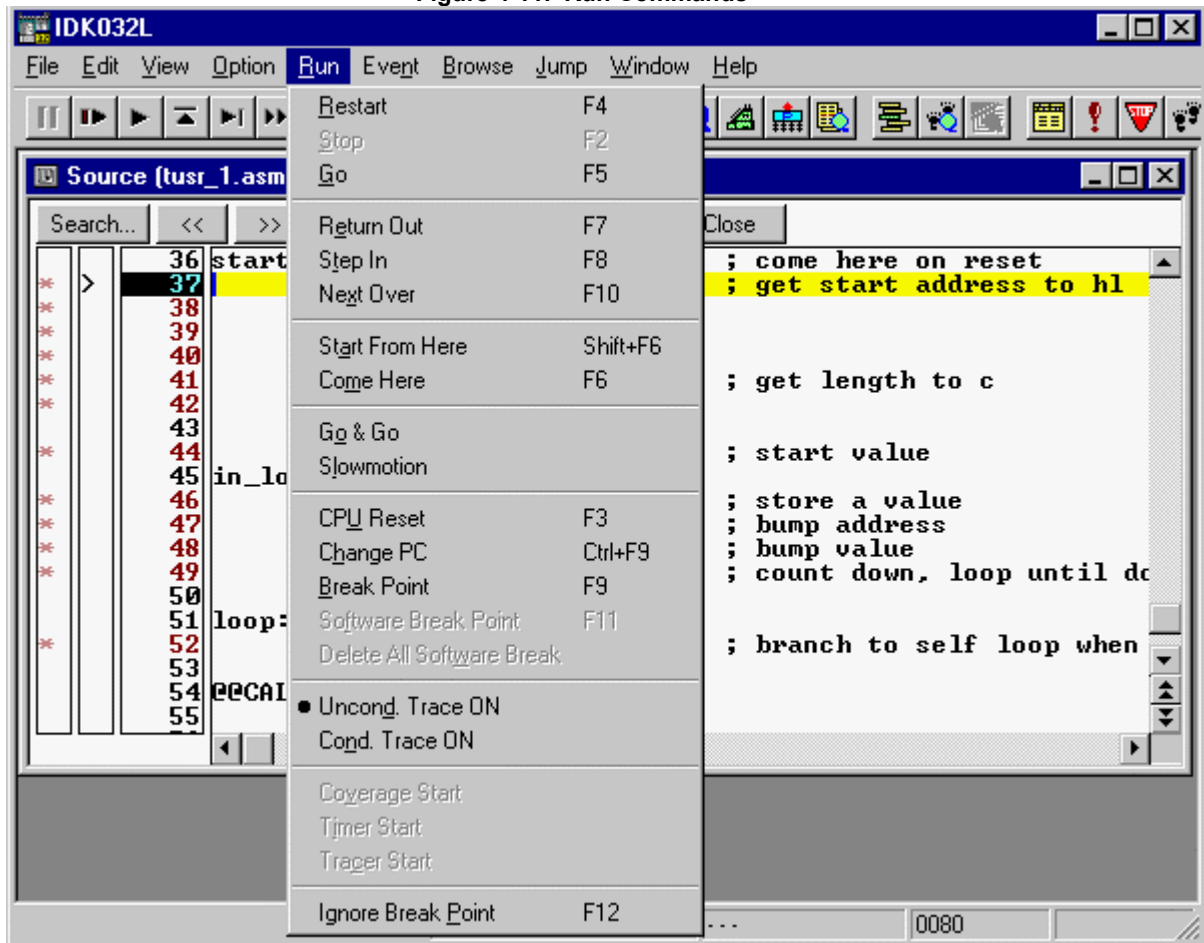










Table 4-7. Run Command Descriptions

Command	Description
Restart	Same as the  button: restarts the debugger (Stop, Reset, Go)
Stop	Same as the  button: stops program execution
Go	Same as the  button: executes the program
Return Out	Same as the  button: executes the program in real time until returning to the CALL function
Step In	Same as the  button: executes the program in steps
Next Over	Same as the  button: executes the next step
Start From Here	Sets the PC on the highlighted source line number or address
Come Here	Executes the program in real time until the PC reaches the highlighted source line number or address
Go & Go	Same as clicking the  button when a break occurs: continues program execution or resumes program execution if a break is generated by a break condition after the window is updated
Slowmotion	Continues step execution
CPU Reset	Same as the  button: opens the Reset Debugger dialog box, where you can reset the target system or the entire debugger
Change PC	Sets the PC register to the address at the selected line

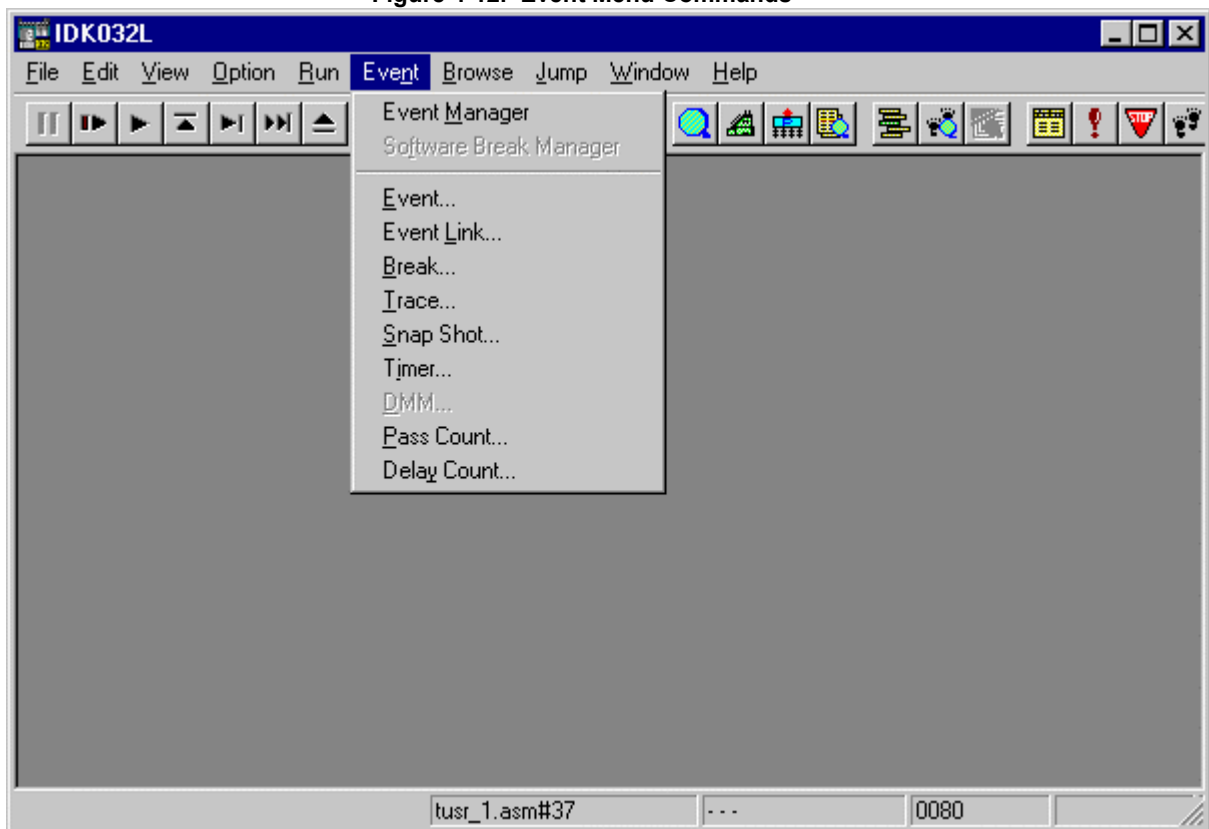
Break Point	Sets the breakpoint in the selected line
Software Break Point	Sets software breakpoint on the selected line

**Table 4-7. Run Command Descriptions (continued)**

Command	Description
Delete All Software Break	Deletes all defined software breakpoints
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
Timer Start	Starts the timer function
Tracer Start	Starts the trace function
Ignore Break Point	Disables active breakpoints

## (6) Event Menu

**Figure 4-12. Event Menu Commands**



**Table 4-8. Event Command Descriptions**

Command	Description
Event Manager	Opens the Event Manager window
Software Break Manager	Opens the Software Break Manager window (Not supported)
Event...	Opens the Event Set window, where you can define events in the Event Manager
Event Link...	Creates complex sequential events

#### 4. OPERATION

---

Break...	Opens the Break dialog box, where you can set breakpoints
Trace..	Opens the Trace Set dialog box, where you can set conditions for the tracer
Snap Shot...	Opens the Snap Shot window to capture contents of registers, memory fields, SFRs ( <b>not supported</b> )
Timer...	Opens the Timer Setting window to set conditions for the timer
DMM...	Not supported
Pass Count...	Sets the number of occurrences of an event condition before the event is triggered
Delay Count...	Performs the number of traces specified by the delay count after the stop condition is satisfied



## (7) Browse Menu

Figure 4-13. Browse Commands

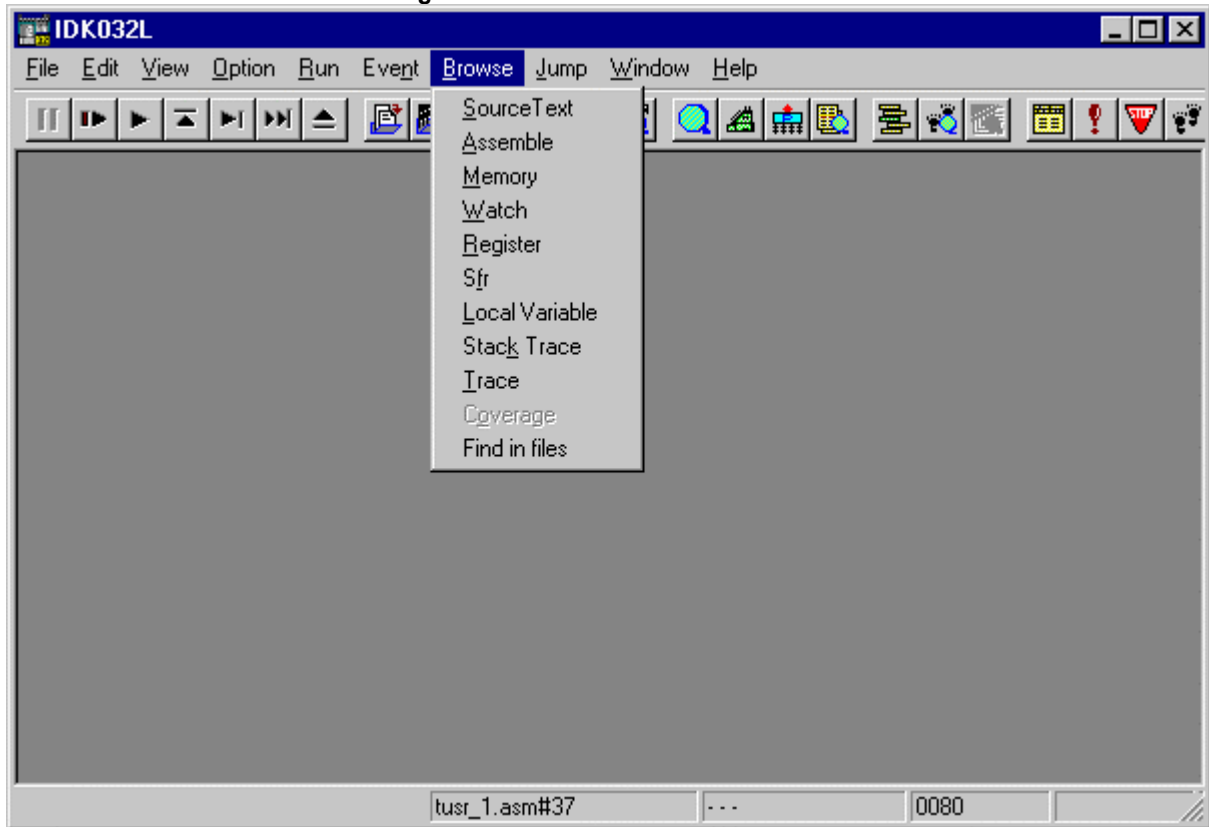










Table 4-9. Browse Command Descriptions

Command	Description
SourceText	Same as the  button: opens the Source Text window and displays the source text
Assemble	Same as the  button: opens the Assemble window and displays the assemble result
Memory	Same as the  button: opens the Memory window and displays the contents of memory
Watch	Same as the  button: opens the Watch window and displays the values of the variables
Register	Same as the  button: opens the Register window and displays the register contents
Sfr	Same as the  button: opens the SFR window and displays the contents of the SFRs
Local Variable	Opens the Local Variable window and displays the local variables
Stack Trace	Same as the  button: opens the Stack Trace window and displays the stack contents
Trace	Same as the  button: opens the Trace window

## (8) Jump Menu

Figure 4-14. Jump Commands



Table 4-10. Jump Command Descriptions

Command	Description
SourceText	Displays the source text for the jump destination address specified by the data value in the current window. If there is no line data in the jump destination address, jumping does not occur.
Assemble	Displays the Assemble result for the jump destination address specified by the data value in the current window
Memory	Displays the contents of memory, beginning at the jump destination address specified by the data value in the current window

## (9) Window Menu

Figure 4-15. Window Commands



Table 4-11. Window Command Descriptions

Command	Description
New Window	Opens a duplicate of the active 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
Refresh	Refreshes the displayed windows
Active	Allows displayed windows to refresh after execution
Static	Preserves the values of the displayed windows during execution

## (10) Help Menu

Figure 4-16. Help Commands

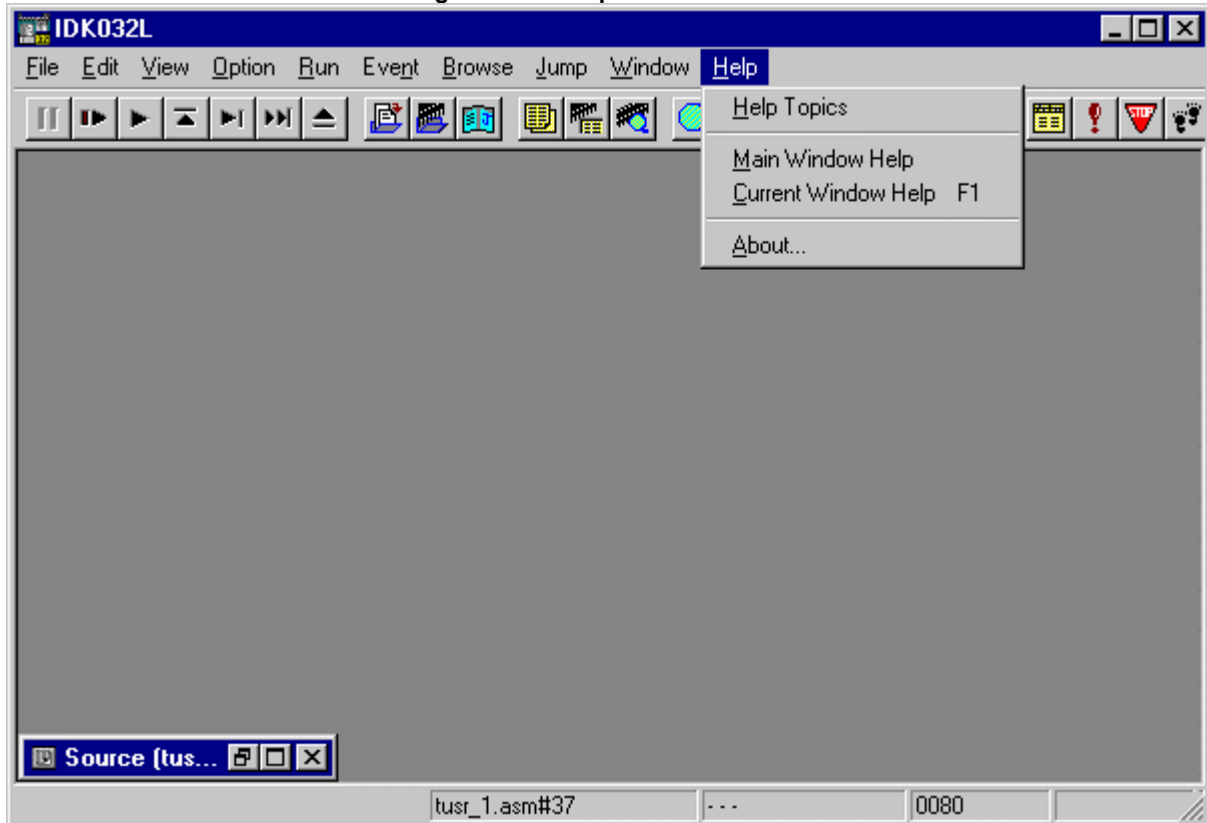


Table 4-12. Help Command Descriptions

Command	Description
Help Topics	Displays the Help window
Main Window Help	Displays the Help window for the Main window
Current Window Help	Displays the Help window for the current window
About...	Displays information about the debugger

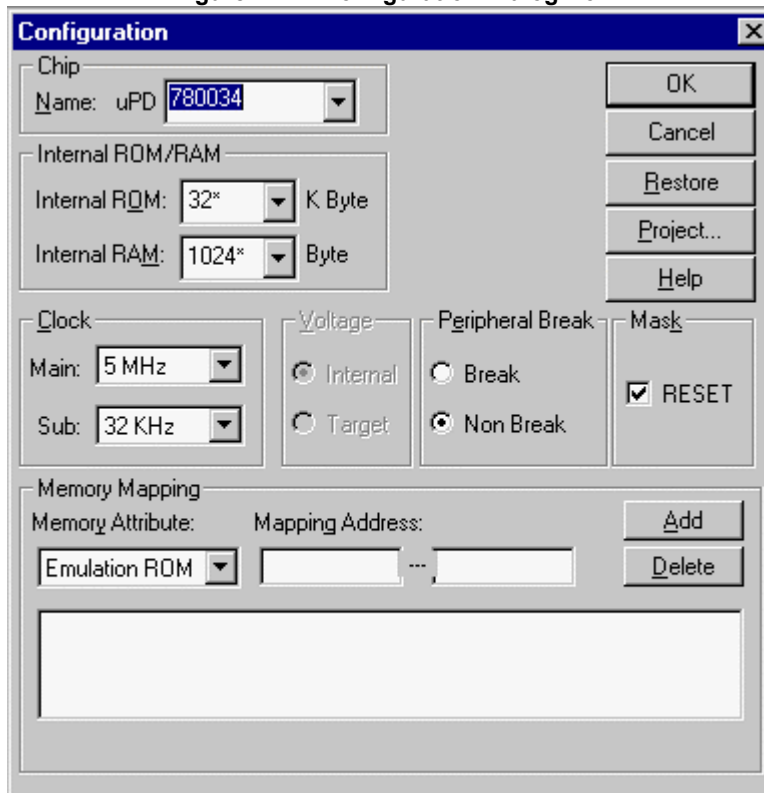
### 4.3.3 Dialog Boxes

#### (1) Configuration Dialog Box

The Configuration dialog box is used to display and set the operating environment of the low-cost emulator. The dialog box opens automatically upon power-up. It can also be opened in the Main window. From the **Option** menu, select **Configuration....**

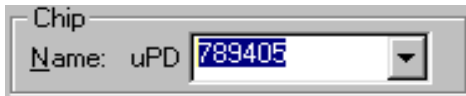
During debugging, this dialog box can be used to modify and add pin mask settings, location settings, and memory mapping settings as needed. The result of reading the project file is also reflected here.

Figure 4-17. Configuration Dialog Box

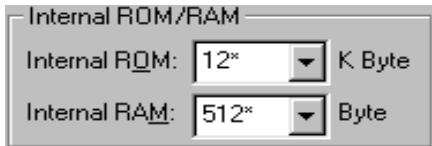


The Configuration dialog box has eight components.

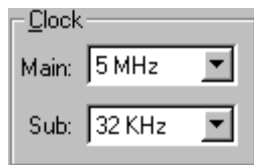
- (a) The **Chip** box allows you to select the emulation CPU upon initial startup.



- (b) The **Internal ROM/RAM** box appears after the emulation CPU is selected and displays the CPU's internal ROM and RAM sizes.



- (c) The **Clock** box allows you to select the main and subclock settings



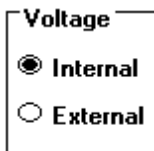
#### Main Clock Selections

Setting	Description
5 MHz	Built-in
10 MHz	Built-in
Alternate	User-installed crystal on the LCE
User	Clock provided by user target

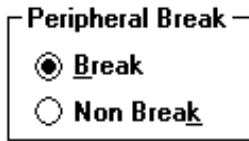
#### Subclock Selections

Setting	Description
32 kHz	Built-in
User	Clock provided by user target

- (d) The **Voltage** box is inactive. The LCE is driven by 3 volts or 5 volts, depending the jumper settings on the motherboard (refer to document no. 50889, *Getting Started*, for more information).

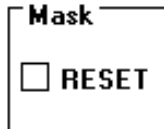


(e) The **Peripheral Break** box is inactive.



The **Peripheral Break** dialog box contains two radio buttons. The top radio button is selected and labeled **Break**. The bottom radio button is unselected and labeled **Non Break**.

(f) The **Mask** box masks the signal of the RESET pin from the target; used when the target is unstable during the debugging phase.



The **Mask** dialog box contains a single checkbox labeled **RESET**, which is currently unselected.

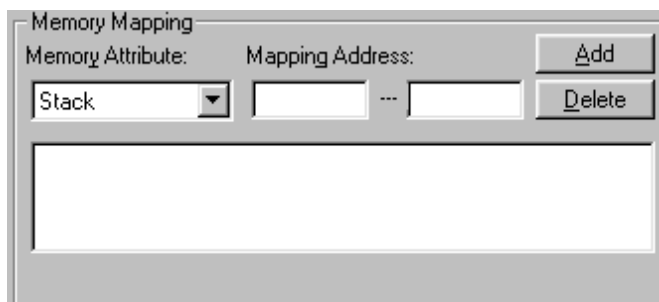
(g) The **Memory Mapping** box specifies the mapping attribute and address, and sets mapping. The following types of mapping attributes can be selected.

*Note, however, that the Emulation ROM, Emulation RAM, Target, and I/O Protect cannot be selected for devices without external space.*

#### Memory Mapping Selections

Setting	Description
Emulation ROM	Selects an in-circuit emulator alternate ROM
Emulation RAM	Selects an in-circuit emulator alternate RAM
Target	Selects a target memory
I/O Protect	Selects an I/O protect area
Stack	Select a memory in the stack area

The I/O Protect area can only be set inside the area set in the Target and the external SFR area. The area set as I/O Protect cannot be read unless it has been registered as an I/O port in the SFR window, or registered in the Watch window. If it is necessary to read this area, execute a forcible read in these windows.



The **Memory Mapping** dialog box features a **Memory Attribute:** dropdown menu currently set to **Stack**, and a **Mapping Address:** field with two input boxes separated by an ellipsis (...). To the right of the address field are two buttons: **Add** and **Delete**. Below these controls is a large empty rectangular area.

The Memory Address area specifies the address to be mapped.

Input the mapping-start address and the mapping-end address from the keyboard.


To add memory mapping:

After specifying the Memory Attribute and Mapping Address, click the  button.






Memory mapping corresponding to the Memory Attribute and to the Memory Mapping address range is added.

Stack area is set only in the internal RAM area (cannot be set in the internal expansion RAM area).

To delete memory mapping:

Select the displayed area where you desire to delete mapping, then click the  button. The currently selected mapping will be deleted.

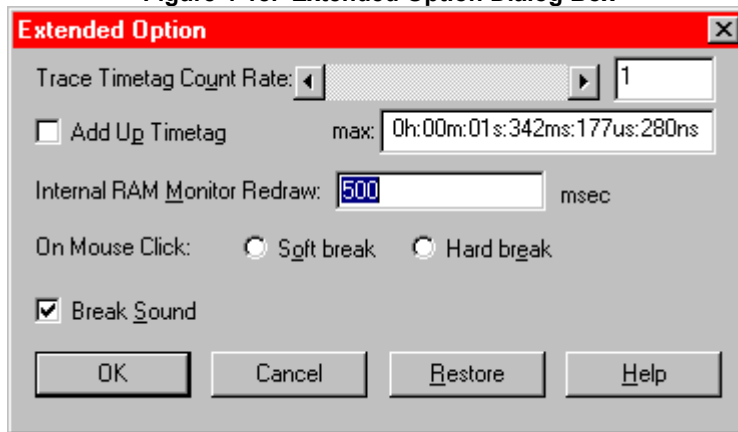
**(h) Command buttons**

Button	Description
	Sets the current environment and closes the dialog box
	Cancels the changes and closes the dialog box
	Restores the previous environment
	Opens a project file
	Opens the Help window to explain the Configuration dialog box

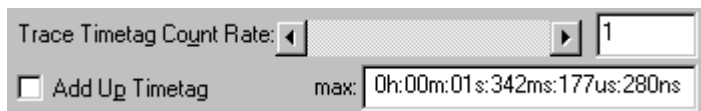
**(2) Extended Option Dialog Box**

The Extended Option dialog box allows you to display and set extended options for the debugger. From the **Option** menu, select **Extended Option...** to open the dialog box.

Figure 4-18. Extended Option Dialog Box



(a) **Trace Timetag Count Rate** allows you to set the rate for the trace timetag; the Add Up Timetag check box allows you to enable/disable the Add Up Timetag for display. (Not supported)



(b) **Real-Time Internal Monitor Redraw** allows you to set the rate (in 1 ms units) for real-time internal RAM sampling.

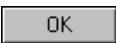


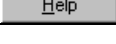


A screenshot of a dialog box with a label 'Internal RAM Monitor Redraw:' followed by a text input field containing the number '500' and the unit 'msec'.

- (c) The break option **On Mouse Click** allows you to select mouse-driven software or hardware breaks (**not supported**). The Break Sound check box allows you to enable/disable the break-triggered sound.

A screenshot of a dialog box showing two radio button options: 'Soft break' and 'Hard break'. Below them is a checked checkbox labeled 'Break Sound'.

- (d) **Command buttons**

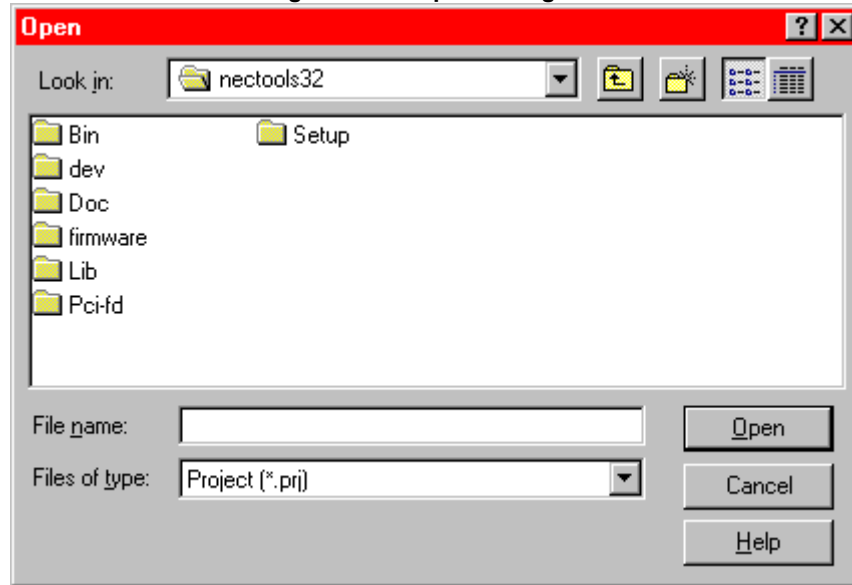
Command	Description
	Accepts this change and closes the dialog box
	Cancels this change and closes the dialog box
	Restores the previous settings of the window
	Opens the Help dialog box

**(3) Open Dialog Box**

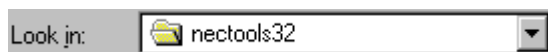
From the Open dialog box, you can restore the debugging environment to its previous state. After a file is loaded, the displayed window size and position revert to their previous states. (The analyzer relationship is not restored.)

To open the dialog box, select **File → Project → Open....**

**Figure 4-19. Open Dialog Box**



- (a) **Look in:** Click the folder name to display the project files in that folder. The display formats are  
 [-x-]: Drive name  
 [xxx]: Directory name



- (b) **File name:** Click the name of a .PRJ file to select it and then click **Open**. (You can also double-click the file name to select and open it.)



- (c) **Command Buttons**

	Loads the selected project file and sets the environment
	Closes the Project File Load dialog box
	Opens the Help window

**(d) Contents of load operation**

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.

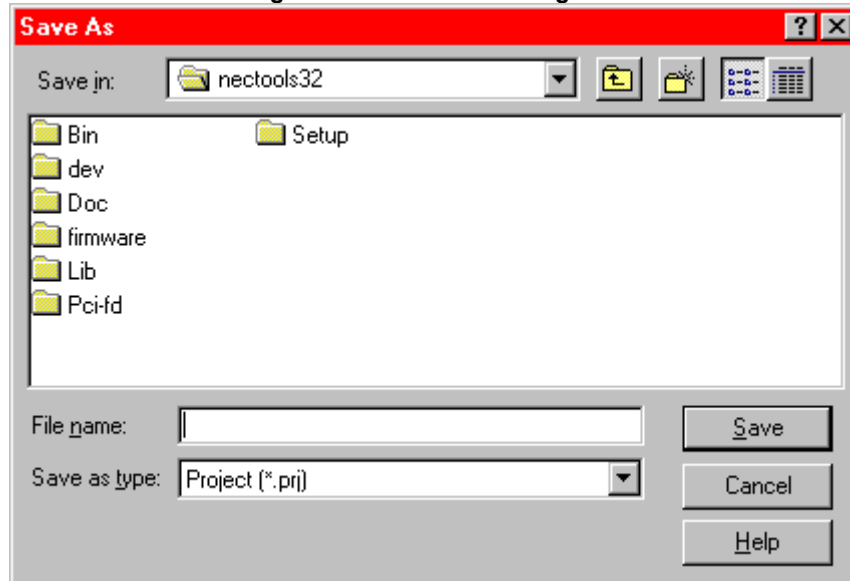
**Table 4-14. Load Contents**

<b>Window</b>	<b>Setting Data</b>
Configuration dialog box	All of the items
Main window	Display position; toolbar, status bar, and button display data; execution mode data; trace on/off data
Load Module Selection dialog box	Download file data
Extended Option Setting dialog box	Setting data
Source Path Specification dialog box	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 box	Window display data
Break dialog box	Window display data
Trace dialog box	Window display data
Event Set dialog box	Window display data
Register window	Window display data; display bank
Variable window	Window display data; displayed variable data

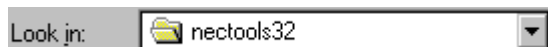
**(4) Save As Dialog Box**

The Save As dialog allows you to save the debugging environment, in other words, the active window. To open this dialog box, select **File → Project → Save As**.

To save a previously loaded or saved project file with the same file name, select **File → Project → Save**. In this case, the Save As dialog box does not open and the save is made to the existing file name.

**Figure 4-20. Save As Dialog Box**

- (a) **Save in:** Click the folder name to display the project files in that folder. The display formats are  
 [-x-] : Drive name  
 [xxx] : Directory name



- (b) **File name:** Click the name of a .PRJ file and then click **Save**. (You can also double-click the file name to save it.)

**(c) Command Buttons**

	Saves the environment in the selected file name
	Cancels the selection
	Opens the Help window

**(d) Saved contents**

The following items are saved in the project file.

**Table 4-15. Contents for Saving**

<b>Window</b>	<b>Setting Data</b>
Configuration dialog box	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 box	Download file data
Extended Option setting dialog box	Setting data
Source Path Specification dialog box	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 box	Window display data
Break dialog box	Window display data
Trace dialog box	Window display data
Event Set dialog box	Window display data
Register window	Window display data, display bank
Variable window	Window display data, displayed variable data

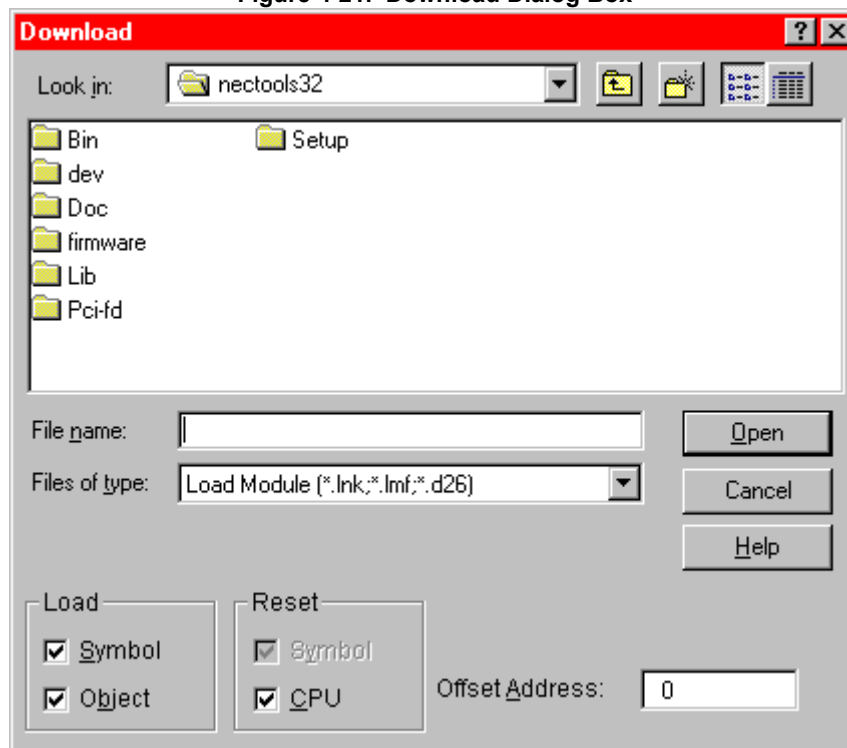
**(5) Download Dialog Box**

The Download dialog box allows you to download a file in one of the following formats to the LCE or target:

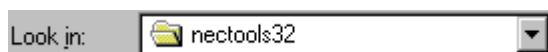
- Object file in the load module format (\*.LNK)
- Intel® extended hexadecimal format (\*.HEX)
- Motorola® hexadecimal S-type format (standard address) (\*.HEX)
- Extended Tektronix® hexadecimal format (\*.HEX)

If a file other than an object file in load module format is loaded, source debugging is not possible. To open the Download dialog box, select **File** → **Download...**

**Figure 4-21. Download Dialog Box**



- (a) **Look in:** Click the folder name to display the project files in that folder. The display formats are
- [-x-]: Drive name
  - [xxx]: Directory name



- (b) **File name:** Click the name of a file to select it and then click **Open**. (You can also double-click the file name to select and open it.) The default extension is .LNK.

File name:

Files of type: Load Module (\*.lnk;\*.lmf;\*.d26)

(c) **Load** and **Reset** allow you to set the load conditions, where

<b>Symbol</b>	Specifies to read or not read symbol data
<b>Object</b>	Specifies to read or not read object data
<b>Reset</b>	Resets the symbol and CPU after downloading
<b>Offset Address</b>	Specifies the offset address

Load:  Symbol  Object

Reset:  Symbol  CPU

Offset Address:

(d) **Command buttons**

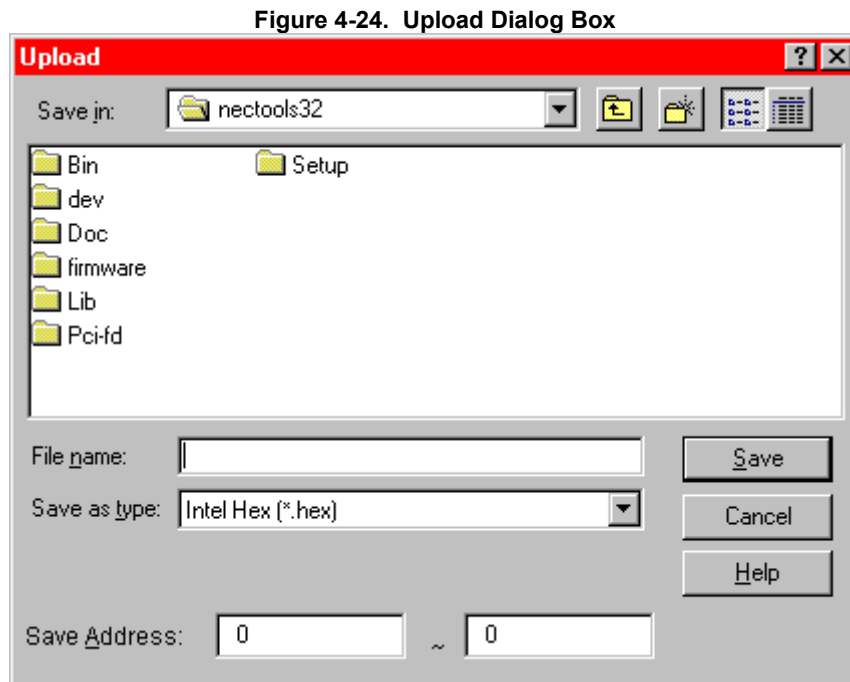
<input type="button" value="Open"/>	Opens the selected file
<input type="button" value="Cancel"/>	Cancels the changes and closes the dialog box
<input type="button" value="Help"/>	Opens the Help window

**(6) Upload Dialog Box**

The Upload dialog box allows you to specify the name and format of the memory contents to be uploaded. The file format can be one of the following:

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

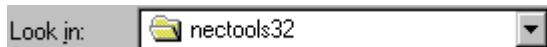
To open this dialog box, select **File → Upload...**



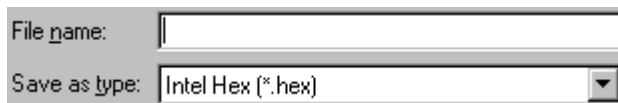
**(a) Save in:** Click the folder name to display the project files in that folder. The display formats are

[-x-] : Drive name

[xxx]: Directory name



**(b) File name:** Click the name of a file to select it and then click **Open**. (You can also double-click the file name to select and open it.) The default extension is .HEX.





(c) **Save Address** specifies the address range in memory to be uploaded.

Save Address:  ~

(d) **Save as type** specifies the file format of the object to be uploaded.

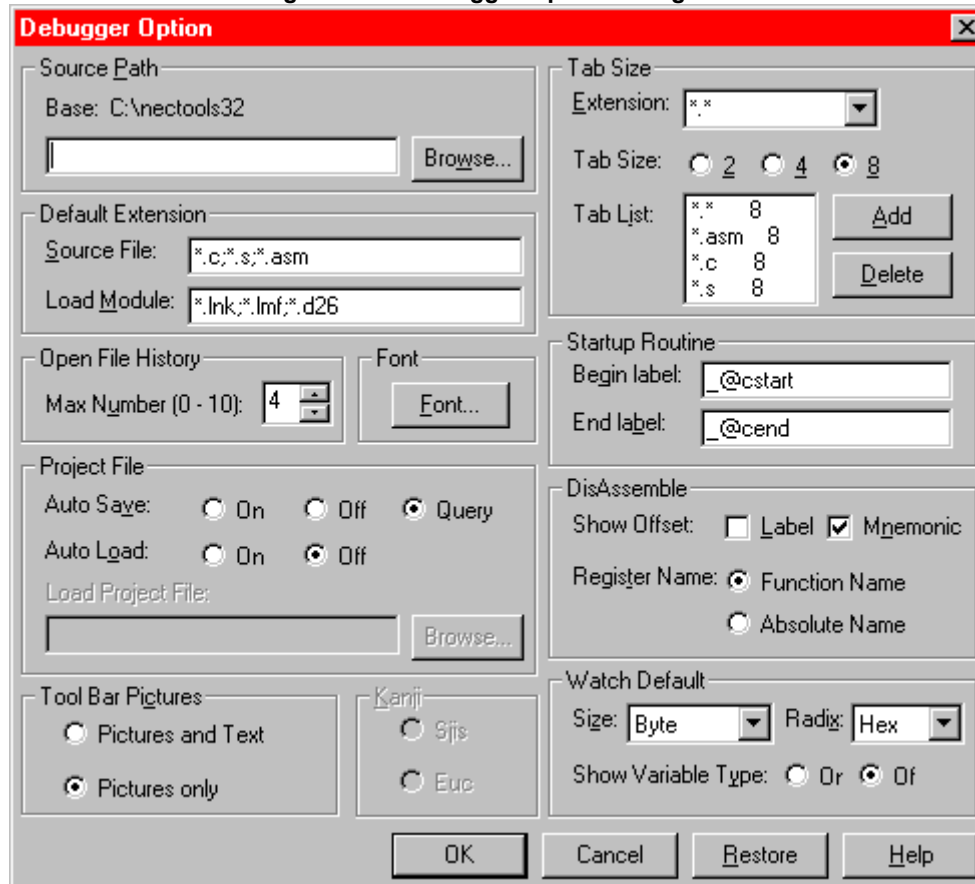
- Intel: Intel extended hexadecimal format
- Motorola: Motorola hexadecimal format S-type format (standard address)
- Tektro: Extended Tektronix hexadecimal format
- Binary: Standard binary data format

(e) **Command Buttons**

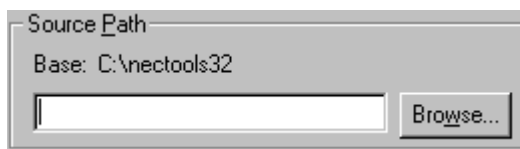
<input type="button" value="Save"/>	Saves the memory contents in the address range to a file in the specified directory
<input type="button" value="Cancel"/>	Closes the dialog box
<input type="button" value="Help"/>	Opens the Help window

**(7) Debugger Option Dialog Box**

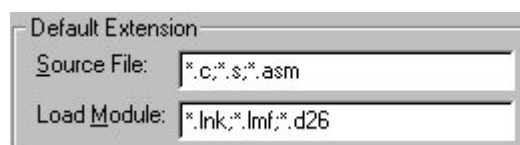
The Debugger Option dialog box allows you to specify the path for sources located in multiple directories. To open the dialog box, select **Option** → **Debugger Option...** from the Main window.

**Figure 4-23. Debugger Option Dialog Box**

- (a) The **Source Path** box allows you to specify a path for the source file. For example, if the source were in directories (1) a:\78k\c, (2) b:\src, and (3) c:\asm, then the source path would be *a:\78k\c b:\src c:\asm*.



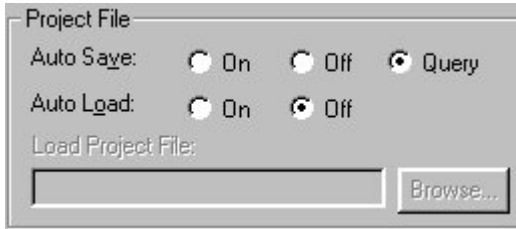
- (b) The **Default Extension** box allows you to specify the default extensions for the source file and the load module file.



- (c) The **Open File History** box displays the number of recently opened files under the File menu and allows for easy loading of a file or project in the LCE.



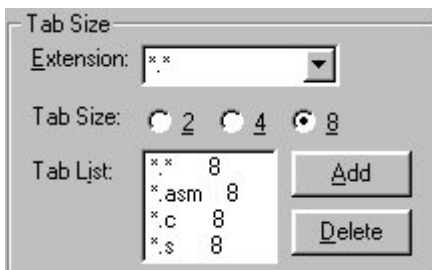
- (d) The **Project File** box allows you to enable or disable the auto save and auto load functions.



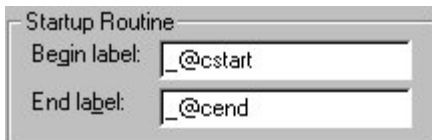
- (e) The **Tool Bar Pictures** box sets the appearance of the icons on the Main window.



- (f) The **Tab Size** box sets the tabs for the display format of the Source window, Assemble window, and so forth.



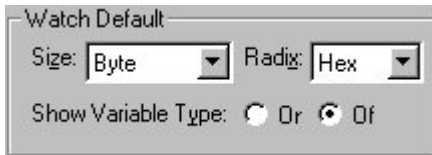
- (g) The **Startup Routine** box indicates the start and end labels of the startup routine.



- (h) The **DisAssemble** box specifies the display format for the Assemble window.



- (i) The **Watch Default** box specifies the display format for the Watch window.

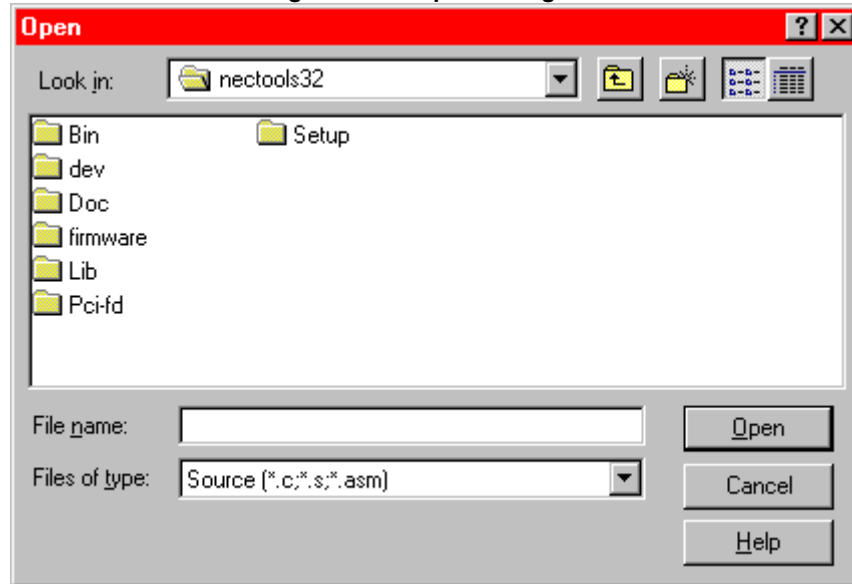


- (j) **Command Buttons**

	Accepts the source path setting and closes the dialog box
	Cancels the source path setting and closes the dialog box
	Restores the previous settings
	Opens the Help window
	Browses for the file

**(8) Open Dialog Box**

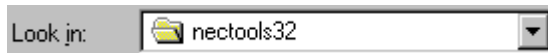
To display text in the Source Text window, select **File → Open...** or press CTRL+O. *The Source Text window must be open to perform this operation.*

**Figure 4-24. Open Dialog Box**

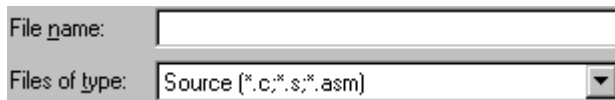
(a) **Look in:** Click the folder name to display the project files in that folder. The display formats are

[-x-] : Drive name

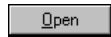
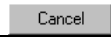

[xxx]: Directory name




(b) **File name:** Double-click the name of the source file to select and open it. In the **Files of type** line, select the appropriate file extension: .C for C language, .S for structured assembly language, or .ASM for assembly language.



(c) **Command Buttons**

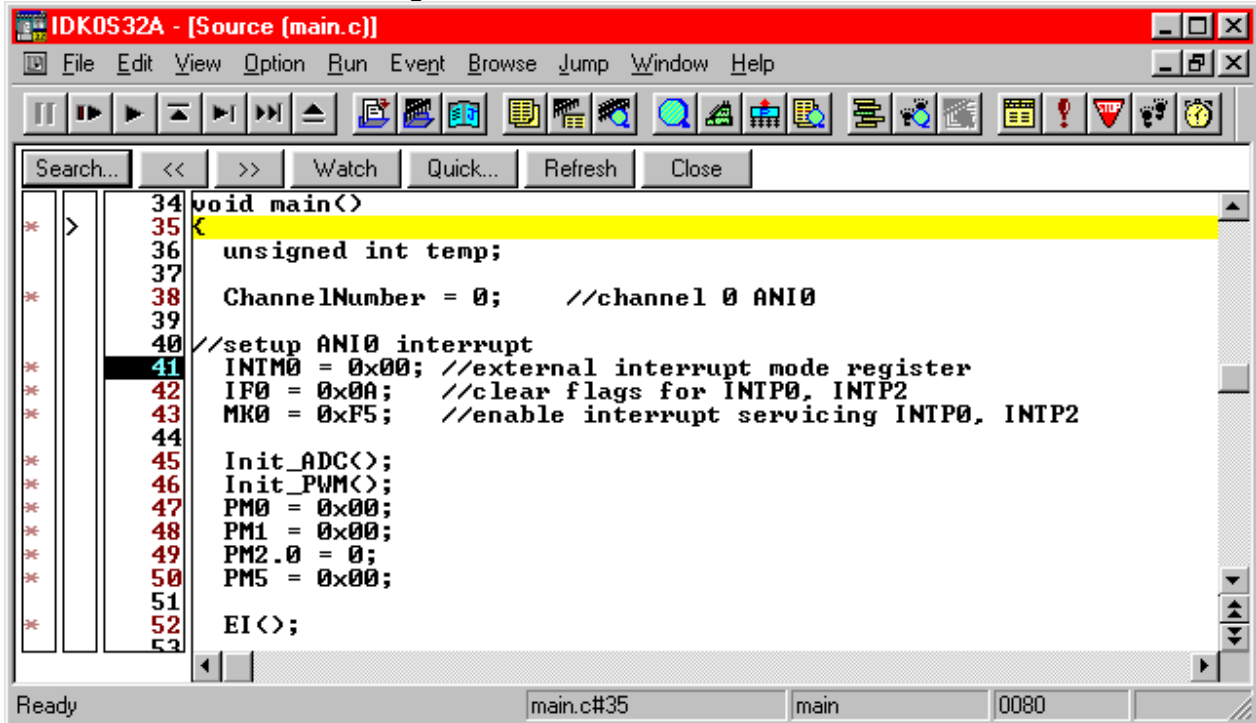
	Opens and displays the selected source file or function in the Source Text window
	Closes the Source File Selection dialog box
	Opens the Help window

**(9) Source Text Window**

To display source text, select **Browse → Source Text** or click the  command button. To display source text from another window, use the jump function. Choose a pointer location and then select **Jump → Source Text** or press CTRL+U.

**Table 4-16. Jump Function (Source Text Window)**

Window	Pointer	Operating Method		
		(1)	(2)	(3)
Disassemble window	Address display 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"/>	-

**Figure 4-25. Source Text Window****(a) Point mark area**

The point mark area is the first column in the Source window. It is used to set and delete breakpoints at the asterisk locations, and to display the event setting state.

*i. Breakpoint set and delete functions*

Clicking in this area allows you to set and delete breakpoints, as explained in the following table.

**Table 4-17. Cursor Position for Breakpoint Set/Delete**

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

*ii. Event display function*

This function displays the event settings. If an execution event or access fetch event is set in the corresponding source line, the mark corresponding to that event type is displayed.

**Table 4-18. Event Mark Display**

Mark	Mark Meaning
E	Event condition is set
L	Last phase in the event link is set
B	Break event is set
T	Trace event is set
A	Multiple events are set

**(b) Current PC position**

The '>' symbol points to the current PC value and highlights the line in yellow. Pressing the mouse at this position displays the PC register value in a pop-up window.

**(c) Line number area**

The line numbers of the source text are displayed in this area. Highlighting and then right-clicking a line displays a window where you can perform the following functions.

*i. Come function*

This function executes the user program until reaching the selected line. In this mode, currently set break events are not generated. This function is executed by selecting the line numbers where the breaks should occur, and then selecting **Run → Come Here** from the Main window.

*ii. Break event setting function*

The break event is set at the first address corresponding to the selected line numbers. To set a breakpoint using execution events, highlight the line number where the break event is set and then select **Run → Break Point** from the Main window.








*iii. Jump function (Assemble or Memory)*

This function jump to the first address corresponding to the selected line numbers in the Assemble window or Memory window. The jump destination window is displayed from the jump pointer.

**(d) Source display area**

Source text can be displayed in the source display area. C and assembly text may be displayed simultaneously in this window. Right-click in any part of the source display area and select Mix from the pop-up window.

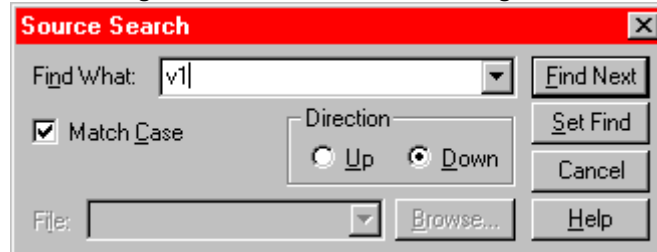
**(e) Command buttons**

	Search for a string in Source window
	Search back for selected text in the Source window
	Search forward for selected text in the Source window
	Opens the watch dialog box and displays the highlighted variable from the Source window
	Opens the Quick Watch window and displays the variables
	Refreshes the Source Text window
	Closes the Source window



**(10) Source Search Dialog Box**

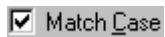
This operation performs a data or string search and displays the search result in the active window. To open the Source Search dialog box, click **Search** from the Source Text, Assemble or Memory window.

**Figure 4-26. Source Search Dialog Box**

- (a) **Find What:** allows you to specify the search data. By default, the string highlighted in the active window is displayed, but when needed, changes can be typed from the keyboard.



- (b) **Match Case:** allows you to specify whether to distinguish between upper- and lower-case letters. The default is to match case.



- (c) **Direction:** specifies whether to conduct a forward search (Up) or a backward search (Down).



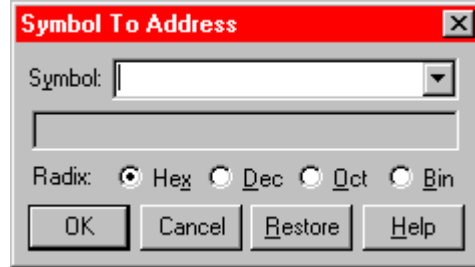
- (d) **Command buttons**

Find Next	Searches for the specified search data in accordance with the conditions
Set Find	Sets the data search
Cancel	Exits the Search dialog box
Help	Opens the Help window

**(11) Symbol to Address Dialog Box**

To display the address of a specified variable, select **View → Symbol...** from the Main window.

**Figure 4-27. Symbol to Address Dialog Box**

**(a) Variable specification area**

The variable name and line number for the address conversion are specified as shown in the following table. After the data is input, press the **RETURN** key from the keyboard to display the address value in the variable address display area.

Function and variable	<symbol name>
SFR	Sfname (SFT name)

To specify a function or variable name, use an underline character (\_) at the beginning and the sharp (#) character as the separator between a file name and a function or variable name.

**(b) Variable address display area**

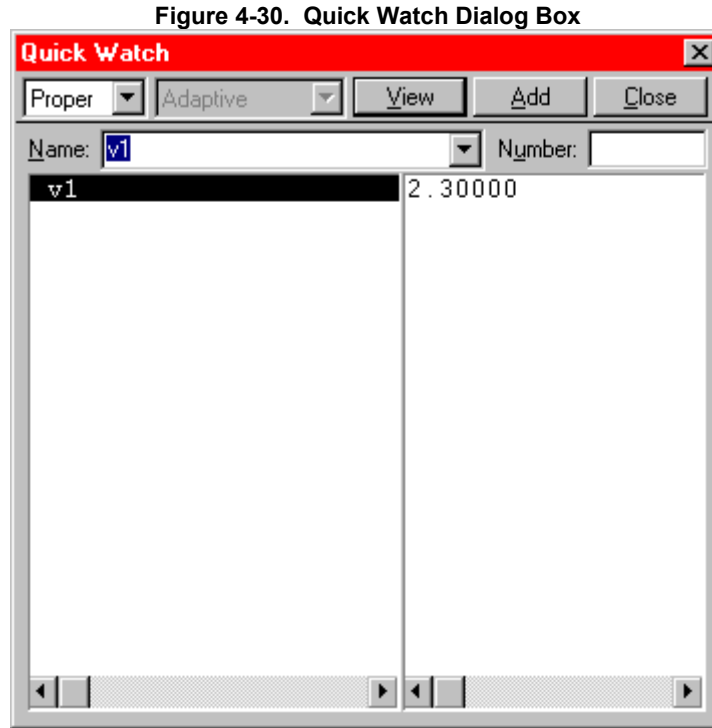
This area displays the address of the variable specified in the variable specification area. The format can be changed by

**(c) Command buttons**

OK	Accepts settings and closes the dialog box
Cancel	Closes window
Restore	Restores previous data
Help	Opens the Help window

**(12) Quick Watch Dialog Box**

This dialog box allows you to temporarily display the value of the variable specified in the Source Text window. To perform this operation, select the variable in the Source window and then select **View** → **Quick Watch** ....



**(a) Name:** allows you enter a variable name to be displayed in the Source Text window.



**(b) Variable value:** displays the specified variable value.



**(c) Command buttons**

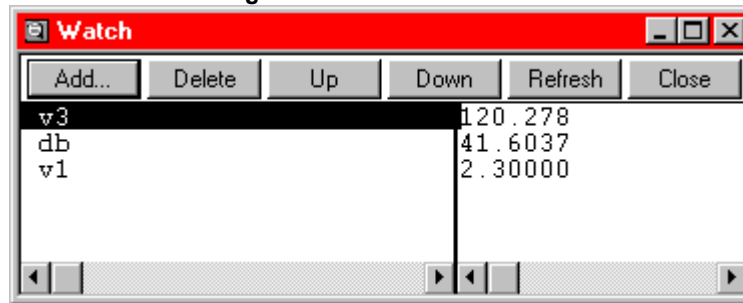
Number: <input type="text"/>	Displays the number variable
Proper <input type="text"/>	Displays the format of the data
Close	Closes the dialog box

**(13) Watch Window**

This window allows you to display (View mode) and change (Modify mode) the value of the variable specified in the Source Text window. The variable display is added for each display requirement. If the same variable is added, the addition is not displayed.

In the Main window, select **View → View Watch** or highlight the variable in the Source window and then click the Watch button in the source window.

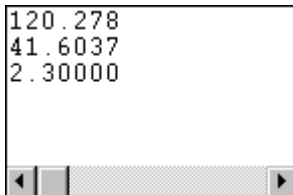
**Figure 4-31. Watch Window**



- (a) The **variable name display 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.



- (b) The **variable value display/setting area** displays the variable values. When the variable is a pointer variable, the address value or data value is displayed. The notation of the display may be changed by right-clicking the value and selecting the desired notation from the pop-up window.



- (c) **Command buttons**

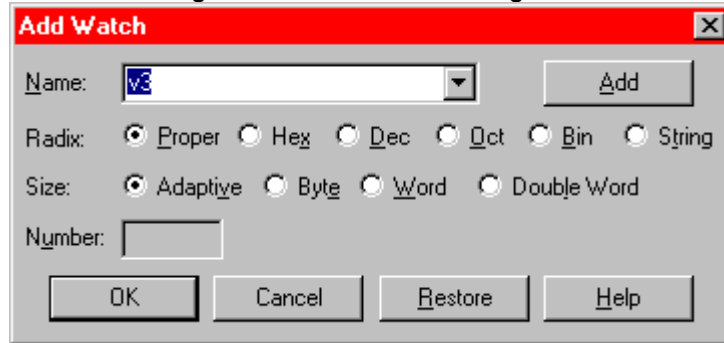
Add	Adds variable to the Watch window
Delete	Deletes selected variable from the Watch window
Up	Scrolls up in the window
Down	Scrolls down in the window
Refresh	Refreshes values of the variables

Close	Closes the window
-------	-------------------

**(14) Add Watch Dialog Box**

The variable displayed in the Watch window is added and registered. To open this dialog box, select **View → Add Watch**.

**Figure 4-30. Add Watch Dialog Box**



**(a) Name:** specifies the variable name to be added.



To specify a variable, use an underline character () at the beginning of the name. Use the sharp (#) character as a separator between the file name and variable name.

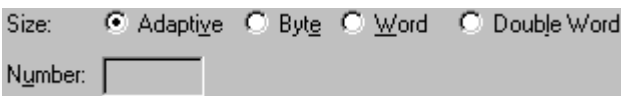
Variables	<u>_</u> func: function, variable name (for example, <u>_</u> int1) file#_func: file = file name (for example, main.x#_int1)
SFR	Sfrname: SFR name (for example, PCC)

**(b) Radix:** allows you to select the notation of the value of the specified variable.



Proper	Automatically selects the proper format
Hex	Hex format
Dec	Decimal format
Oct	Octal format
Bin	Binary format
String	String format (ASCII)

**(c) Size and Number:** allow you to specify the size and number of the added variable. Selecting C language in the Type area cannot be specified.





- i. **Size:** specifies the variable size: adaptive (automatically selects the format), byte, word, or double word
- ii. **Number:** specifies the number of variables

**(d) Command buttons**

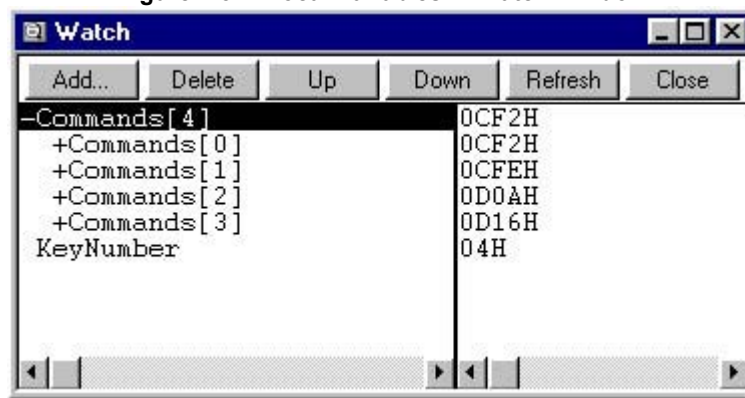
OK	The variable is added and registered to the Variable window
Cancel	Closes the dialog box
Restore	Restores previous data
Help	Opens the Help window

**(15) Local Variables in Watch Window**

This window allows you to display and change the local variables in the current function. The boundary line between the local variable name display area and the local variable value display/setting area can be moved by dragging and dropping the cursor once it changes from  to .

To open the window, select **Browse → Local Variable...** from the Main window.

**Figure 4-31. Local Variables in Watch Window**



- (a) **Display area** displays the local variable names. A variable displayed with "+" at the beginning indicates a pointer variable. Double-clicking a pointer variable displays that data value in the variable value display/setting area. The "+" display switches to the "-" display.

```

-Commands[4]
+Commands[0]
+Commands[1]
+Commands[2]
+Commands[3]
KeyNumber

```


- (b) **Local variable value display/setting area** displays local variable values. When the variable is a pointer variable, the address value or data value is displayed

```

0CF2H
0CF2H
0CFEH
0D0AH
0D16H
04H

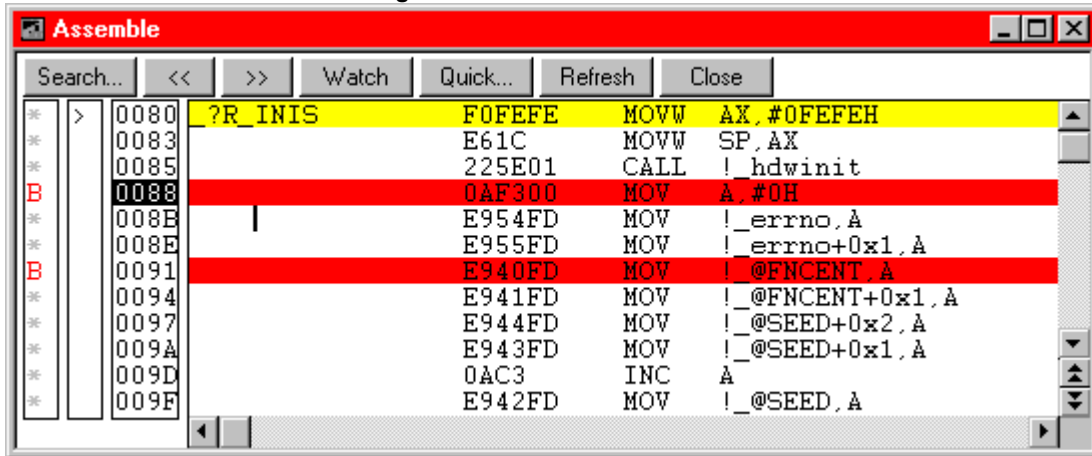
```

(17) Assemble Window

This window allows you to assemble online and display the disassemble result. To open the Assemble window, select **Browse → Assemble** from the Main window or click .

To display the corresponding assemble line from another window, move the program cursor to the selected position, right click and select **Assemble**.

Figure 4-33. Assemble Window



(a) Point mark area: breakpoints are set and deleted and events displayed in this area.



i. Breakpoint set/delete function

Clicking the mouse in this area sets and deletes breakpoints.

Table 4-20. Cursor Position for Breakpoint Set/Delete

Location	Color	Mouse Operation	Operation
'B' mark is displayed.	Red, black	Click	Delete breakpoint
'B' mark is not displayed or something else is displayed.	–	Click	Set breakpoint
	–		

**ii. 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.

**Table 4-21. Event Mark Display**

Mark	Description
E	Event condition is set
L	Last phase in the event link is set
B	Break event is set
T	Trace event is set
A	Multiple events are set

- (b) **Current PC location:** Holding down the mouse at the current PC location displays the corresponding PC register value in a pop-up window.



- (c) **Address display:** The assemble starting address is displayed in this area. Highlighting a line and right-clicking displays a pop-up window where you can perform the following functions.

```
0080
0083
0085
0088
008E
008E
0091
0094
0097
009A
009D
009F
```

**i. Come function**

This function executes the user program until reaching the selected address. In the Main window, select **Execute → Come**.

**ii. Break event setting function**

A breakpoint can be set at a selected address using an execution event. Select the address where you want to set the breakpoint and select **Run → Break Point** from the Main window.

**iii. Jump function (Assemble or Memory)**

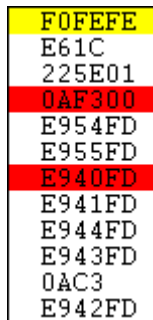
With the selected address as the jump pointer, you can jump to the Source Text or Memory window. The jump destination window is displayed from the jump pointer. Highlight the address destination in the Source Text window and then select **Jump → Source Text...** or press CTRL+U.



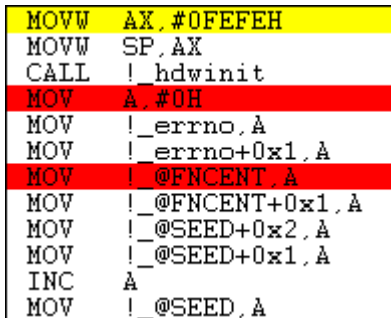
(d) **Label display area** displays the labels.



(e) **Data display area** displays the mnemonic data.




(f) The **Mnemonic display/modify area** displays the disassembly result and can be modified directly.



(g) **Command Buttons**

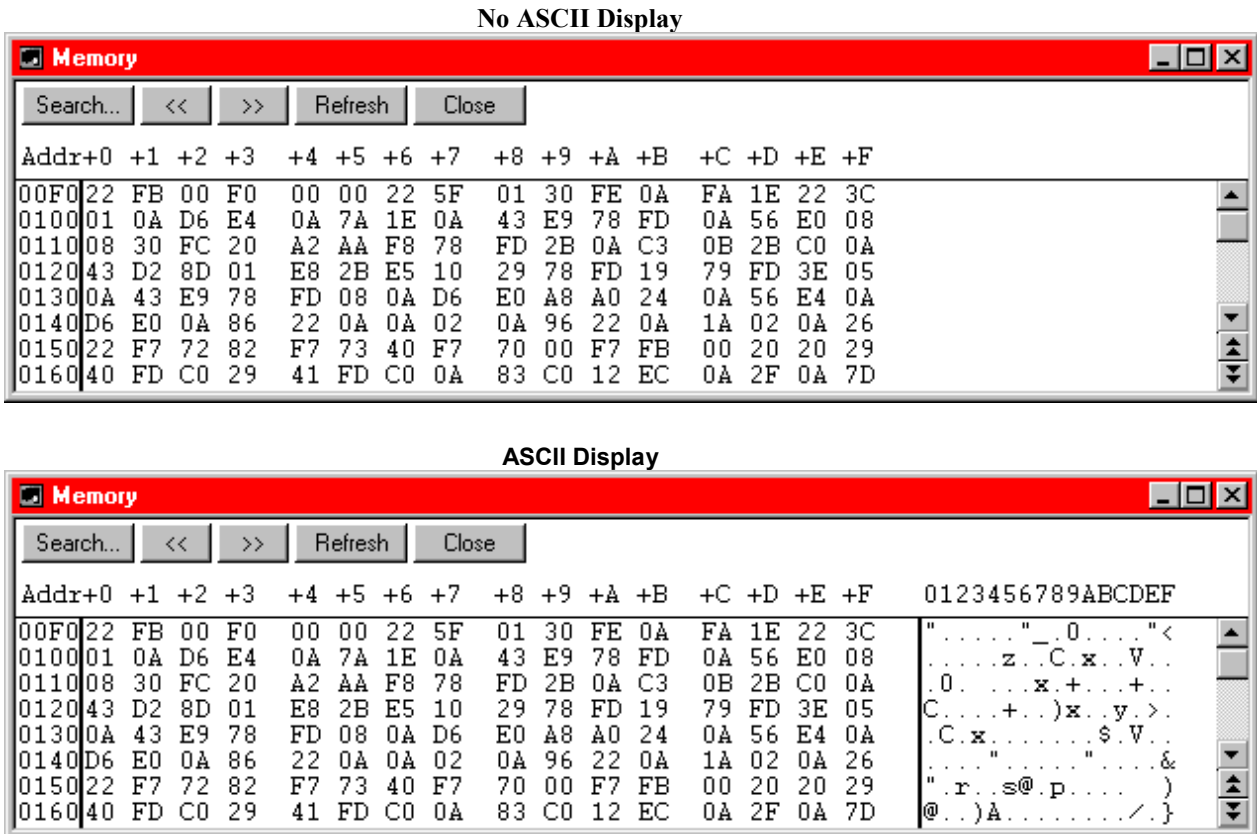
Search...	Searches for the highlighted string
<<	Searches backward
>>	Searches forward
Watch	Opens the Watch window with variable
Quick...	Opens the Quick Watch window
Refresh	Refreshes data from the window
Close	Closes the window

(18) **Memory Window**

The Memory window allows you to display and change its contents. To open the window, select **Browse → Memory...** or click . Right-clicking any part of the window displays a pop-up window

where you can change the display format.

Figure 4-34. Memory Window



(a) **Address display area:** displays the memory addresses using two functions.

```

Addr
00F0
0100
0110
0120
0130
0140
0150
0160
    
```

#### 4. OPERATION

(b) **Memory display area** displays the contents of memory, which can be modified directly.

+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
22	FB	00	F0	00	00	22	5F	01	30	FE	0A	FA	1E	22	3C
01	0A	D6	E4	0A	7A	1E	0A	43	E9	78	FD	0A	56	E0	08
08	30	FC	20	A2	AA	F8	78	FD	2B	0A	C3	0B	2B	C0	0A
43	D2	8D	01	E8	2B	E5	10	29	78	FD	19	79	FD	3E	05
0A	43	E9	78	FD	08	0A	D6	E0	A8	A0	24	0A	56	E4	0A
D6	E0	0A	86	22	0A	0A	02	0A	96	22	0A	1A	02	0A	26
22	F7	72	82	F7	73	40	F7	70	00	F7	FB	00	20	20	29
40	FD	C0	29	41	FD	C0	0A	83	C0	12	EC	0A	2F	0A	7D

(c) **ASCII display area** displays the contents of memory in ASCII format. The contents can be changed directly from the window.

0123456789ABCDEF

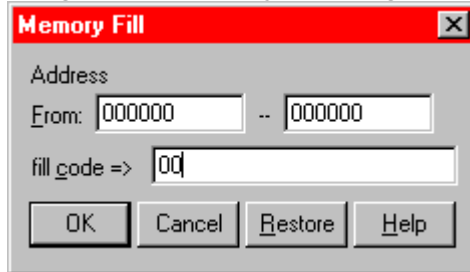
```
" ..... ".0....."<
.....z..C.x..V..
.0.....x.+...+...
C.....+..)x..y.>.
.C.x.....$.V..
.....".....&
".r..s@.p.....)
@...)A...../..}
```

(d) **Command buttons**

Search...		Searches for the highlighted string
<<		Searches backward
>>		Searches forward
Refresh		Refreshes data from the window
Close		Closes the window

**(19) Memory Fill Dialog Box**

In this dialog box, the memory contents are initialized to the specified code. This dialog box is in the active window state and can be opened from the Main window by selecting **Edit → Memory → Fill...**

**Figure 4-41. Memory Fill Dialog Box**

- (a) **Address range specification area** specifies the address range for the memory contents to be initialized. The input is from initialization-start-address to initialization-end-address.

**Address**

From:  --

- (b) **Data specification area** specifies the initialization data, up to 16 bytes of string data.

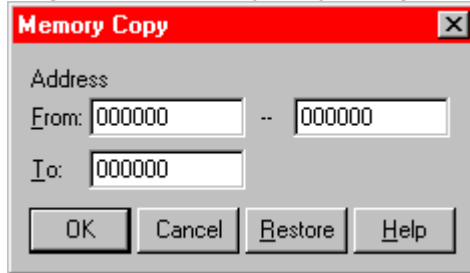
fill code =>

**(c) Command buttons**

OK	Initializes the memory
Restore	Restores the input data to the original value
Cancel	Closes the Memory Fill dialog box
Help	Opens the Help window

**(20) Memory Copy Dialog Box**

This dialog box is in the active window state and copies memory from one location to another. The dialog box can be opened from the Main window by selecting **Edit → Memory → Copy...**

**Figure 4-36. Memory Copy Dialog Box**

- (a) **Address range specification area** specifies the beginning and ending addresses of the copy source and the beginning address of the copy destination.

**Address**

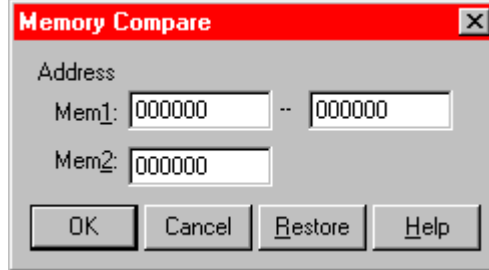
From:  --   
 To:

**(b) Command buttons**

OK		Copies the memory
Restore		Restores the input data to its original values
Cancel		Closes the Memory Copy dialog box
Help		Opens the Help window

**(21) Memory Compare Dialog Box**

This dialog box allows you to compare memory contents. This dialog box is in the active window state and can be opened from the Main window by selecting **Edit → Memory → Compare...**

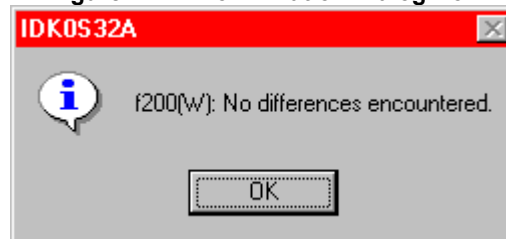
**Figure 4-37. Memory Compare Dialog Box**

- (a) **Comparison range specification area** specifies the source and destination address of the memory contents.

**Mem1:** [compare-source-starting-address] – [compare-source-ending-address].

**Mem2:** Input the address of the compare destination.

Click **OK** to start. If there are no differences, the Confirmation dialog box appears. Otherwise, the differences are displayed in the Memory Comparison Result dialog box. To end the compare, click **OK**.

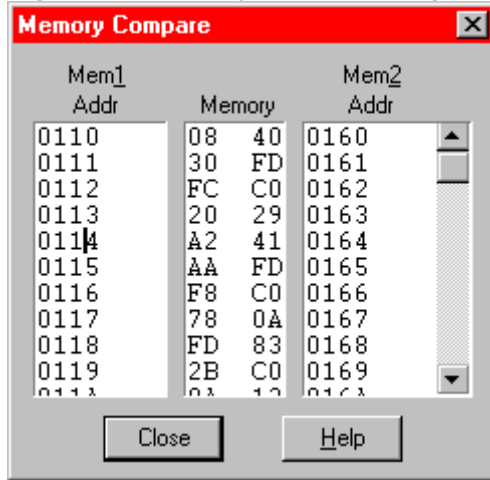
**Figure 4-44. Confirmation Dialog Box****(b) Command Buttons**

<b>R</b> estore		Restores the input data to its original values
<b>C</b> ancel		Closes the Memory Compare dialog box
<b>H</b> elp		Opens the Help window

(22) Memory Compare Result Dialog Box

This dialog box displays the result of a memory comparison when the compare operation shows differences in the memory contents.

Figure 4-39. Memory Compare Dialog Box



(a) Comparison result display area displays the memory comparison result.

Source Addr	Memory	Destination 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

**Source Addr** or **Mem1 Addr** displays the compare source address where there was a comparison error.

**Memory** displays the data where there was a discrepancy. Comparison source data is displayed on the left and comparison destination data on the right.

**Destination Addr** or **Mem2 Addr** displays the destination address where there was a comparison error.

(b) Command buttons

Close	Closes the Memory Compare dialog box and highlights the searched address in the Memory window
Help	Opens the Help window

**(23) Stack Trace Window**


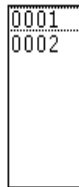
The stack contents of the current user program are displayed. To open the window, select **Browse** → **Stack Trace** from the Main window or click the  command button.

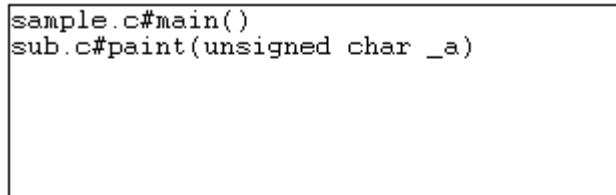
Figure 4-40. Stack Trace Window



- (a) **Stack frame number display area** displays the number assigned to the stack contents. This number starts at 1 and becomes larger as the nesting of the stack becomes shallower. In other words, a function with a stack number one less than the stack number for another function becomes the calling number of that function. This area also has a jump feature that allows you to jump to the starting address of the function pointed to by the selected stack frame number in the Source Text, Assemble, or Memory windows. To execute, select the stack frame number, right-click and select either **Source**, **Assemble**, or **Memory**.



- (b) **Stack contents display area** displays the stack contents in the format [file-name#function-name(parameters)]. The sharp (#) character is used as a separator between a file name and a function name.

**(c) Command buttons**

Refresh	Refreshes the data in the window
Close	Closes the Stack Trace window.

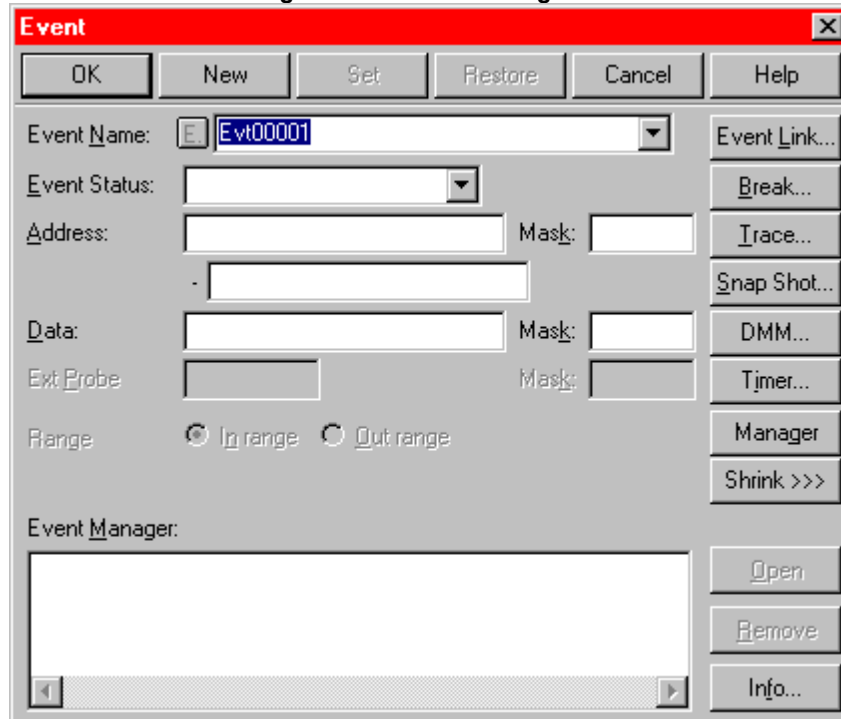


**Caution:** When functions such as *noauto* or *norec* do not move the frame pointer when the `stack` or `-qf` options in the compiler are included for optimization, the stack trace display function is not properly displayed.


**(24) Event Set Dialog Box**

This dialog box allows you to register and display event conditions in the Event Manager. To open the dialog box, select **Event** → **Event...** in the Main window.

Figure 4-41. Event Dialog Box



A maximum of 64 KB fetch events and 256 data values can be registered as event conditions, where the data value can be qualified as address, read, write or read/write values. However, the number of events that can actually be used in a break and tracer is 3 points. The events that can be simultaneously used have 12 points and can be set in multiple event conditions, such as breaks, tracers, and event links.

- (a) **Event name setting area** selects the event name. “\*\*NEW\*\*” is displayed by default. Press  to display the list of event names. Event names can have a maximum of eight characters.



- (b) **Address setting area** specifies address conditions in the format range  $0 \leq \text{address value} \leq 0\text{xffff}$ . There are two types of address conditions: those that set the address value and those that input the mask value.



**i. Address**

Input in the order of  – .

The address condition has 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.

An address condition can also be specified by a symbol, as follows.

Function and variable	<u>_</u> fnc (function or variable name) file#_fnc (for static functions and variables)
SFR	Sfrname (SFR name)
Line number in the source text	file:no (file name and line number)

Function or variable names are specified with an underline character (\_) at the beginning of the symbol name. The sharp (#) character is used as a separator between a file name and a function or variable name. The colon (:) is used as a separator between a file name and a line number.

**ii. Mask**

A mask can be set as an address condition. The default is 0x0000, 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.

- (c) The **Event Status** area specifies the status condition for execution and access events, which can be simultaneously determined.

Event Status:  

**Table 4-23. Contents of Status Condition**

Status	Event Type	Meaning
Run	Execution event	Program execution
R	Access event	Data read
W		Data write
R/W		Data read/write



(d) **Data setting area:** specifies the data condition in the format  $0 \leq \text{Data} \leq 0\text{xff}$ .

There are two types of data conditions: those that set the data value and those that input the data mask value.

i. **Data** specifies the data value, as follows.

Function and variable	<u>_</u> func (function or variable name) file#_func (for static functions and variables)
SFR	Sfrname (SFR name)
Line number in the source text	file:no (file name and line number)

Function or variable names are specified with the underline character (\_) at the beginning and with the sharp (#) character as a separator between a file name and a function or variable name. The colon (: ) is used as a separator between a file name and a line number.

ii. The **Mask** option sets a mask for the data value by means of an OR condition. The default is 0xff. The data condition becomes invalid (matches the condition for any data).

**Example:** When the settings are Data  Mask , the condition is matched when the data value is 0x7F.

When the settings are Data  Mask , the condition is matched for the data values from 0x80 to 0x8F.

When the settings are Data  Mask , the condition is matched when bit 1 is 1.

When the settings are Data  Mask , the condition is matched when bit 1 is 0.

(e) The **External Probe** box is not supported.

(f) **Event Manager**

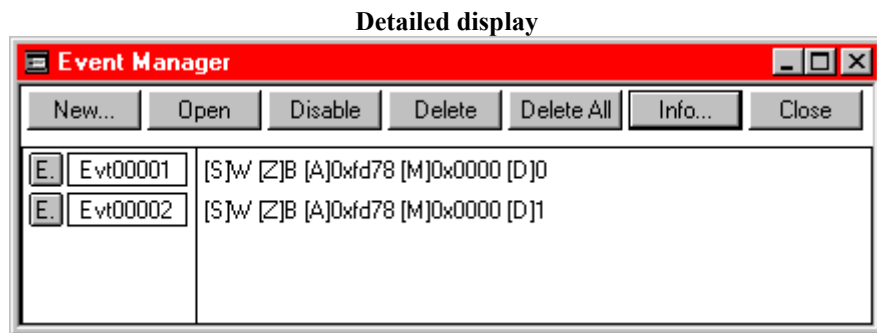
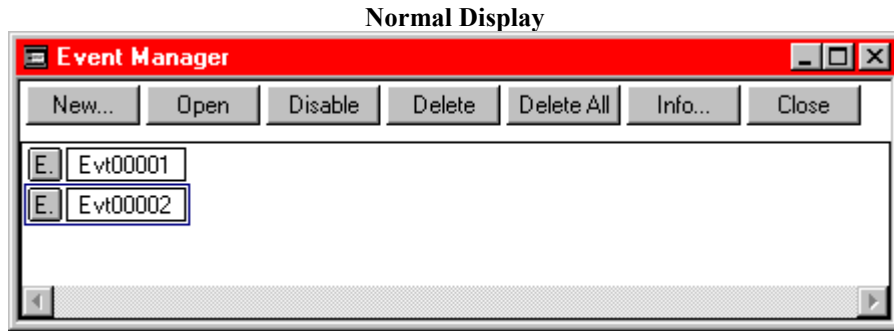
**(g) Command Buttons**

OK	Accepts the settings and closes the window
New	Creates a new event
Set	Enables the event (Event icon turns red)
Restore	Restores the event condition
Cancel	Cancels the Event setting and closes the window
Help	Opens the Help window
Event Link...	Opens the Event Link Setting window
Break...	Opens the Break Setting window
Trace...	Opens the Trace Setting window
Snap Shot...	Not supported
DMM...	Not supported
Timer...	Not supported
Manager	Opens the Event Manager window
Shrink >>>	Displays the Event Manager in the Event Setting window
Open	Opens a previously saved event setting
Remove	Removes the highlighted event
Info...	Displays detailed information regarding the event

(25) Event Manager

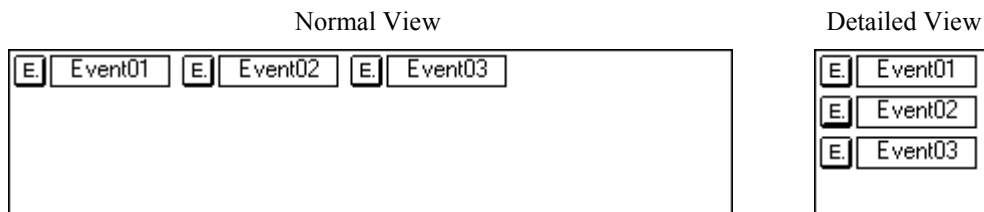
The Event Manager allows you to display and delete various events. In other words, an event condition registered in the Event Set dialog box or Event Link dialog box can be assigned to a break or a trace. To open, select **Event** → **Event Manager...** from the Main window.

Figure 4-42. Event Manager



- i. **Event display area** displays icons of registered events.

Figure 4-43. Event Display Area







An icon is a mark indicating the event type and event name.









Table 4-24. Mark List

Mark	Description
------	-------------

	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.

**Table 4-25. Letter Color in Mark List**

Letter Color	Applicable Marks	Description
Red	 , 	Indicates that events and event link conditions are 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 also has two functions.

**i. Jump function**

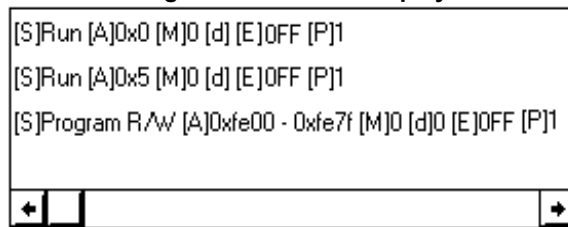
This function jumps to the Source Text window, Assemble window, or Memory window when the address condition of the selected icon is used as the jump pointer. The Jump Destination window is displayed from the jump pointer. To execute, select the icon. Right-click on the icon and select either Source, Assemble, or Memory to jump to the respective window.

**ii. Delete function**

This function deletes the event registration and setting of the selected icon. If event condition **E** and event link condition **L** are deleted, they cannot be used in events **B** and **T**. To use them in other events, first delete the events being used. Select the icon → **Delete**.

**iii. Event display**

**Figure 4-44. Event Display**





The details for each event are displayed in Detailed View mode by right-clicking and selecting detail from the pop-up window. The contents are displayed in order of status condition, address condition, address mask condition, and data condition using the various keys as separators.

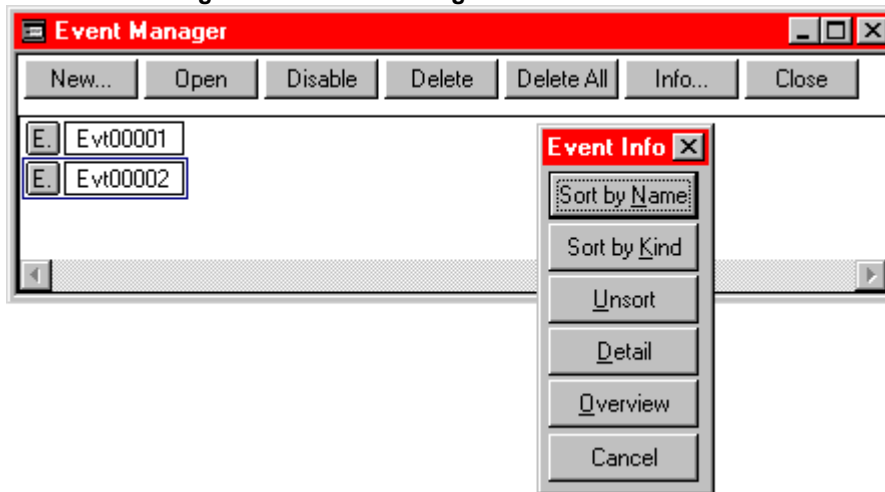
**Table 4-26. Key Data Correspondence**

For Event Conditions	
Key Data	Description
[S]	Status condition
[A]	Address condition
[M]	Address mask condition
[d]	Data condition

For Event Link Conditions	
Key data	Description
[P1] – [P4]	nth event link condition

For Break and Trace Conditions	
Key data	Description
[B]	Break condition
[Q]	Qualify trace condition

**Figure 4-45. Event Manager—Event Info Window**



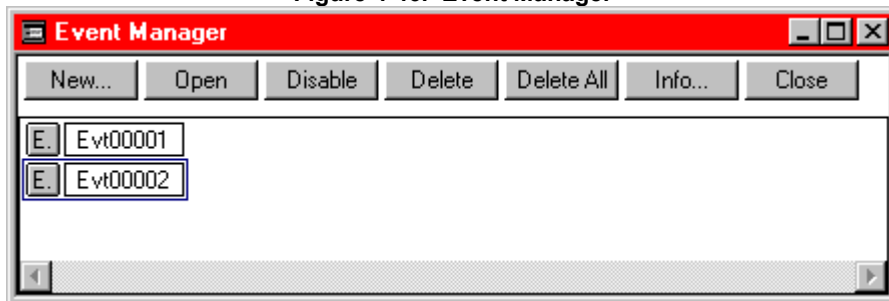
iv. **Command buttons**



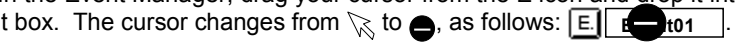


New	Opens the Event Setting window to create a new event
Open	Opens the Setting File Selection dialog box and loads the event setting file (the event register/setting contents before loading are lost)
Disable	Disables the highlighted event or event-based function
Delete	Deletes the highlighted event or event-based function from the Event Manager
Delete All	Deletes all the events and event-based functions from the Event Manager
Info...	Displays the Event Info window
Close	Closes the Event Manager
Sort by Name	Sorts events by name
Sort by Kind	Sorts events by type (E=events, B=breaks, and so forth)
Unsort	Unsorts registered events in the Event Manager
Detail	Displays the detailed information of the events
Overview	Hides the details of the events
Cancel	Cancels the settings

v. **Event Setting**

- i. Select **Event → Event...** to open the Event Set dialog box.
- ii. Set the conditions for Event01 and Event02.
- iii. Select **Event → EventManager....** to see the events registered in the Event Manager.

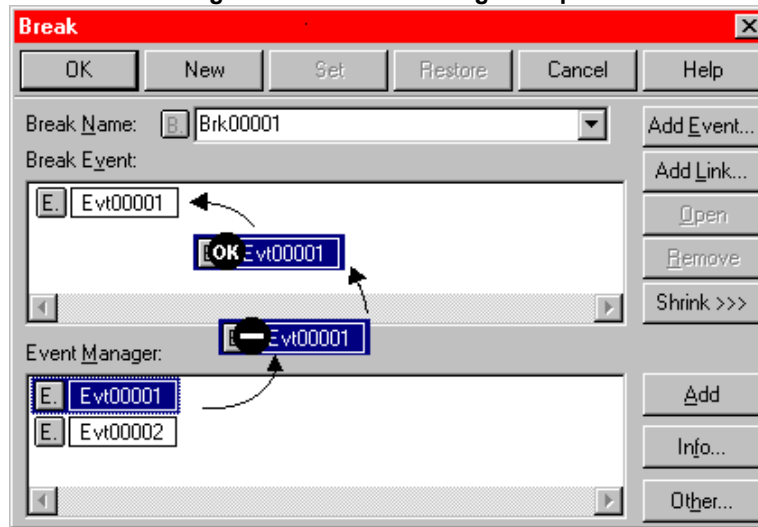
Figure 4-46. Event Manager



- iv. Open the Trace, Break, or Event Link dialog box.
- v. In the Event Manager, drag your cursor from the E icon and drop it into the corresponding event box. The cursor changes from  to , as follows: .
- vi. Drag the cursor from the event box and drop it into the dialog box (Trace, Break, Timer, or Event Link) to copy the icon. The cursor changes from  to .

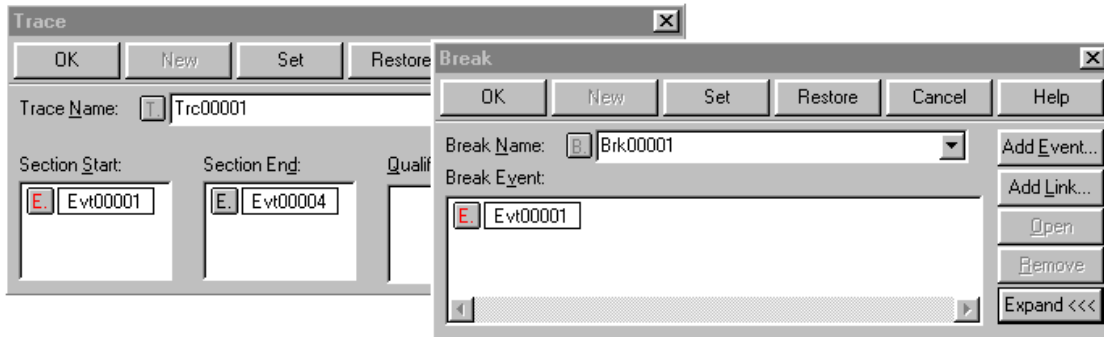
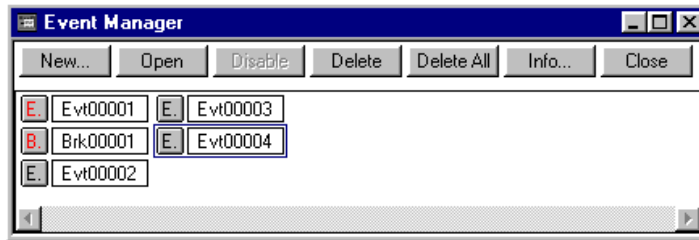
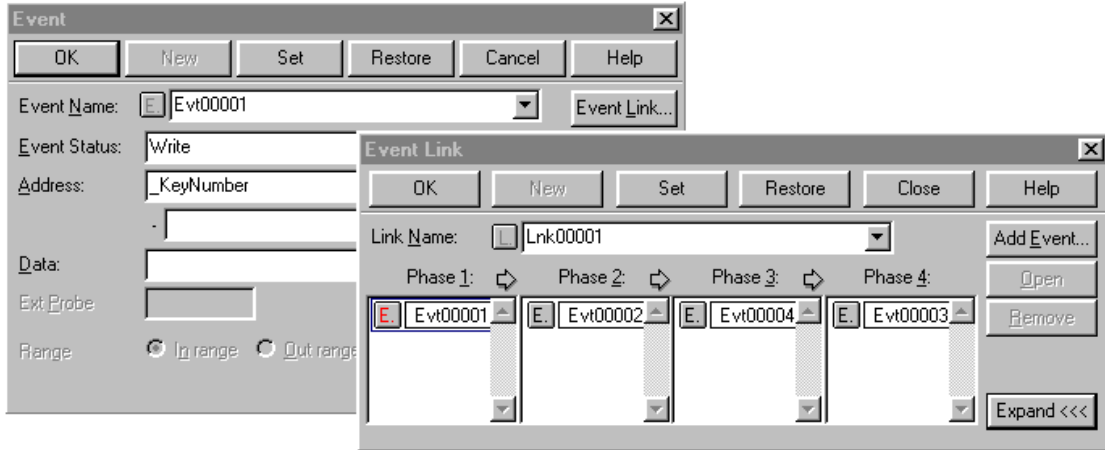
- vii. Enter the break event name and click the **Set** button to register the event.

Figure 4-57. Event Setting Example



- (h) Events Managed by the Event Manager

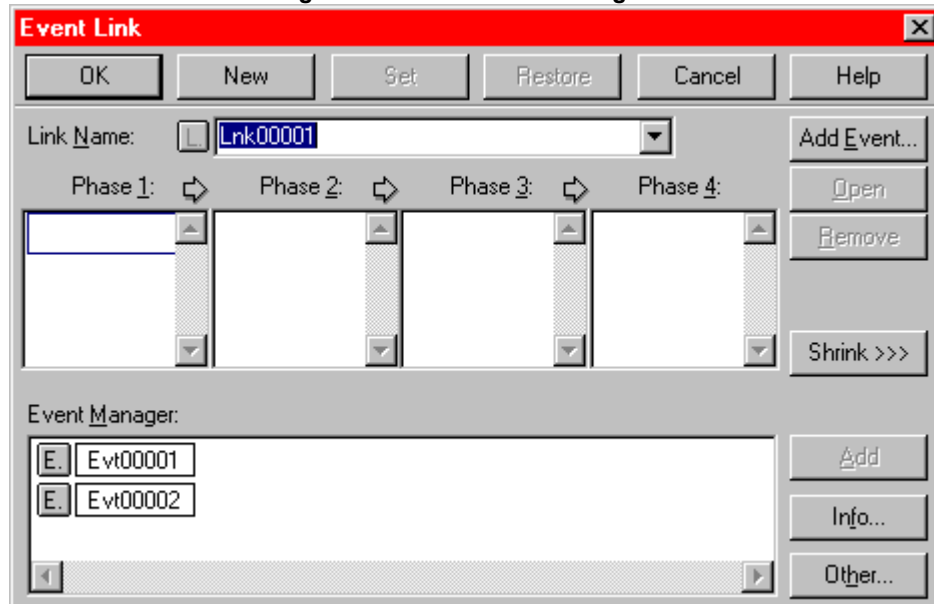
Figure 4-48. Event-Related Images



**(26) Event Link Dialog Box**

Event link conditions are registered in this dialog box and then automatically displayed in the Event Manager. From the Main window, select **Event** → **EventLink...** to open the Event Link dialog box.

Figure 4-59. Event Link Dialog Box

**(a) Functions**

Event link conditions are set with only the execution event conditions registered in the Event Manager. An event link condition can have up to four phases, but only one can be used at a time. When the event link condition is used, break and trace events are ignored.

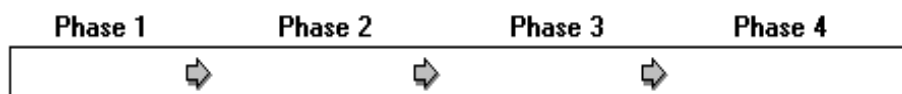
Events are generated during program execution in the order specified. However, if the conditions specified in the last phase are satisfied before the conditions specified in phases 1 to 3, then the satisfied event conditions are initialized and the first event condition becomes the detection target.

**(b) Event Link**

The **Event Link** box allows you to select and set an event link. Press the down scroll arrow to display the list of event link names (maximum eight characters). **\*\*NEW\*\*** is the default.







**(c) Link Conditions**

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.





**(d) Setting Event Link Conditions**

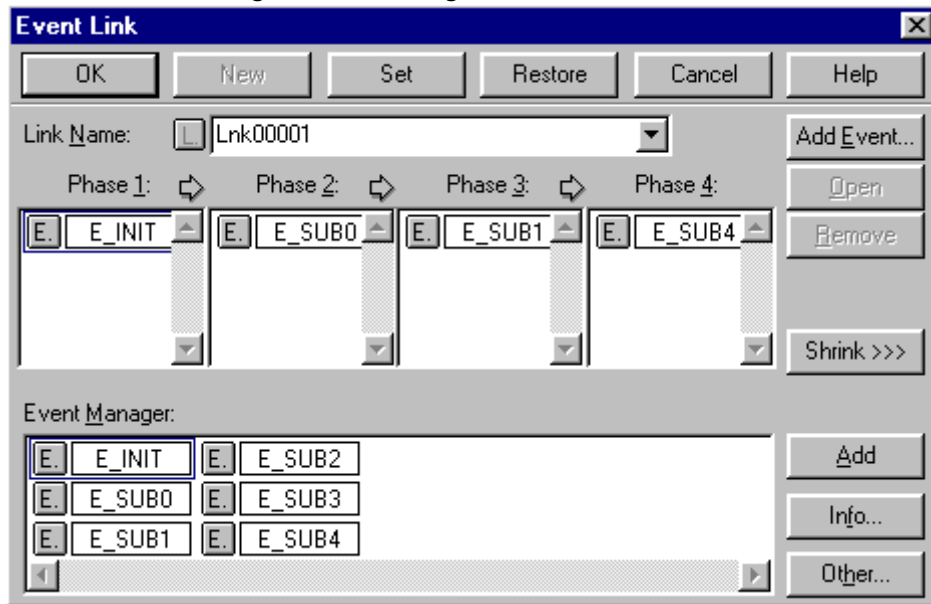
- i. Select **Event** → **EventSet...** to open the Event Set dialog box.
- ii. Create the event conditions for E\_INIT, E\_SUB0, E\_SUB1, E\_SUB2, and E\_SUB4.
- iii. Select **Event** → **EventManager...** to open the Event Manager.
- iv. Select **Event** → **EventLinkSet...** to open the Event Set dialog box.
- v. In the Event Manager, drag your cursor from the E icon and drop it into the corresponding event box. The cursor changes from  to , as follows:   t01.
- vi. Drag the cursor from the event box and drop it into the Event Link dialog box. The cursor changes from  to .
- vii. Repeat steps vi and vii to register the settings listed in Table 4-27.

**Table 4-27. Settings in Event Link Dialog Box**

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

- viii. Enter the name of the event link (E\_LINK).
- ix. Click **Set** to register the condition for E\_LINK in the Event Manager.

Figure 4-50. Setting Event Link Conditions

**(e) Example**

This example shows how to set event link conditions for the program shown in Figure 4-49, which has the following structure.

## Main processing

1. Initialization process (INIT)
2. Subprogram 0 (SUB0)
3. Condition decision
  - (a) Subprogram 1 (SUB1)
  - (b) Subprogram 2 (SUB2)
4. Subprogram 4 (SUB4)

The execution route of this program follows routes (1) and (2) shown in Figure 4-52. When an event is generated, the event condition shown in Table 4-28 is set. Setting the event link condition shown in Figure 4-51 causes the desired events to be generated.

Table 4-17. Setting in Event Link Dialog Box

Setting Position	Set Event
Phase 1	E_INIT
Phase 2	E_SUB0
Phase 3	E_SUB1, E_SUB2
Phase 4	E_SUB4



Figure 4-51. Event Link Dialog Box

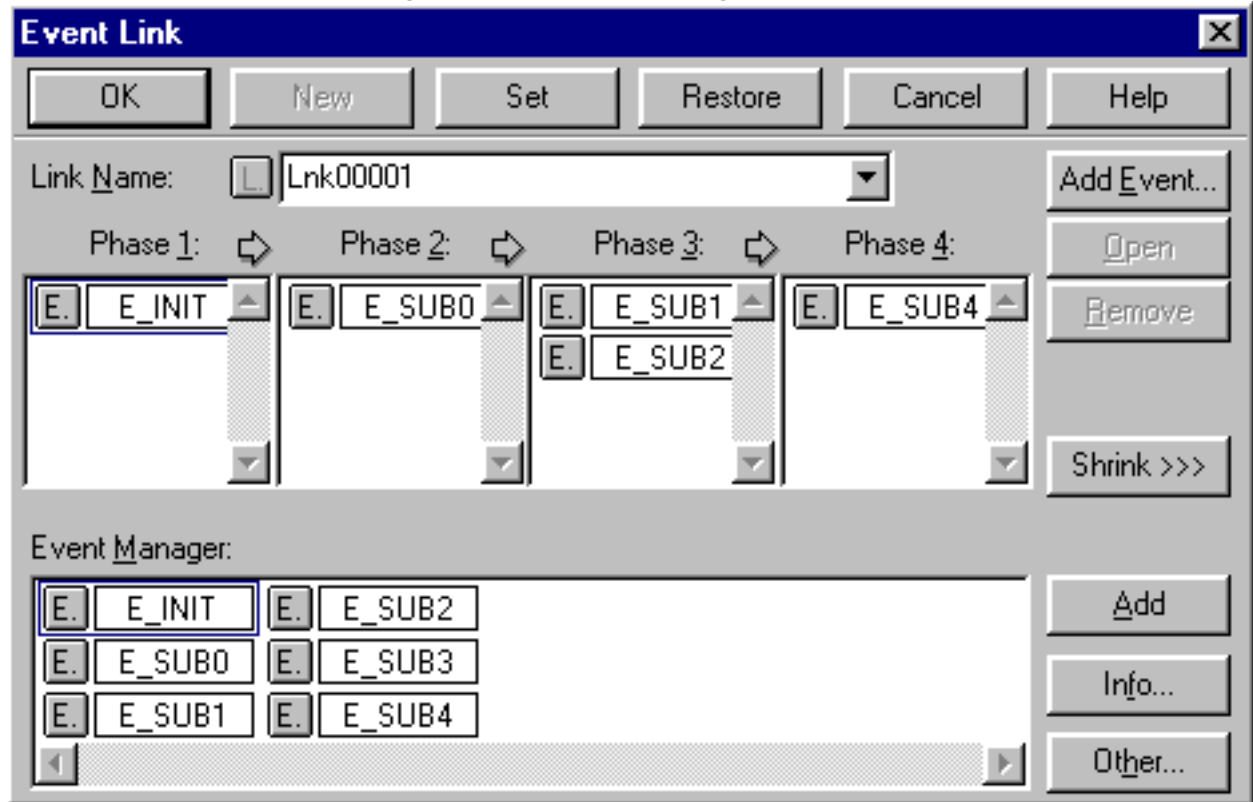
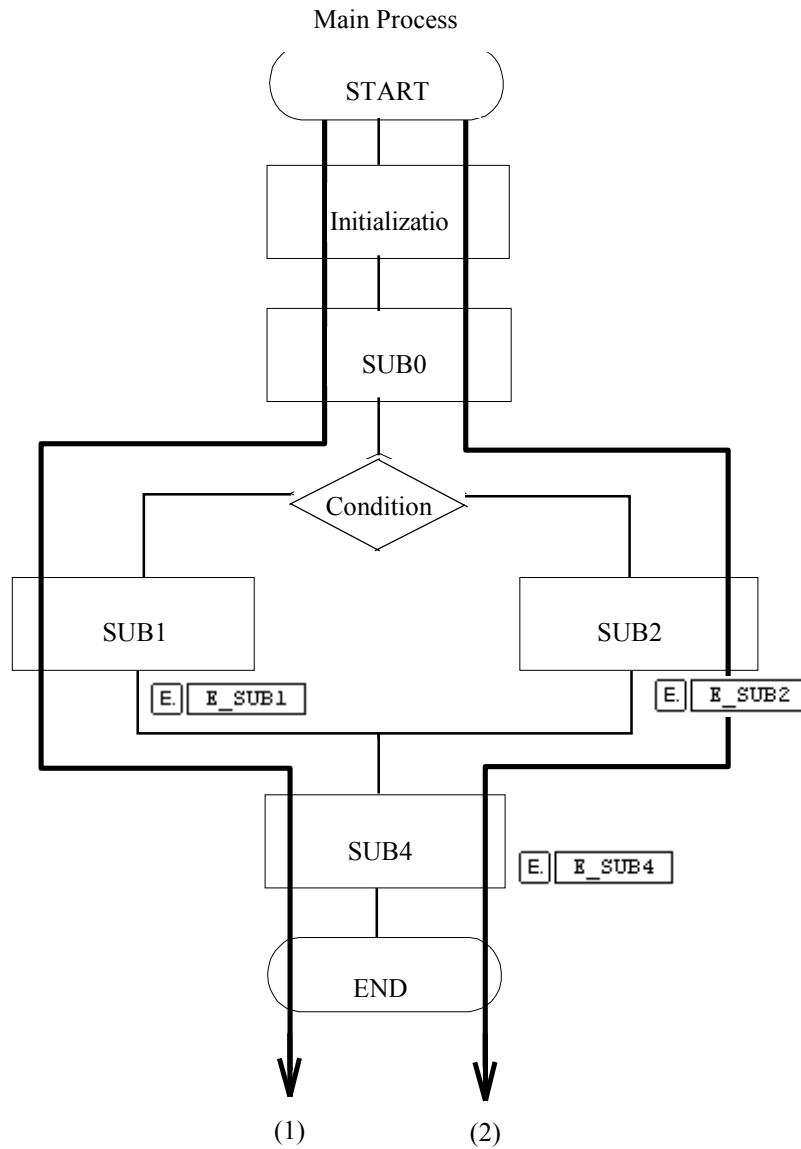


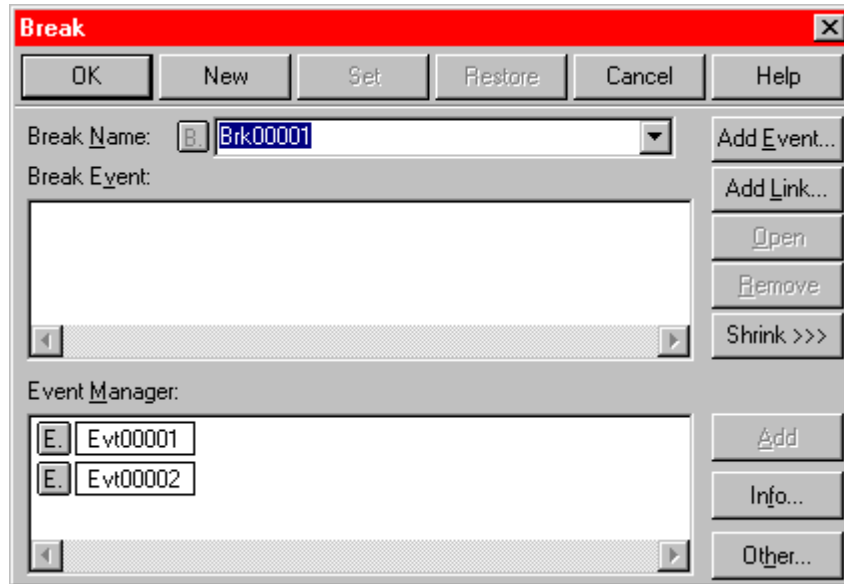
Figure 4-52. Example Using Event Link Conditions

**(27) Break Dialog Box**

Break event conditions are set, registered, and displayed from this dialog box. In the Main window, select **Event** → **Break...** or in the Event Manager, select **New** → **Break...**

Break event conditions are registered using the event and event link conditions registered in the Event Manager. A maximum of 64K break event conditions can be registered, but only ten can be used simultaneously.

Figure 4-53. Break Dialog Box



- (a) The **Break Name** box allows you to select a break event name using a maximum of eight characters (“NEW” is the default). Press the down arrow to select from the list.







- (b) The **break condition** box is where you set a break event condition by dragging the event icon and dropping it into an event name. A maximum of 12 conditions can be set.



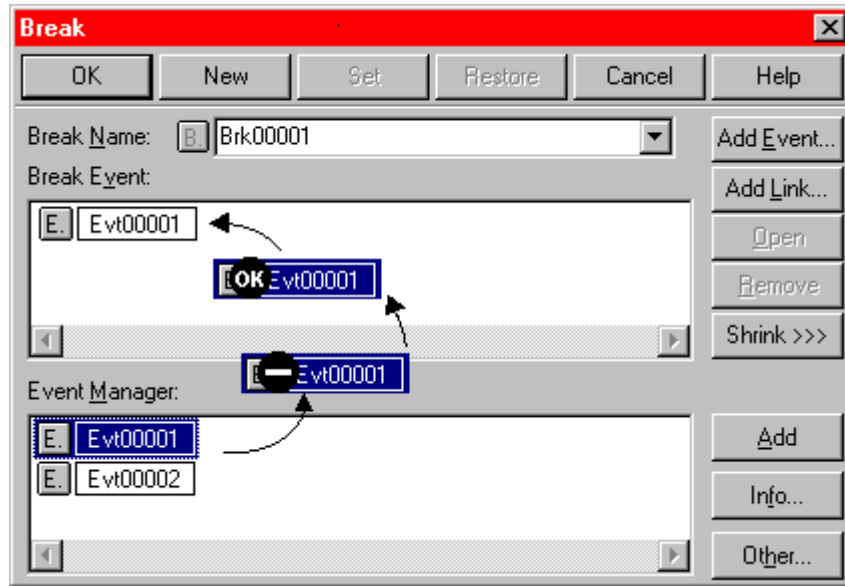
**(c) Command buttons**

OK	Registers the break condition in the Event Manager
New	Creates a new break condition
Set	Enables the break condition, whereby break events are generated
Restore	Restores the break condition
Cancel	Disables the break condition, whereby break events are not generated
Help	Opens the Help window
Add Event...	Opens the Event window to define new events
Add Link...	Opens the Add Link window to link events
Open	Opens the setting window for the highlighted event icon
Remove	Removes the highlighted event
Shrink >>>	Displays the contents of the Event Manager window at the bottom of the Break window

**(d) Example**


- i. Open the Event Set dialog box. Select **Event** → **Event...**
- ii. Create event conditions for Evt00001 and Evt00002.
- iii. Open the Break dialog box. Select **Event** → **Break...**
- iv. Drag the event icon from the Event Manager to the Break window (the cursor changes from  to .
- v. Drag the cursor and drop it into Break dialog box to copy the event (the cursor changes from  to .
- vi. Enter the break event name: Brk00001.
- vii. Click **Set** to register the break in the Event Manager. (The **Set** button becomes the **Enable** button.).
- viii. Click **Enable** to enable the break event condition (the **B** marker changes from black to red).

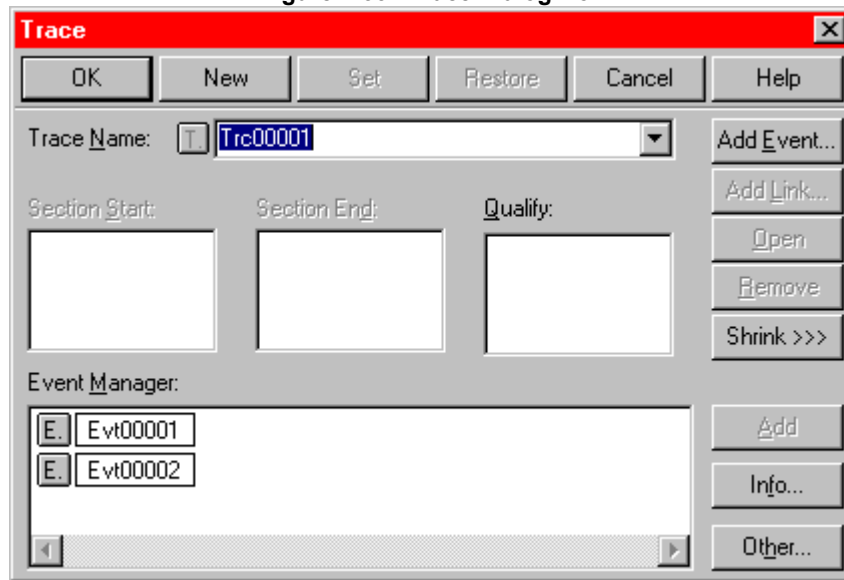
**Figure 4-54. Setting Break Event Conditions**



**(28) Trace Dialog Box**

Trace event conditions are registered, set, and displayed in this dialog box and then automatically registered in the Event Manager. A maximum of 64K trace event conditions can be registered, but only one is enabled.

From the Main window, select **Event** → **Trace...** or click  or from the Event Manager, select **New** → **Trace...**. To operate the tracer in accordance with the trace event conditions, always select **Run** → **Cond. Trace ON** from the Main window.

**Figure 4-55. Trace Dialog Box**

- (a) The **Trace Name** box allows you to select a break event name using a maximum of eight characters (“NEW” is the default). Press the down arrow to select from the list.



There are three trace modes: All Trace, Qualify Trace, and Sectional Trace.

**Table 4-29. Trace Modes**

Mode	Description
All Trace	Traces all of the causes
Qualify Trace	Traces only the locations with matched event conditions
Sectional Trace	Traces between specified event conditions

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.

**Table 4-30. Trace Mode Settings**

Mode	Execute Setting in Main Window	Trace Mode Setting	Delay Conditions
All Trace	Uncond. Trace ON	–	None
Qualify Trace	Cond. Trace ON	Qualify	Yes
Sectional Trace	Cond. Trace ON	Start and End Sections	Yes

- (b) The **Qualify** box sets event conditions for a qualify trace. To set an event condition, drag and drop the event icons in the Event Manager. Only access events can be set in the Qualify box.



- (c) **Command buttons**

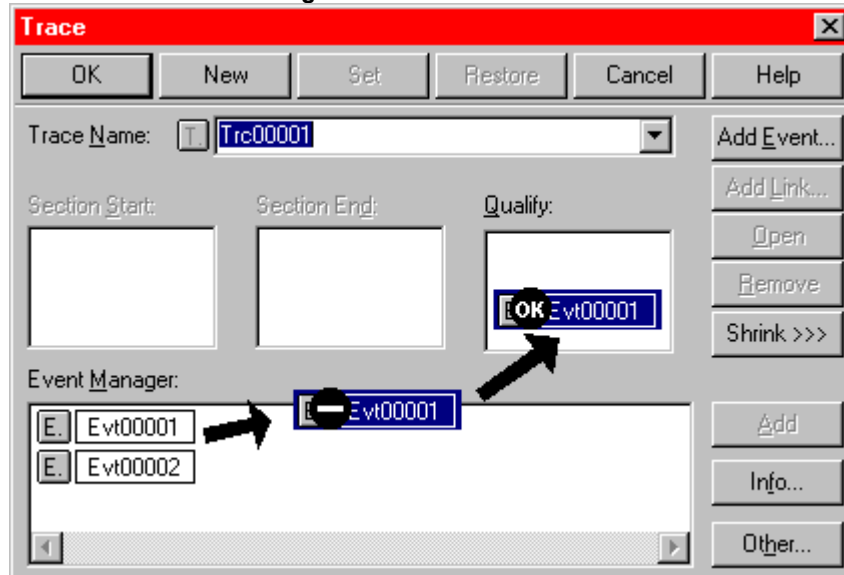
OK	Registers the trace to the Event Manager
New	Creates new trace
Set	Enables the trace conditions (the T marker changes to red)
Restore	Restores the trace condition
Cancel	Disables the trace condition (the T marker changes to black)
Help	Opens the Help window
Add Event...	Opens the Event window to define new events
Add Link...	Opens the Add Link window to link events
Open	Opens the setting window for the highlighted event icon
Remove	Removes the highlighted event
Shrink >>>	Displays the contents of the Event Manager at the bottom of the Break window

- (d) **Example**

- i. Select the **Run → Cond. Trace ON** in the menu bar in the main window.
- ii. Open the event set dialog box. Select **Event → Event...**
- iii. Enter event condition in the Event Set dialog box for two events named Evt00001 and Evt00002.
- iv. Open the trace dialog box. Select **Event → Trace...**
- v. Drag the event icon from the Event Manager to the Trace window.


- vi. Enter trace event name, for example *TRACE*.
- vii. Press **Set** to register it in the Event Manager. (The **Set** button becomes the **Enable** button.)
- viii. Press **Enable** to enable it.

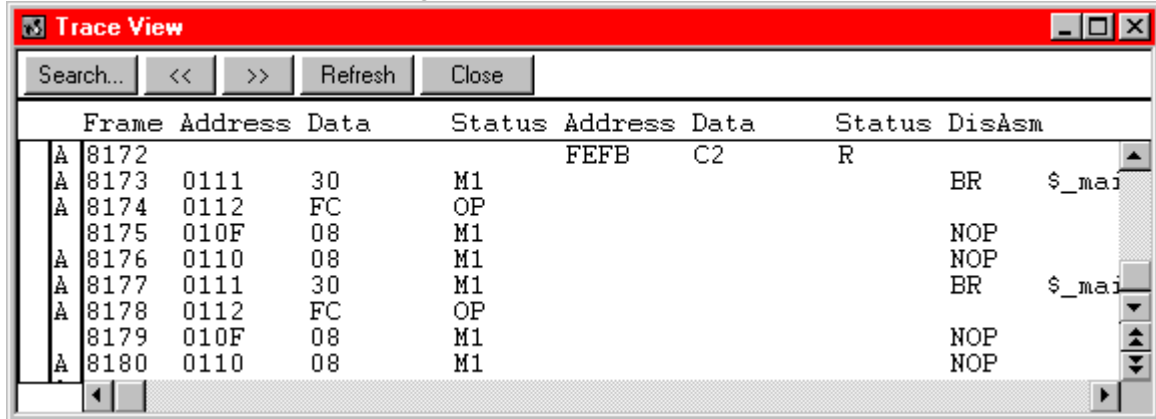
Figure 4-56. Trace Window





**(29) Trace View Window**

The Trace View window displays the trace result. From the Main window, select **Browse → Trace** or click .

**Figure 4-57. Trace View Window**

The tracer has a capacity of 65,535 frames and a ring structure. If more than 65,535 frames of data are written, the oldest is overwritten. The oldest data is frame 0, and the frame numbers are displayed in order. During pauses in program execution, the block data is written to the tracer and displayed as one horizontal line in each display area.

**Table 4-20. Block Data Write**

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

**Table 4-21. Block Data**

Normal Break	
Step Break	Step break
Event Break	Event break
Fail-Safe Break	
Fetch Guard	Fetch guard break
Write Protect	Write protect break
SFR Illegal	SFR illegal access break
Stack Overflow	Stack guard break; stack overflow break
Unspecified Illegal	Other breaks

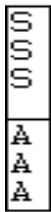
- (a) The **Point Mark** box displays the event settings. If the execution or access fetch event is set at the corresponding trace address, this box displays the mark corresponding to the event type.



**Table 4-16. Event Mark Display**

Mark	Description
E	Event condition is set
L	Last phase of an event link is set
B	Break event is set
T	Trace event is set

- (b) The **Trace Mode** box displays the type of trace mode, where A=All Trace, Q=Qualify Trace, and S=Step Execution Trace.



- (c) The **Trace View** window displays the trace results and is used for the jump and window synchronize functions.

**Figure 4-58. Trace View Window**

Frame	Address	Data	Status	Address	Data	Status	DisAsm
8172				FEFB	C2	R	
8173	0111	30	M1				BR \$_mai
8174	0112	FC	OP				
8175	010F	08	M1				NOP
8176	0110	08	M1				NOP
8177	0111	30	M1				BR \$_mai
8178	0112	FC	OP				
8179	010F	08	M1				NOP
8180	0110	08	M1				NOP

- i. The Trace window can be synchronized with the Source, Assembly, or Memory windows so that the code displayed in each window coincides with the Trace window. To initiate the window synchronize function from the Trace window, right-click and select **Window Synchronize** <any window> on the pop-up window.

**Table 4-17. Connection Window**

Items in <u>Window Connect</u>	Connect Window
<u>S</u> ourceText	SourceText window
<u>A</u> ssemble	Disassemble window
<u>M</u> emory	Memory window

- ii. Highlight the trace result display area of the Trace View window.
- iii. With the address of the trace result selected in step ii as the pointer, highlight the display areas of each window selected in step i.

The window synchronize function differs from the jump function, because the area selected in the Trace View window moves and the result is reflected in each window of the connection target.

**(d) Trace frame number display (Frame)**

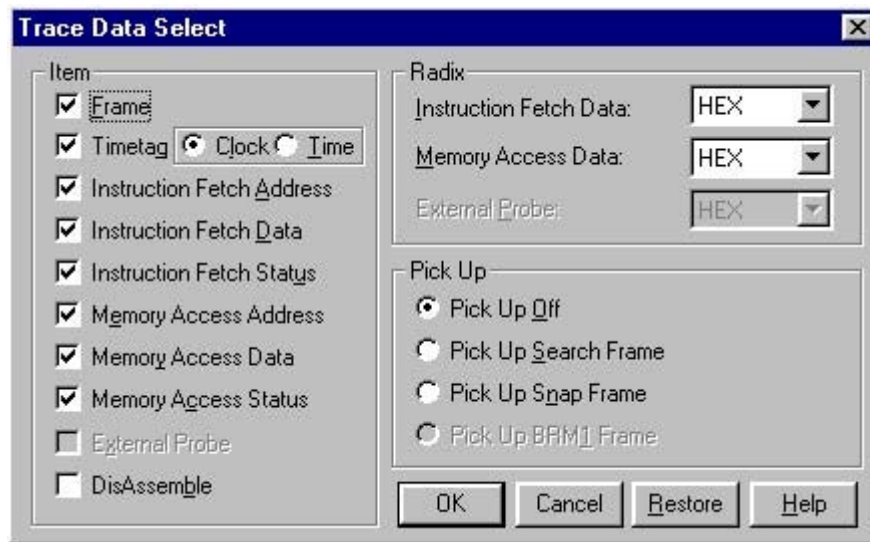
Range:  $0 \leq \text{Trace frame number} \leq 65,535$

**(e) Fetch access display (Address Data Status)**

Addr	Fetch address display
Data	Fetch data display

The fetch access display can be selected from the Trace Data Select dialog box. From the Main window with the Trace window active, select the **View** → **Select...**

**Figure 4-59. Trace Data Select Dialog Box**



(f) **Data access result display** can be selected from the Trace Data Select dialog box (Figure 4-59).

**Table 4-25. Data Access Result Display**

Status	Display Contents
RW	Data read/write by user program
RD	Data read by user program
WD	Data write by user program

Addr	Address display
Data	Data display

(g) **Mnemonic display (DisAsm)** displays the disassemble result and the status for BRM1 and M1.

(30) Register Window


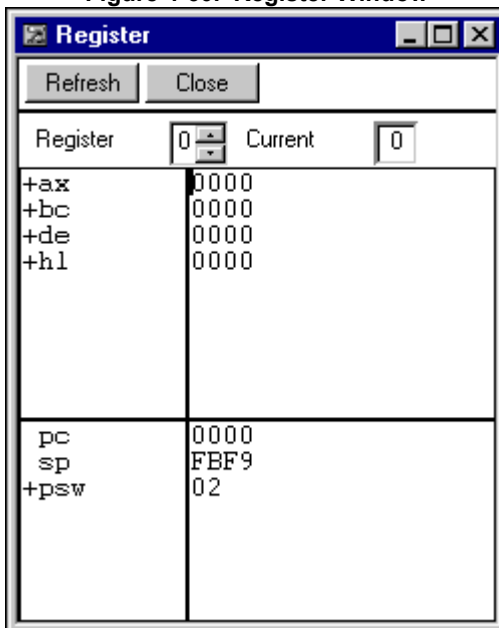
General-purpose and control register values are displayed and changed in the Register window, which can be opened from the Main window by selecting **Browse** → **Register** or by clicking .

Figure 4-60. Register Window



(a) The **register bank setting** area displays and sets the bank number of the general-purpose registers.

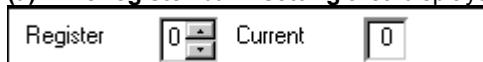

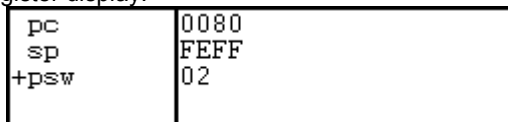


Table 4-26. Register Bank Setting Area

Item	Description
Register Bank:	Displays and sets the register bank displayed in the general-purpose register display area. Changing the bank number is performed using the  button.
Current Bank:	Displays the register bank number currently set to the target (current bank).

(b) The **control register** box displays the control register values. To change a value, highlight and type over it. Press **ENTER** to execute the change. Double-clicking on the register with a “t” expands the register display.



(c) The **general-purpose register** box displays and changes general-purpose register values. To change a value, highlight and type over it. Press to execute the change. Right-click the display area and select between Absolute name display and Function name display and the notation for display.

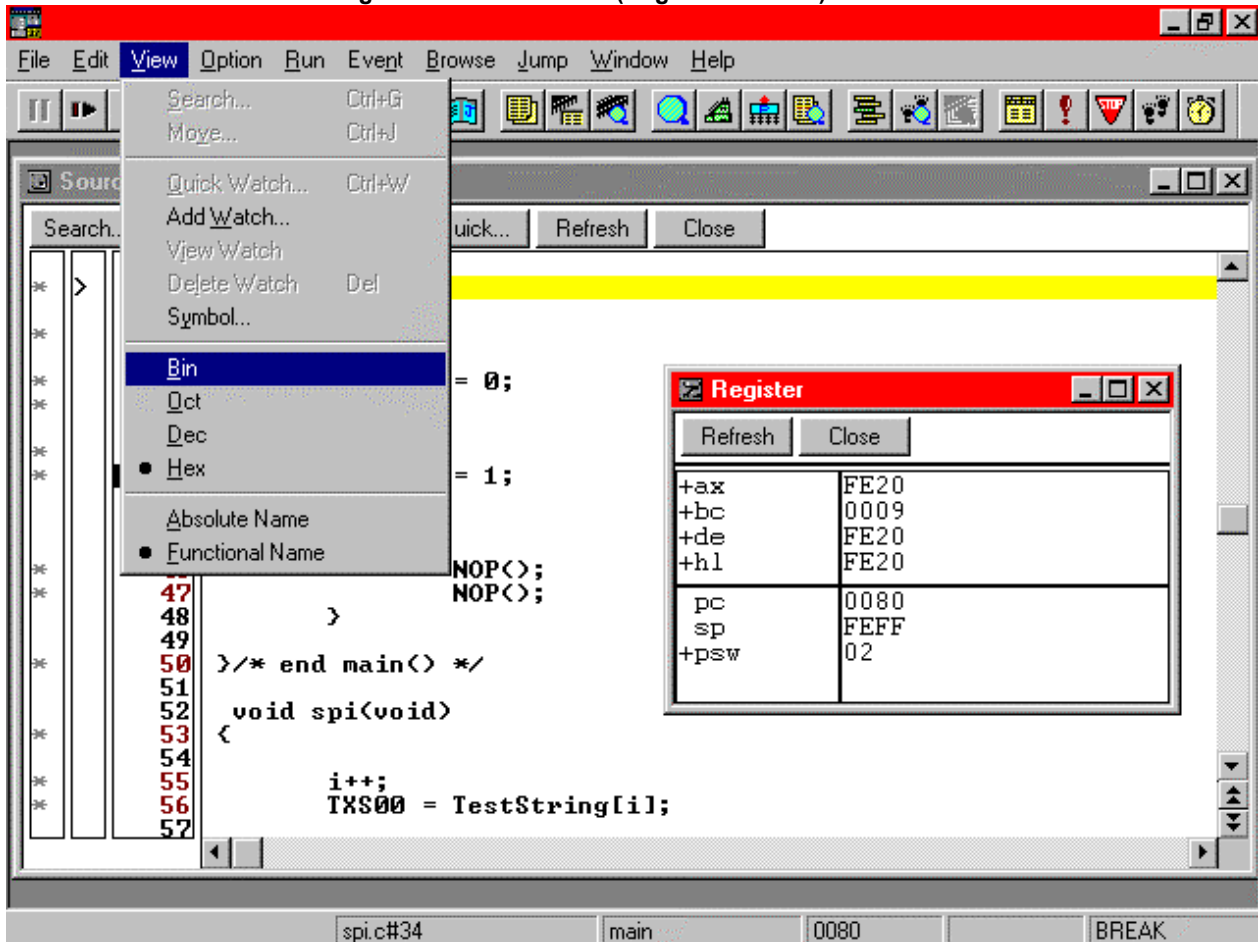
+ax	FE20
+bc	0009
+de	FE20
+hl	FE20

Figure 4-60. General-Purpose Register Display

Function Name and Register Pair		Function Name and Register	
+ax	FE20	-ax	000200
+bc	0009	x	200
+de	FE20	a	000
+hl	FE20	-bc	000000
		c	000
		b	000
		-de	000000
		e	000
		d	000
		-hl	000063
		l	063
		h	000
Absolute Name and Register Pair		Absolute Name and Register	
+rp0	000200	-rp0	000200
+rp1	000000	r0	200
+rp2	000000	r1	000
+rp3	000063	-rp1	000000
		r2	000
		r3	000
		-rp2	000000
		r4	000
		r5	000
		-rp3	000063
		r6	063
		r7	000




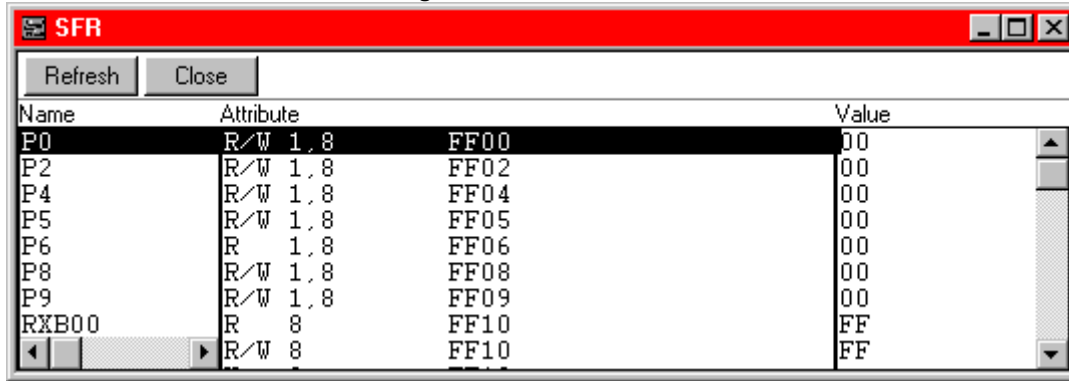
Figure 4-61. View Menu (Register Window)



<b>Absolute Name</b>	Displays the register names as absolute names
<b>Functional Name</b>	Displays the register names as functional names
<b>Bin</b>	Displays values in binary format
<b>Oct</b>	Displays values in octal format
<b>Dec</b>	Displays values in decimal format
<b>Hex</b>	Displays values in hexadecimal format

**(31) SFR Window**

SFR values are displayed and changed in the SFR window, which can be opened from the Main window by selecting **Browse** → **Sfr** or by clicking .

**Figure 4-78. SFR Window**

A read-only SFR is displayed in gray, and ones that cannot be changed are highlighted. The display format and reading method of the SFR display can be specified in **View** menu.

(d) The **Name** box displays the SFR names.

```
P6
P7
POL
POH
CR00
CR01
CR10W
```

(e) The **Attribute** box displays the SFR read/write attributes, access type, and address. The attribute display can be selected from the **View** menu.

```
R/W 1,8 OFFF06
R/W 1,8 OFFF07
R/W 1,8 OFFF0E
R/W 1,8 OFFF0F
R/W 16 OFFF10
R/W 16 OFFF12
R/W 16 OFFF14
```

**Table 4-32. SFR Attributes**

Attribute	Description
R	Read-only (displayed in gray)
W	Write-only
R/W	Read/write

Table 4-33. SFR Access Types

Access Type	Description
1	Bit-accessible
8	Byte-accessible
16	Word-accessible

- (f) The **SFR contents** box displays the SFT values based on the SFR attribute: write-only is displayed as “—”. An SFR whose value changes when read in real time is displayed as “\*\*\*”. To change a value, highlight and type over it. The change is displayed in red. Press **Enter** to execute the change, which will then appear in black.

```
00
00
00
00
0000
0000
FF00
```

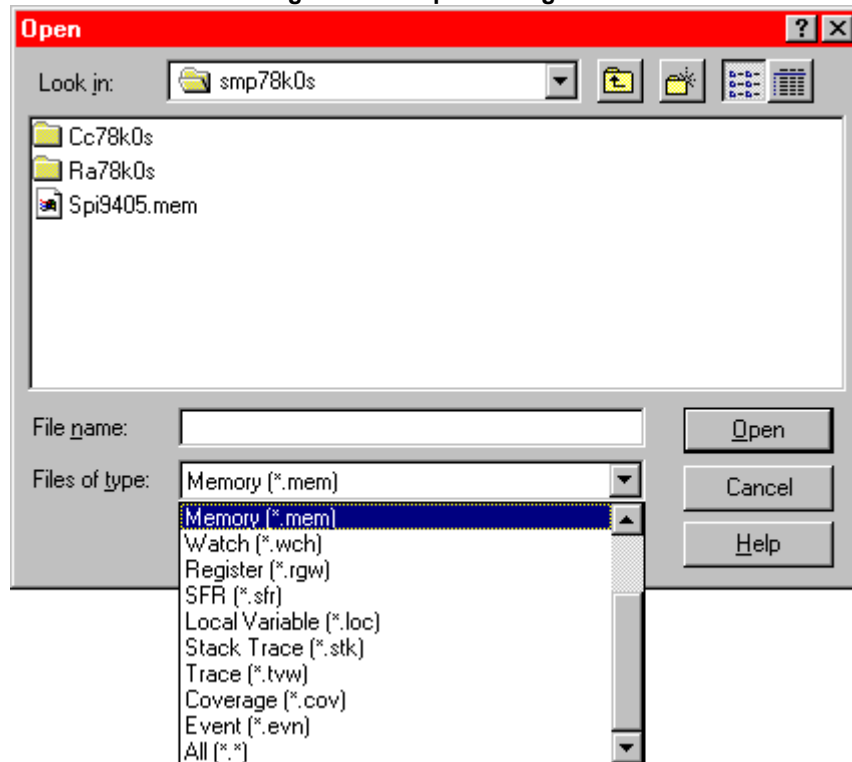
- (g) Command buttons

Refresh	Refreshes the displayed window
Close	Closes the SFR Window

**(32) Open Dialog Box**

The view file corresponding to the current window when this dialog box was opened is read and the reference window is opened. The Open dialog box can be opened in two ways. When the window to be referenced is a Local Variable, Disassemble, Memory, Stack Trace, SFR, or Trace View window,

- (a) The window you want to reference becomes the current window.
- (b) Select **File** → **Open**.

**Figure 4-63. Open Dialog Box**

- (a) The **File name** box is where you specify the file to be loaded. Click the file name to select it. Double-click the file name or click **OPEN** to open the file.

File name:

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

- (b) The **Look in** box specifies the folder containing the file to be loaded. Click the down arrow to view the list. Double-click a folder name to display its contents.



- (c) **Command buttons**

Open	Opens the file selected
Cancel	Closes the dialog box
Help	Opens the Help window

### (33) Save As Dialog Box

The Save As dialog box is used to save the contents of the current window. When the window to be saved is a Local Variable, Disassemble, Memory, Stack Trace, SFR, or Trace View window,

1. The window to be saved becomes the current window.
2. Select **File** → **Save As...** in the Main window.

**Figure 4-81. Save As Dialog Box**

*When the window to be saved is a Local Variable, Disassemble, Variable, Stack Trace, SFR or Register window, or a window in the hold state:*

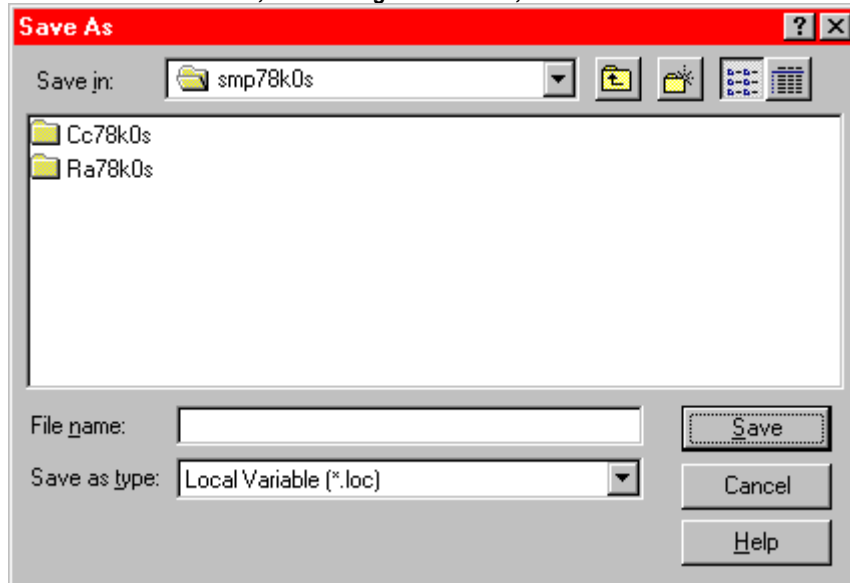
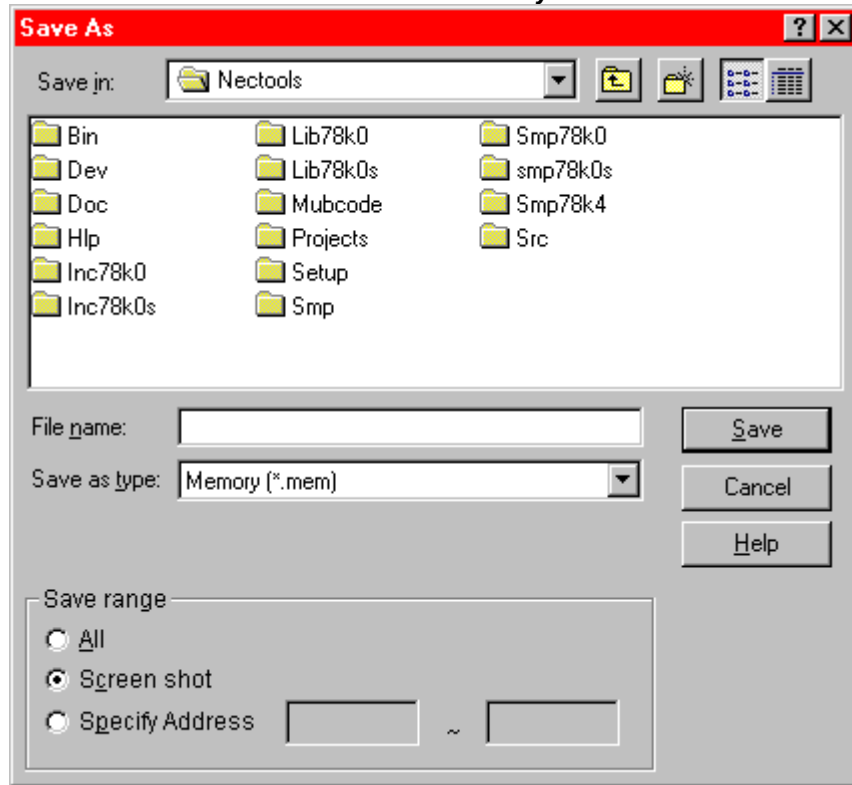
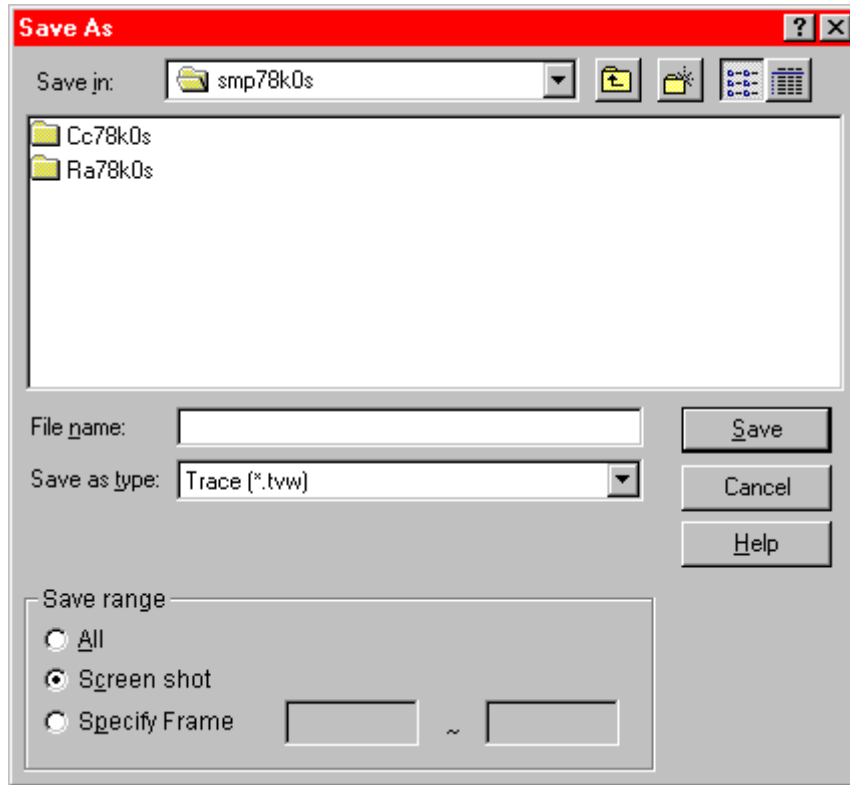


Figure 4-81. Save As Dialog Box (continued)

*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:*





- (a) The **File name:** box is where you specify the file name to be saved. Select and highlight a file name. Double-click the file name or click **SAVE** to save the file. The default extensions are listed in the following table.

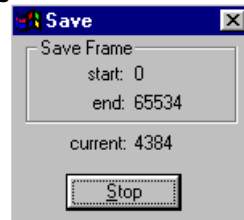
Window	Default Extension
Variable window	VAR
Local variable window	LOC
Disassemble window	DIS
Memory window	MEM
Register window	REG
Stack trace window	STK
SFR window	SFR
Trace view window	TVW
Event manager	EVN

- (b) The **Save in:** box specifies the folder in which to save the file. Double-click the folder to display its contents.
- (c) The **Save range** box is displayed when the window to be saved is a Memory or a Trace View window.
- i. When the current window is a Memory window, this box specifies an address range.

- ii. When the current window is a Trace View window, this box specifies the range to be saved, where the specification range is  $0 \leq \text{frame number} \leq 65,535$ .

The Save Message box appears if a range above 100 frames is specified. To abort the save, click **STOP**.

**Figure 4-82. Save Message**

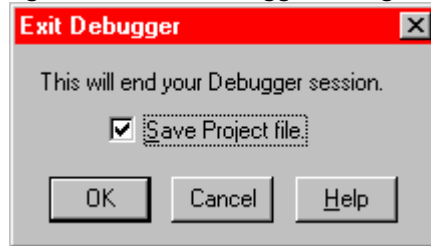


**(d) Command buttons**

Save	Saves the selected window to the specified name
Cancel	Closes the dialog box
Help	Opens the Help window

**(34) Exit Debugger Dialog Box**

The Exit Debugger dialog box allows you to save the debugging environment in a project file and exit the debugger. From the Main window, select **File** → **Exit**.

**Figure 4-66. Exit Debugger Dialog Box**

Clicking **OK** in this dialog box when the Save Project file option is enabled opens the Save dialog box where you can save the current debugging environment in a project file. After the save operation, the project windows closed and exits the program.

If the Save Project file is not selected (default), clicking **OK** closes the windows and exits the program.

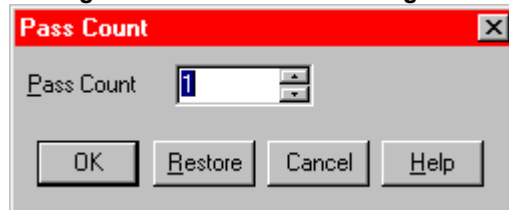
**Command buttons**

OK	If the Save Project file option is selected, OK opens the Save dialog box, closes the windows and exits the program If the Save Project file is not selected, <b>OK</b> closes the windows and exits the program
Cancel	Cancels the operation
Help	Opens the Help Window

### (35) Pass Count Dialog Box

The Pass Count dialog box allows you to set the Pass Count. From the Main window, select **Event → Pass Count...**

Figure 4-88. Pass Count Dialog Box







The Pass Count is used by the debugger to stop execution of the program and tracer after a specified number of events have occurred. The Pass Count may be set to values from 1 to 255.

The Pass Count is set to 1 as a default. In this case, the debugger will stop on each event which would normally cause an execution break. If the Pass Count is set to a value other than one, the debugger will continue execution until the specified number of events have occurred. The value set in the Pass Count remains until it is changed again with the Pass Count Dialog Box.

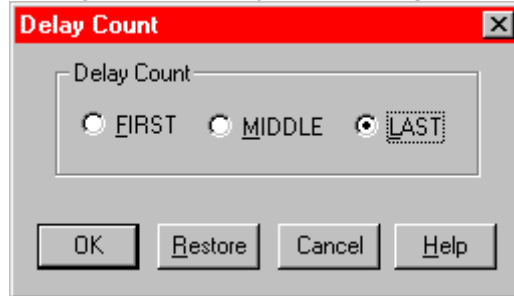
Note that setting the Pass Count to a value other than one will cause some program breakpoints to be skipped, resulting in unexpected operation. If the debugger is not stopping where you have set a breakpoint, make sure that the Pass Count is set to the correct value.

#### Command buttons

	Accepts the change and closes the dialog box
	Restores the default value
	Closes the Pass Count dialog box
	Opens the Help window

**(36) Delay Count Dialog Box**

The Delay Count dialog box is where you set delay count conditions. From the Main window, select **Event → Delay Count...**

**Figure 4-89. Delay Count Dialog Box**

After the stop condition is satisfied, the number of traces specified in the delay count condition is performed.

- (a) The **Delay Count** box allows you to specify FIRST, MIDDLE, or LAST.

**Table 4-40. Delay Count Condition**

Condition	Meaning
FIRST	Stops tracing after approx. 8,000 frames
MIDDLE	Stops tracing after approx. 4,000 frames
LAST	Stops tracer immediately

When **Run → Cond. Trace ON** is set, conditions are effective for delay counts and all trace/qualify traces.

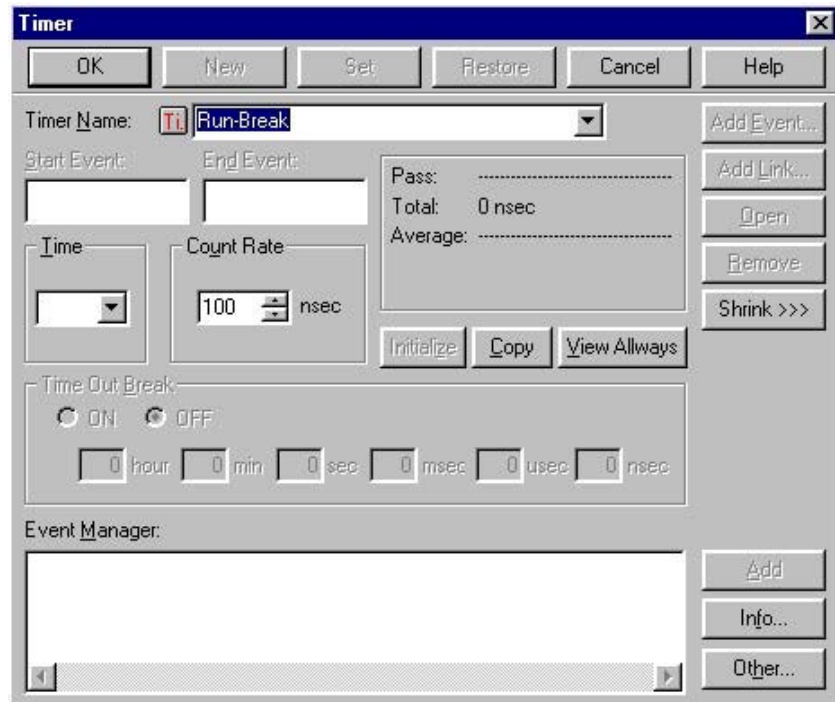
**Command buttons**

OK	Accepts the change and closes the dialog box
Restore	Restores the default value
Cancel	Closes the Delay Count dialog box
Help	Opens the Help window

**(37) Timer Dialog Box**

The timer function measures run time from the beginning of execution until a break. The Timer dialog box is invoked by selecting **Event → Timer...** from the Main window. The Timer dialog box only supports the Run-Break (Run until Break) function. Event-based timing is not possible.

**Figure 4-69. Timer Dialog Box**



### (38) Flash Programming Dialog Box


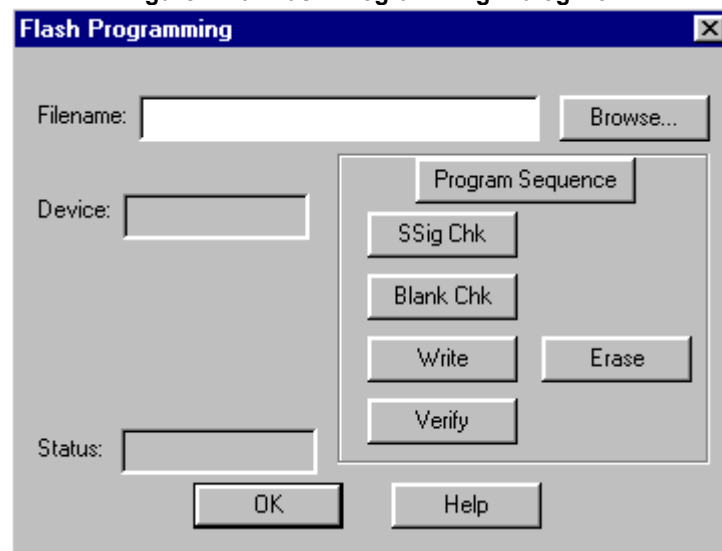
The K0-LCE emulator system incorporates a fully functional flash programmer within the motherboard unit. The programmer's graphical user interface (GUI) may be invoked from the Main window by selecting **File → Flash Program** or clicking .

Figure 4-70. Flash Programming Dialog Box



(a) The **Filename** box allows you to specify the name of a .BIN or .HEX file to be programmed.

Filename:

(b) The **Device** box displays the device detected after you click

Device:

(c) The **Status** box displays the status of each programming operation.

Status:

**Table 4-41. Programming Status**

Button	While Wxecuting	Successful	Not Successful
SSig Chk	SSig checking...	Device name displayed in Device box	Unrecognized Device
Blank Chk	Blank checking...	Blank	Not Blank
Erase	Erasing...	Erased	Failed
Write	Writing...	Done	Failed
Verify	Verifying...	Verified	Failed

(d) **Programming functions**

<input type="button" value="SSig Chk"/>	Performs silicon signature to check for the device
<input type="button" value="Blank Chk"/>	Checks whether the device is blank
<input type="button" value="Erase"/>	Erases the device completely
<input type="button" value="Write"/>	Programs the device with the name specified in the Filename box
<input type="button" value="Verify"/>	Verifies the content of the device
<input type="button" value="Program Sequence"/>	Performs Silicon Signature Check, Blank Check, Erase (if necessary), Write, and Verify in sequence.

(e) **Command buttons**

<input type="button" value="OK"/>	Closes the dialog box
<input type="button" value="Help"/>	Opens the Help window





## 5. FUNCTIONAL OVERVIEW

This chapter describes the ID's functional operation.

### 5.1 Operating Modes

The ID has three modes that control system operation and the emulation and analyzer functions (Table 5-1). The operating mode is shown on the status bar in the Main window.

**Table 5-1. Operating Modes**

Mode	CPU	Tracer
Break mode	Disabled	Disabled
Emulation mode	Enabled	Disabled
Trace mode	Enabled	Enabled

#### 5.1.1 Break Mode

In Break mode, the emulation and analyzer functions are disabled.

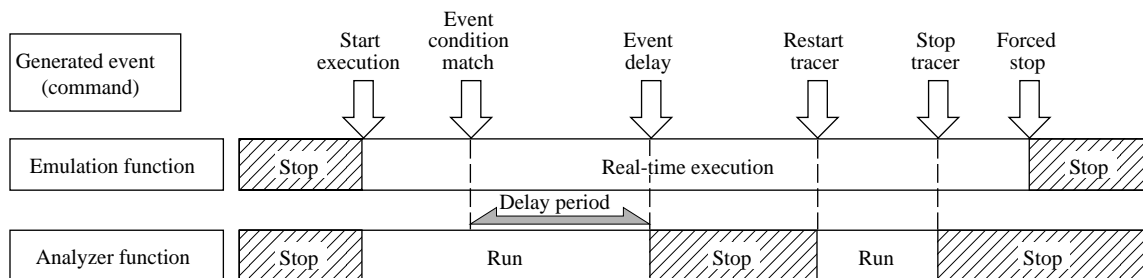
#### 5.1.2 Emulation Mode

In Emulation mode, the emulation function is enabled and the trace function is disabled. The user program is executed and the analyzer functions are performed.

#### 5.1.3 Trace Mode

In Trace mode, the emulation and analyzer functions are enabled.

**Figure 5-1. Example of System Operation**



## 5.2 Basic Functions

This section describes the ID's debugging functions.

### 5.2.1 Clock Selection Function

This function specifies the clock source to be supplied to the target device and is set at the time of power up or by means of the Configuration dialog box. There are two methods for supplying the clock.

### 5.2.2 Mapping function

This function enables you to map address regions, except those for internal ROM and the SFRs. Mapping is set at power up or by means of the Configuration dialog box.

### 5.2.3 Stack area

Creating a stack area prevents stacking outside the stack area. If the target device attempts to perform stacking outside the stack area, the system generates a stack overflow break condition.

### 5.2.4 Reset function

This function resets the LCE or the target device and is specified in the Reset Debugger dialog box.

### 5.2.5 Load function

This function separately loads the debugging environment, object files, load module files, and symbol files. Two types of files are loaded: a view file for screen reference and a data file that updates the data in the ID. A view file records screen data.

**Table 5-2. View Files**

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 the 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 the trace data

Table 5-3. Data Files

File	Window	Description
Object file (File name: XXXXXXXX.HEX)	Load Module Selection dialog box	Stores the object code (Motorola®, Intel®) of the user program
Symbol table file (File name: XXXXXXXX.SYM)	Load Module Selection dialog box	Stores the symbols defined in the source by the user for the user program
Load module file (File name: XXXXXXXX.LMF or XXXXXXX.LNK)	Load Module Selection dialog box	Stores the object code and symbols of the user program and the source data
Project file (File name: XXXXXXXX.PRJ)	Project File Load dialog box	Stores the debugging environment and sets the data in the: <ul style="list-style-type: none"> <li>• Configuration dialog box</li> <li>• Extended Option Setting dialog box</li> <li>• Load Module Selection dialog box</li> <li>• Source Text window</li> <li>• Source Path Specification dialog box</li> <li>• Disassemble window</li> <li>• Memory window</li> <li>• Stack Trace window</li> <li>• SFR window</li> <li>• Local Variable window</li> <li>• Trace View window</li> <li>• Event manager</li> <li>• Event Link dialog box</li> <li>• Break dialog box</li> <li>• Trace dialog box</li> <li>• Timer dialog box</li> <li>• Register window</li> <li>• Variable window</li> </ul>
Event setting file (File name: XXXXXXXX.EVN)	Event Manager	Stores the event setting data

### 5.2.5 Emulation function

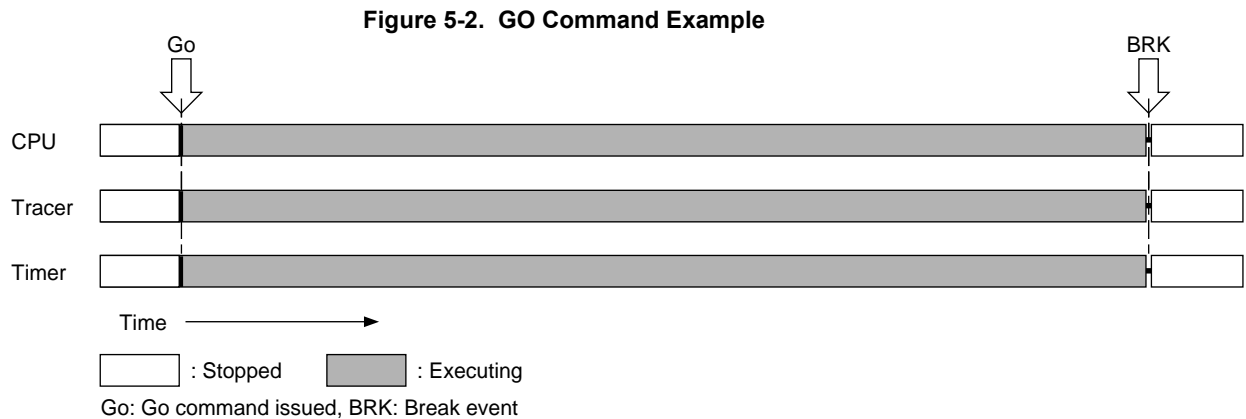
The emulation function starts user program execution by the target device and the analyzer.

**Table 5-4. Real-Time Execution**

Command	Function
GO (▶)	Executes the program starting from a specified address and continuing until a break event is encountered; each analyzer executes the program, entering the stop state based on each event
RETURN (▲)	Executes the program in real time until returning to the CALL function; no action without a CALL function
GO & GO	After a break event, repeats program execution in real time and updates window
COME	Executes the program in real time until reaching the target address or source line; does not generate break events during execution
CPU RESET & GO	Executes the program in real time after the emulation CPU is reset

#### (a) GO Command

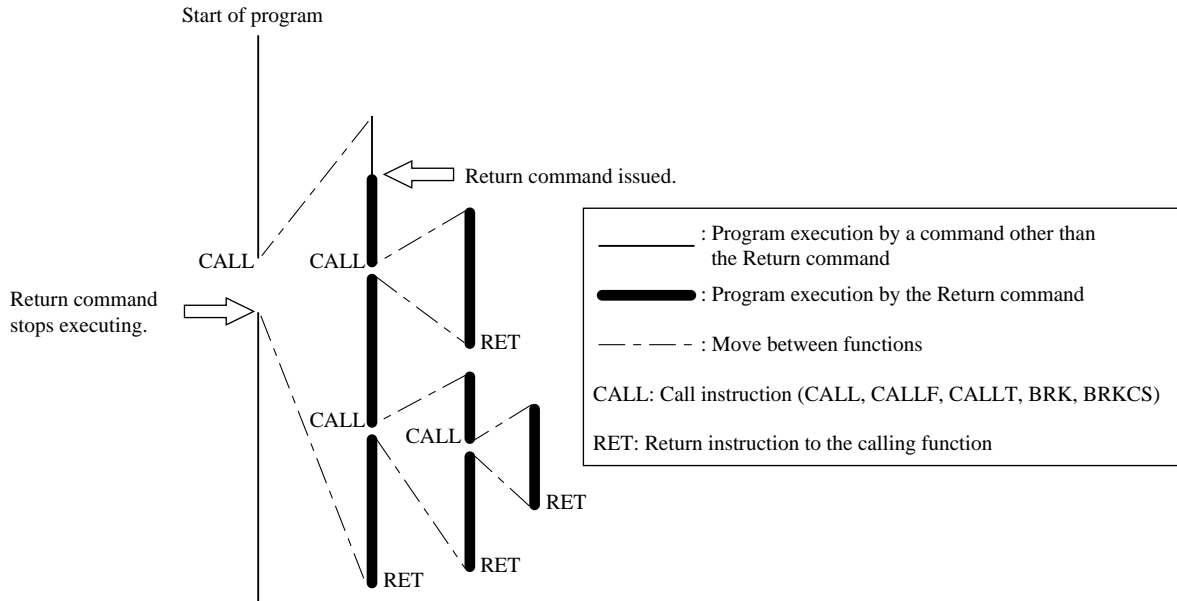
*Real-time execution by the GO command executes the user program from the specified address and stops execution of the user program when a break event is generated (Figure 5-2). Each analyzer enables program operation, and executes or enters the stop state based on each event.*



**(b) RETURN Command**

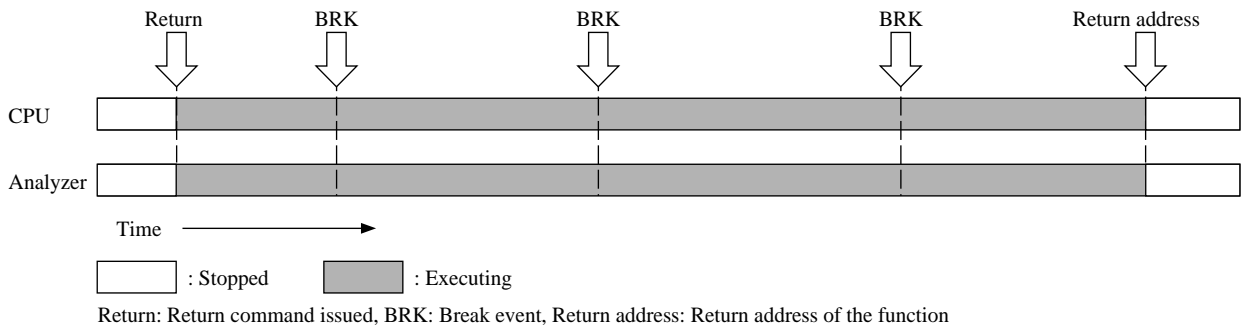
Real-time execution by the RETURN command executes in real time until returning to the CALL function (Figure 5-3).

**Figure 5-3. Conceptual Diagram of RETURN Command**



The RETURN command sets an execution break at the return address of the function and executes in real time (Figure 5-4).

**Figure 5-4. Example of RETURN Command**

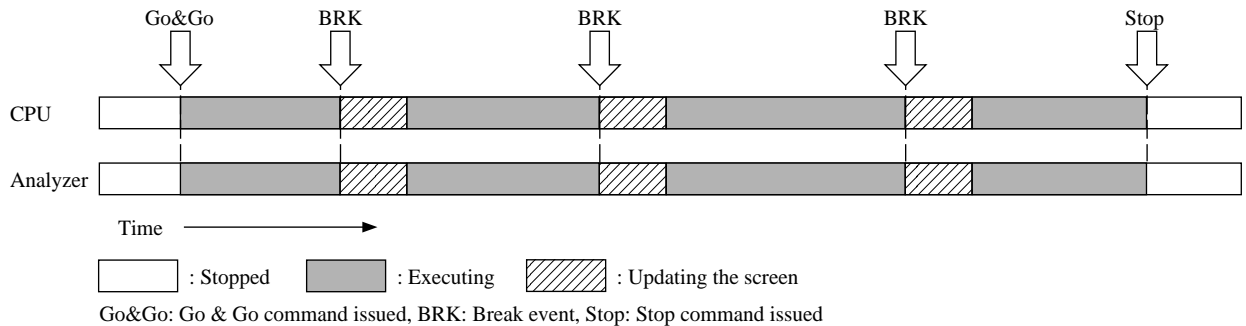


**(c) GO & GO Command**

The GO & GO command executes the program from the specified address. When a break event is generated, the program stops execution and updates the screen of each window. Execution begins again starting from the address where the program stopped. The process repeats until the STOP command is issued.

Each analyzer enables program operation, executing the program according to each event and stopping upon command (Figure 6-5).

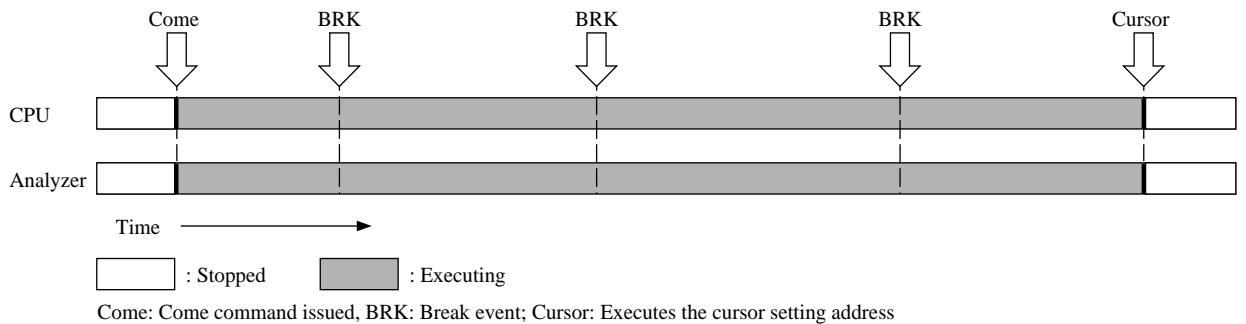
**Figure 5-5. Example of GO & GO Command**



**(d) COME command**

To execute a COME command, move the cursor to the location in the Source Text or Disassemble window where you want to stop program execution. Issue the COME command to execute the program starting from the address in the PC register continuing until reaching the specified stop address. A break event does not cause a break during program execution (Figure 6-6).

**Figure 5-6. Example of COME Command**



**(e) CPU RESET & GO command**

The CPU RESET & GO command resets the emulation CPU and executes the program by the reset vector (Figure 5-7). The operation before a program is executed and after the emulation CPU is reset is identical to a GO command.

**Figure 5-7. Example of CPU RESET & GO Command**

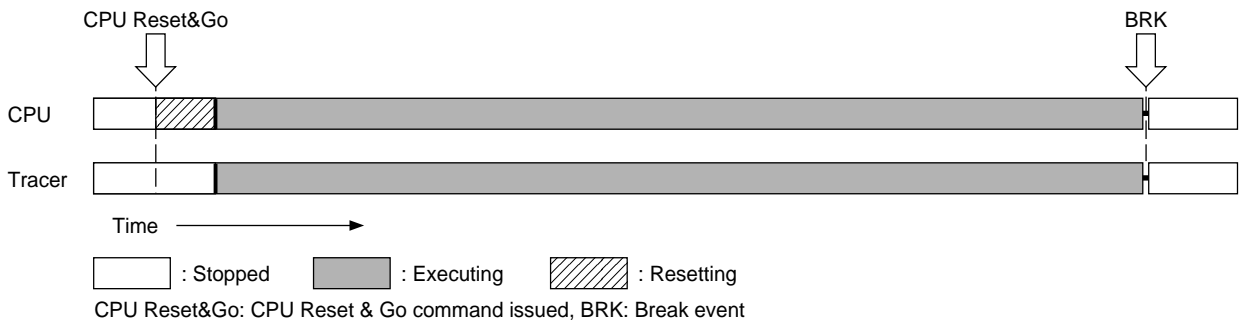


Table 5-5. Non-Real-Time Execution

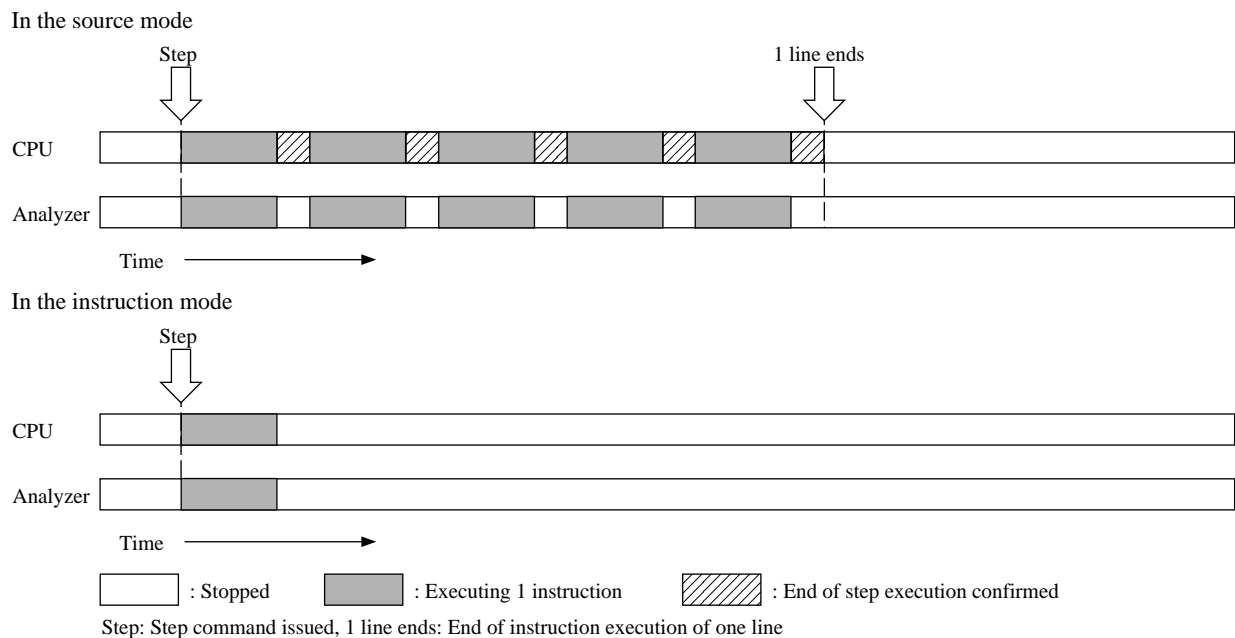
Command	Function
Step (▶)	Step executes at the source level in Source mode
	Step executes at the instruction level in Instruction mode
Next (▶▶)	Next step executes at the source level in Source mode
	Next step executes at the instruction level in Instruction mode
Slowmotion	Continuously step executes

Non-real-time execution functions are broadly classified into functions that execute in steps.

#### (d) STEP command

The STEP command step executes the program in one-line segments starting from a specified source line in Source mode or instruction line in Instruction mode (Figure 5-8). After execution, each window is updated.

Figure 5-8. Example of STEP Command



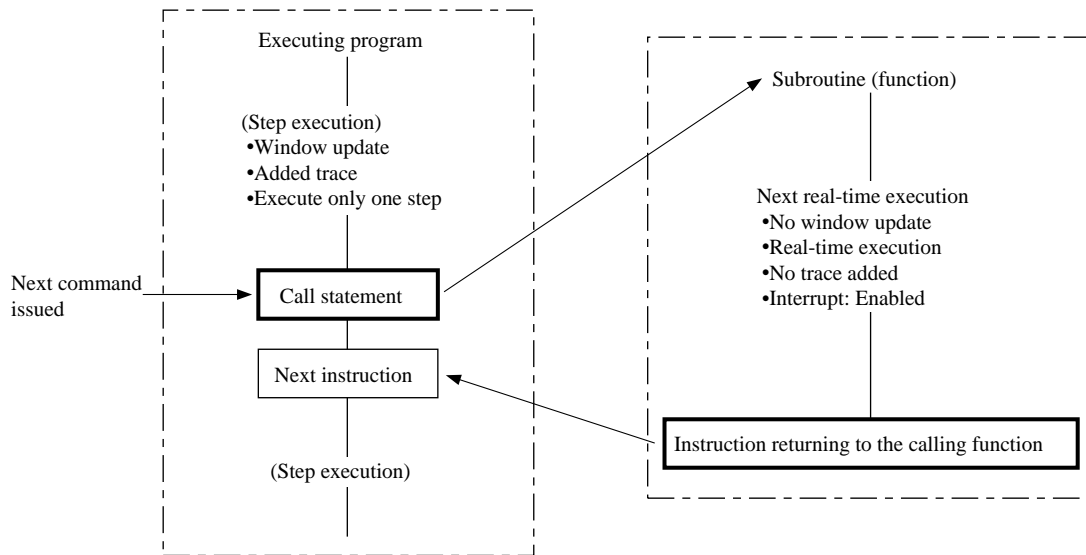
#### (e) NEXT command

The NEXT command varies depending on whether a CALL statement is executed or a statement other than a CALL statement is executed (Figure 5-9). The CALL statement becomes one of the following instructions, depending on the debugging mode.

- In Source mode, CALL becomes the line calling the function
- In Instruction mode, CALL becomes the CALL, CALLF, CALLT, BRK, and BRKCS instructions

When executing the CALL statement, the NEXT command sets the break in the line or instruction following the CALL statement and then executes it in real time. When executing a statement other than the call statement, the NEXT command is identical to a STEP command.

Figure 5-9. Conceptual Diagram of NEXT Command

**(f) SLOWMOTION command**

The SLOWMOTION command executes the program beginning from the specified address in steps of one-line units in Source mode or one-instruction units in Instruction mode. The window is updated after each step. Execution in this manner continues until the STOP command is issued.

**5.2.6 Break Function**

The break function stops execution of the user program by the emulation CPU and stops the analyzer (Table 5-6). The four types of break functions are broadly divided into the following:

- ◆ Event detected break
- ◆ Break caused by satisfying a condition during step execution
- ◆ Forced break
- ◆ Fail-safe break



Table 5-6. Break and Emulation Functions

	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	×	×	○	○

**(1) Event-detected break**

An event-detected break stops user program execution after detecting a specified event condition. This type of break is valid for the GO, GO & GO, and CPU RESET & GO commands. 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 Set dialog box, Event Manager, and Break dialog box.

**(2) Break caused by satisfying a condition during step execution**

A break caused by satisfying a condition during step execution stops program execution by satisfying the stop condition of 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.

**(3) Forced break**

A forced break forcibly stops the execution of a user program and is valid for all of the commands executed in the program. There are two types of forced breaks: those resulting from a STOP command and those resulting from a RESET command. The STOP command is used to temporarily stop a program; the RESET command is used to execute a program starting from the beginning.

**(4) Fail-safe break**

The fail-safe break stops execution when the user program is prohibited from using the memory and the registers. There are three types of fail-safe breaks: a nonmapping break generated when a nonmapping region is accessed; a write-protected break generated when writing to memory that cannot be written (such as ROM), and an illegal SFR access break generated after illegal access to an SFR

region. A fail-safe break occurs if there is a problem in the user program or a mistake in the environment settings of the debugger.

### 5.2.7 Trace functions

A trace function accesses the memory during user program execution 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 trace conditions can be set in the Trace dialog box. The settings for the trace data display can be specified in the Trace Data Select dialog box. The main functions related to trace execution and trace display are summarized below.

#### Trace operation

- Operation during real-time execution
- Operation during step execution
- Operation during next step execution

#### Trace condition setting function (trace dialog box)

- Trace mode specification
- Qualify trace setting

#### Trace data display, format, and search condition settings

- Trace data display specification
- Trace data search condition setting

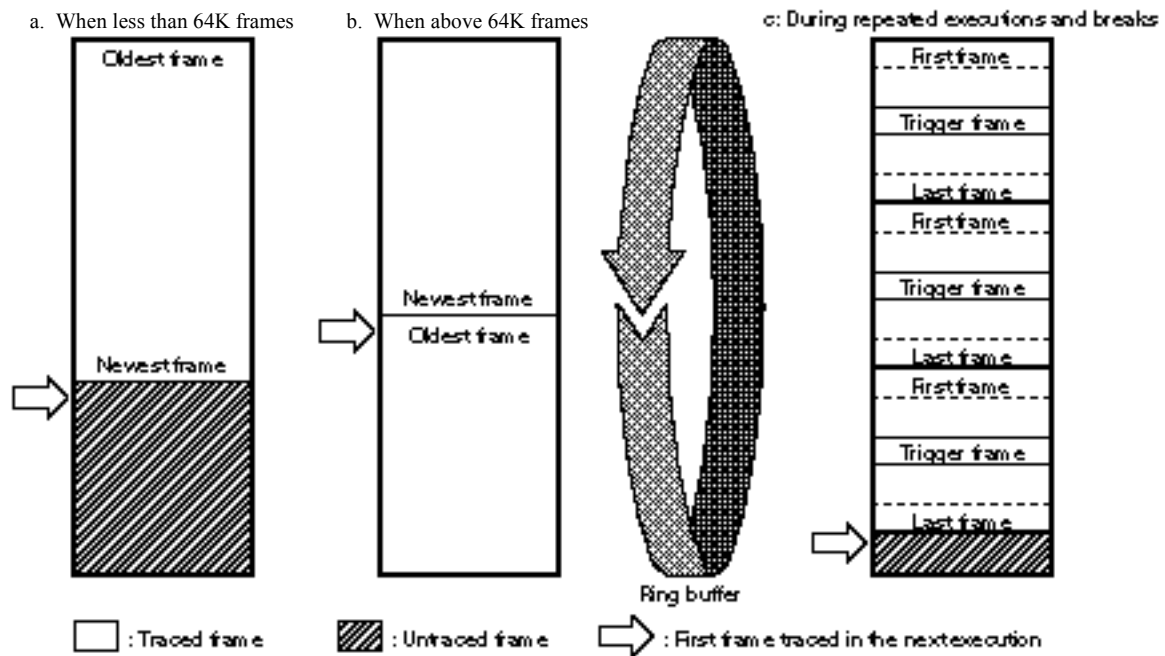
### (1) Relationship between trace execution and trace memory

The trace is divided into trace blocks according to the periods shown below.

- ◆ Block from real-time execution to a break by an event
- ◆ Block from emulation execution until the generation of a fail-safe break
- ◆ Block from emulation execution until a forced break
- ◆ Step execution block

The trace memory is a 64K frame ring buffer. Therefore, if 64K frames are exceeded during a trace, the latest trace data is overwritten in the oldest frame.

Figure 5-10. Trace Memory Concept

**(2) Trace operation**

Tracer operation depends on the execution state.

**(a) 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 box are set up, the trace operation ends.

**(b) Operation during step execution**

The tracer runs for each step execution. The trace data in one step is added to the tracer for each subsequent step execution.

**(c) Operation during next step execution**

When executing an instruction other than a call instruction (CALL, CALLF, CALLT, BRK, BRKCS), operation is identical to the operation during step execution. When executing a call instruction (CALL, CALLF, CALLT, BRK, BRKCS), operation is identical to the operation during real-time execution. Real-time execution stops by returning to the call function.

**(3) Trace condition setting function (Trace dialog box)**

The following specifications can specify the trace conditions. If these specifications are not made, "All trace" is performed. In other words, trace data is recorded for each instruction in the user program. A complete trace or the type of conditional trace is specified. There are two types of conditional trace: qualify trace and section trace. The trace mode specified becomes valid.

**(a) 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.

**(b) Sectional trace setting**

Tracing is controlled by the specified start and end events.

**(4) Trace data display, format, and search condition settings**

The data can be displayed or hidden in the trace view window, and the display conditions can be set.

**(a) Trace data display specification**

The display screen can be effectively used by specifying the display of the trace data. Trace data display can specify displaying or hiding the following data by clicking **View** → **Trace View**.

**Table 5-7. Trace Data Display**

Menu	Trace View Window	Description
Frame number ( <u>E</u> )	Frame	Temporal order written to the trace memory by the frame number in the trace memory (range from 0000 to 8192)
Instruction fetch address ( <u>A</u> )	Addr	Fetch address
Instruction fetch data ( <u>D</u> )	Data	Fetch data
Memory access address ( <u>R</u> )	Addr	Access address
Memory access data ( <u>M</u> )	Data	Access data
Memory access status ( <u>S</u> )	Statu	Access status RW: data read or write by a user program RD: data read by a user program WD: data write by a user program
Disassemble ( <u>I</u> )	DisAsm	Disassemble result

**(b) Trace data search condition setting (Not supported)**

The search conditions for trace data can be selected and specified by any or all of the items in Table 5-8 in the trace window dialog box.

**Table 5-8. Trace Search Items**

Specification Item	Description	Specified Range	Default
Address	Search address	0 to 0FFFFH	0XXXXH
Data	Search data	0 to 0FFH	0XXH
Kind of frame ?	Search data type All Frame: all of the frames Step: step execution frames Next: frames other than step execution frames	Same as on left	All frames

### 5.2.8 Event setting and detection function

The event setting and detection functions set the conditions for stopping user program execution by the emulation CPU and for starting and stopping the trace operation by the analyzer. There are four types of event condition setting and detection functions:

- ◆ Bus event condition setting function
- ◆ Execution event condition setting function
- ◆ Event condition link setting function
- ◆ Integrated function of event detection function (break event setting and trace event setting)

#### (1) Event condition setting function

This function sets the Event Condition register to stop user program execution and to start or stop a trace by the analyzer. The event-detected condition specified in Event Set dialog box or Event Link dialog box is not valid unless it is set in the Event Mode register by the Event Manager, Break dialog box, or Trace dialog box. There are three types of functions set by the event-detected condition.

##### (a) 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.

##### i. **Bus event condition register**

A maximum of four conditions can be set in the Bus Event Condition register (BRA) in the Event Set dialog box.

##### ii. **Event condition**

Table 5-9 lists the items that can be set in the event-detected condition.

**Table 5-9. Path Event Detection Condition**

Item	Status	Description
Address	Address	Address (address range)
	Mask	Address mask
Status	R	Read by a program
	W	Write by a program
	R/W	Read or write by a program
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

**(b) 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.

**i. Execution event condition register**

A maximum of eight conditions can be set in the Execution Event Detection register (BRS) in the Event Set dialog box.

**ii. Event condition**

Table 5-10 lists the items that can be set in the event-detected condition.

**Table 5-10. Execution Event Detection Condition**

Item	Status	Description
Address	Address	Address (address range)
	Mask	Address mask
Status	Run	Program execution
Data	Data	Data value
	Mask	Data mask value

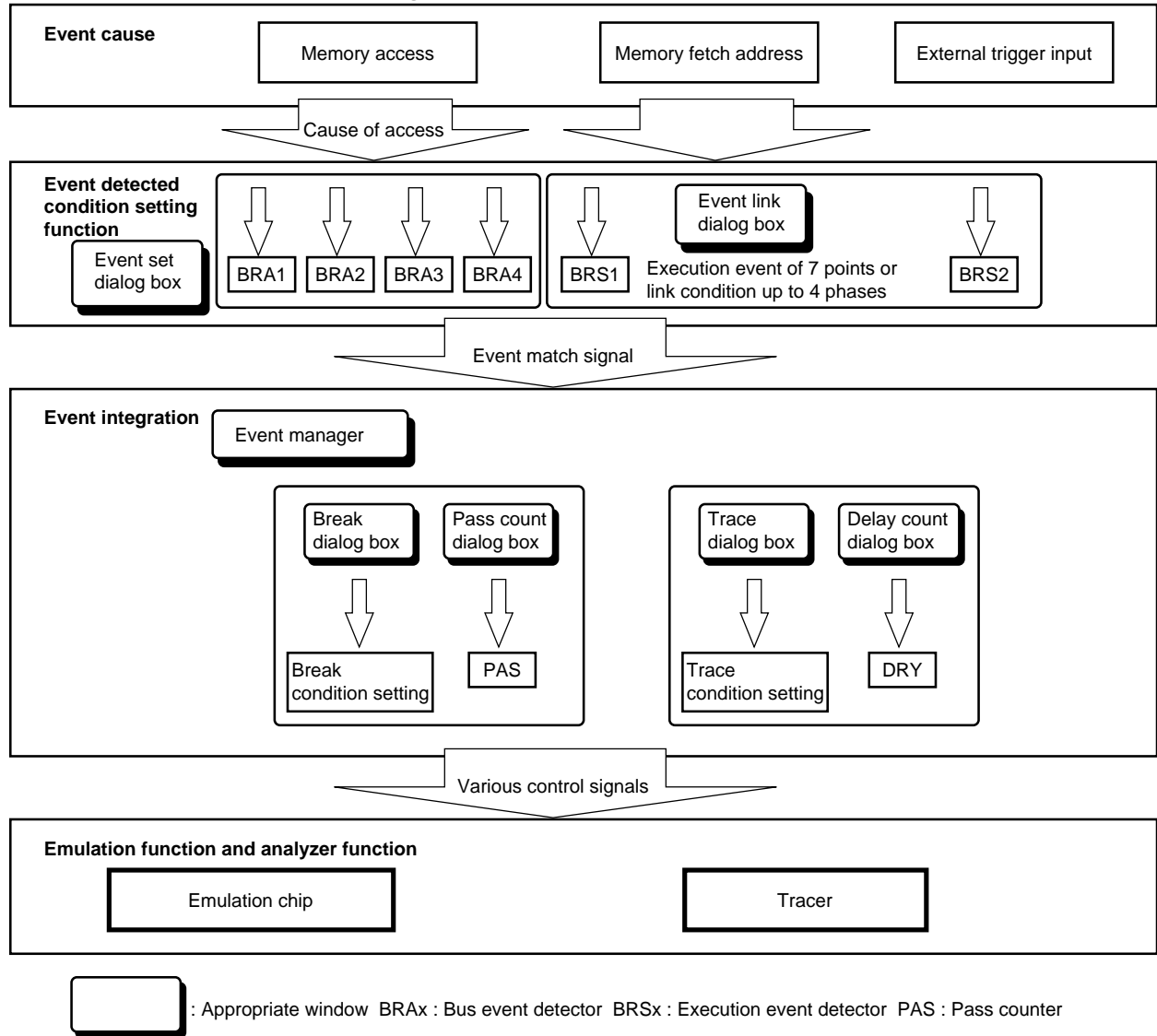
**(c) Event condition link setting function**

The execution event conditions registered in the Event Set dialog box can register the event connect conditions in the event link dialog box. However, when the event link is used, a normal execution event cannot be used.

**(2) Event detection**

The steps for setting and detecting event conditions are illustrated in Figure 5-11.

Figure 5-11. Event Detection



### 5.2.9 Register manipulation functions

The register manipulation functions display and change the contents of the general-purpose registers and SFR.

#### (1) General-purpose register manipulation function (Register window)

This function displays and changes the contents of the control registers (PC, SP, PSW) and general-purpose registers (RP0, RP1, RP2, RP3, AX, BC, DE, HL). The PSW flag names (Z, AC, CY) are displayed or changed for PSW.

#### (2) Special register manipulation function (SFR window)

This function displays and changes the contents of the special function register (SFR), which can be manipulated by bits.

### 5.2.10 Memory manipulation functions

These functions, available in the Assemble window and the Memory window, use mnemonic codes, hexadecimal codes, and ASCII characters to change the memory contents.

### 5.2.11 Save function

The save function stores the object codes in the low-cost emulator and the debugging environment in a file on a disk drive connected to the host machine.

### 5.2.12 Time measurement function

This function measures the entire run time, until a break after execution begins. The time measurement measures the accumulated run time.

**Table 5-11. Timer Specifications**

Item	Contents
Accumulated run time	203.45 ns resolution Maximum 14 minutes, 33 seconds
Time measurement count	Maximum 65,535 times

### 5.2.13 Source debugging

In debugging mode, object programs and source programs can be debugged. Debugging of the source program is called source debugging. Compared to debugging of object programs, source-level debugging has several advantages:

- ◆ Debugging is possible while examining the C language or structured assembler source.
- ◆ Breakpoints can be set in the source and step execution can be performed.

Generally, if a breakpoint is set, the real address of the breakpoint is specified. However, in source-level debugging, the position where a breakpoint is set is specified in the source program using the cursor. 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.

- (1) If assembling or compiling using NEC software, the options must be specified to include the source debugging data in the object.

**Table 5-12. Source Debugging and Option Specification**

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 the -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 box.
- (3) In source-level debugging, always load the load module file created by the linker. Even if the object file created by the object converter is loaded, source debugging will not be possible.



## Appendix A Error Messages

This appendix explains error and warning messages. As shown in Figure A-1, an error message has the following format: `[Error number] + [Type] + [Message]`.

Figure A-1. Error Message

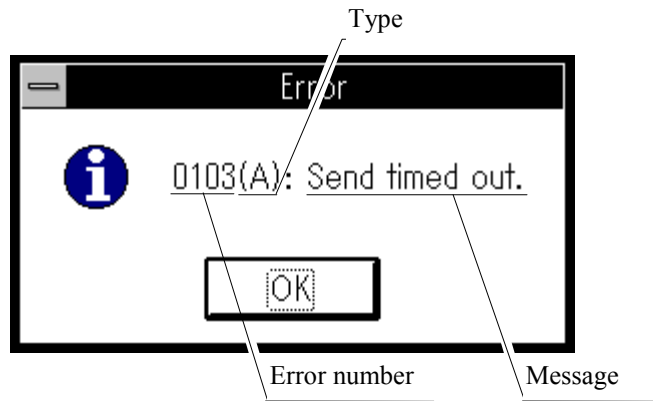


Table A-1. "Type" Codes

Type	Description
A	Fatal error ( <u>A</u> bort error); processing stops and debugging terminates
F	Syntax error ( <u>F</u> atal error); processing stops and open windows and dialog boxes close
W	Warning ( <u>W</u> arning); processing stops and open windows and dialog boxes remain open

Table A-2. "Message" Codes

Message	Description
xxx	Inputs 3 digits in the device name
yyy	File name
zzz	Function name

Table A-3. Error Number Descriptions (1/9)

Error Number	Type	Message	Description
-	-	Can't open this file. Please make sure, now Active Window.	Illegal format of the project file or corrupted file contents; loading of the project file stops
-	-	Cannot find "string"	Cannot find search string; search stops or file open stops (if the specified file did not contain data)
-	-	Event Name is not set.	No event name; event name not registered
-	-	Even number already exists.	An event with the same number cannot be registered twice. Change the number of the event to be 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 to save the changes. After freeing more memory, execute again.
-	-	Other view mode window exists.	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 contents of the specified view file (.MEM, .TVW, .DIS) are longer than 1,000 lines. The display was stopped.
-	-	"Event name" already exists.	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 timed out	Data cannot be sent to the in-circuit emulator (IE). 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 again.
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.

Table A-3. Error Message List (2/9)

Error Number	Type	Message	Description
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, restart.
01ad	F	No match device file of version	ID number of the emulation board does not match the number in the device definition file. Check that the specified device (device file) is correct.
02a0	F	Bus hold error	Bus hold. The user program cannot execute.
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 move 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.	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	Trace is 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.
0316	F	This event number cannot be used	An event that cannot be used is used. Specify an event that can be used.
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	Tried to set more than the number of event conditions that can be simultaneously used. A maximum of four bus event conditions and a maximum of four execution event conditions can be used simultaneously.

Table A-3. Error Message List (3/9)

Error Number	Type	Message	Description
0407	F	Initialized data overflow	The amount 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 MB.
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.)
04a6	F	External Trigger event conditions overflow	More than 2 external trigger conditions are set. The maximum number of external trigger conditions is 1.
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 of 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 MB.
100b	F	Program Is running.	This command cannot be used while a user program is running.
100c	F	Different Bus Size	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.

Table A-3. Error Message List (4/9)

Error Number	Type	Message	Description
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	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.
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 and 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	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.

Table A-3. Error Message List (5/9)

Error Number	Type	Message	Description
3011	F	Different Access Size, Source and 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.
4005	F	Illegal data	The data is illegal.
4006	F	Can't access 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 box.
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.

Table A-3. Error Message List (6/9)

Error Number	Type	Message	Description
5004	A	Cannot close file	The file close failed.
5005	A	Illegal device format	The format of the device file differs.
5006	A	Cannot initialize device	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.
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 debugger 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 reset	The user program break failed. The CPU was reset and the IE is unstable. Check that the IE is okay and restart.
7f04	F	Illegal address	Tried to execute from an unmapped region.

Table A-3. Error Message List (7/9)

Error Number	Type	Message	Description
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.
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 debugger 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 closed	The file close failed.
a00c	A	File not read	The file read failed. The file may be 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 debugger window.



Table A-3. Error Message List (8/9)

Error Number	Type	Message	Description
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. Or, 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.
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 the load module file.
c008	F	Old coff version	The version of the load module file is different.

Table A-3. Error Message List (9/9)

Error Number	Type	Message	Description
c009	A	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugger 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.
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 MB.
c014	F	Cannot write file	Cannot write to the file.
c100	F	Not supported	The Tektronix format is not supported.
d001	F	Not enough memory	The memory is insufficient. Exit unused applications, or close the debugger 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 debugger window.
e009	F	Illegal file format	The file format is illegal.






## Appendix B Key Functions

Special function keys can be used to debug effectively with ID operations. In the descriptions, because the key expression differs with the type of keyboard used, common generic key characters are adopted.

**Table B-1. Special Function Key Function List**

Key		Function
PC-9801, 9821 Series	IBM PC/AT Series	
<b>[BS]</b>	<b>[BackSpace]</b>	Deletes the character before the cursor. The cursor moves to the position of the deleted character. The string after the cursor is moved forward.
<b>[COPY]</b>	<b>[PrintScreen]</b>	The entire display screen is written to the clipboard as a bit image (Windows function).
<b>[ESC]</b>	<b>[Esc]</b>	(1) Closes the pull-down menu. (2) Closes the modal dialog.
<b>[GRPH]</b>	<b>[Alt]</b>	Moves the cursor to the menu bar.
<b>[HELP]</b>	<b>[End]</b>	The last line is displayed. The cursor simultaneously moves to the last line.
<b>[HOME CLR]</b>	<b>[Home]</b>	The first line is displayed. The cursor simultaneously moves to the first line.
<b>[ROLL UP]</b>	<b>[PageUp]</b>	The screen scrolls up one screen. The cursor simultaneously moves to the top of the screen.
<b>[ROLL DOWN]</b>	<b>[PageDown]</b>	The screen scrolls down one screen. The cursor simultaneously moves to the top of the screen.
<b>[SPACE]</b>	<b>[Space]</b>	Inserts one space.
<b>[TAB]</b>	<b>[Tab]</b>	The cursor moves to the next item.
<b>[↑]</b>	<b>[↑]</b>	The cursor moves up. If the cursor is at the top of the screen, the screen scrolls down by one line each time.
<b>[↓]</b>	<b>[↓]</b>	The cursor moves down. If the cursor is at the bottom of the screen, the screen scrolls up by one line each time.
<b>[←]</b>	<b>[←]</b>	The cursor moves left. If the cursor is at the left of the screen, the screen scrolls to one item to the right.
<b>[→]</b>	<b>[→]</b>	The cursor moves right. If the cursor is at the right of the screen, the screen scrolls to one item to the left.
<b>[↵]</b>	<b>[↵]</b>	Confirms the input data.

Table 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 Assemble window opens.
B	Sets a breakpoint at the selected line.
C	Copies the selected string to the clipboard buffer.
F	The window switches to Modify mode; same as <span data-bbox="1101 583 1214 632">ToModify</span>
G	Runs the program; same as 
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 are displayed from that address. The Memory window opens.
O	When the Source Text window is current: The source view file is selected. The Source File Selection dialog box is opened. Otherwise: The appropriate view file for the current window is displayed. The View File Save dialog box opens.
P	Program execution pauses; same as 
R	Step executes until returning to the calling function; same as 
S	The displayed contents of the current window are saved in the view file.
T	Executes in steps; same as 
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	The window switches to View mode; same as <span data-bbox="1089 1539 1203 1587">ToView</span>
X	Executes the next step; same as 
Z	The previous editing operation is undone.

---

## Appendix C INDEX

### [A]

Active state	53
Address specification	125
Addresses	21, 111
ASCII display	69, 136

### [B]

Break cause	65
Break event setting function	105, 131
Break function	232
Break mode	84
Breakpoint set and delete functions	104
Button	
Function button	51
Push button	51
Radio button	51
Scroll arrow	52

### [C]

Character set	18
Check box	51
Click	51
Clock selection function	224
CPU status	65

### [D]

Data size specification	152
Debugger file list	26
Debugging modes	
Instruction level	48
Source level	48
Delay count	220
Dialog boxes	
Auxiliary dialog boxes	59
Confirmation dialog boxes	59
Display dialog boxes	59
Display/setting dialog boxes	59
Modal dialog boxes	58
Modeless dialog boxes	58
Selection dialog boxes	58
Setting dialog boxes	58
Specification dialog boxes	58

Disassemble	128
Double click	51
Drag & drop	51
Drive voltage selection	79
Drop-down list	53

### [E]

Emulation CPU selection	79
Emulation execution function	227
Environment	25
Equipment connections	25
Errors and warnings	54, 243
Event	
Break event condition	170
Event condition	149
Event link condition	164
Event management	154
Trace event condition	174
Event display function	104, 130
Event setting and detection function	237
Execution control	55
Exiting	39, 214
EXPC.INI file	25
Expressions	22

### [F]

File specification	19
Files	19, 48
Font	103
Function specification	49

### [G]

GUI function	17
--------------	----

### [H]

Hold state	53
------------	----

### [I]

Icon	76, 106, 124, 133, 137, 148, 186, 200
Installation	25

### [J]

Jump function-----105, 131, 136, 147, 157, 192

**[L]**

Line specification -----50

List of debugging windows -----60

List of key functions -----253

Load function ----- 85, 91, 201, 225

**[M]**

Mapping function----- 80, 225

Mark----- 156

## Memory

    Compare ----- 142, 144

    Copy ----- 140

    Initialization----- 138

    Manipulation ----- 134, 240

Menu bar-----54, 62, 66, 116, 158, 193

Modify mode -----54

Mouse -----51

**[N]**

Non-real-time execution functions-----230

Numerical values -----20

**[O]**

On-line assembly -----128

Operands -----20

Operators -----22

**[P]**

Pass count----- 160

Pin mask -----80

Point mark ----- 103, 183

Program counter setting function ----- 105, 131

Project file -----85, 88

Pull-down menu-----52

**[R]**

Real-time execution -----227

Real-time RAM sampling -----50, 84

Register display -----191

Register manipulation function -----240

Registers-----21, 191

Reset function -----211, 225

**[S]**

Save function-----88, 205, 240

Scroll bar ----- 52

Search -----107

SFR display-----197

Software break----- 84

Source debugging -----241

Source file----- 99

Source path-----101

Source text display -----102

Stack frame number-----49

Starting ----- 37

Status bar ----- 53

Step execution----- 64, 71

Structures -----49

Symbols -----22

Symbol to address -----110

System operating modes-----223

System operating states-----224

**[T]**

Terms ----- 24

Time measurement function----- 179, 240

Tool bar----- 52, 64

Trace function-----234

Trace mode-----181

Trace view -----187

**[V]**

## Variable

    Display ----- 111, 113, 115

    Specification----- 111, 113, 119

Version-----213

View mode----- 54

**[W]**

Watch function ----- 17

Wild cards----- 19

## Window

    Display window ----- 56

    Display/setting window----- 56

    Execute window ----- 55

    Management window----- 57

Window connect function -----105, 131, 136, 184

Write mode ----- 84



