To all our customers

# Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Simple Emulator System Package for 740 Family

# M3XXXXT-PAC User's Manual

# PDB38M

## CONTROL SOFTWARE

Second Edition August, 2001

Mitsubishi Electric Corporation
Mitsubishi Electric Semiconductor Application Engineering Corporation

RENESAS
Renesas Technology Corp.

Second Edition August 3rd 2001



Keep safety first in your circuit designs!

* Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation put the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Precautions to be taken when using this manual

* These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation, Mitsubishi Electric Semiconductor Application Engineering Corporation or a third party.
* Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation assume no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
* All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and is subject to change by Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation, Mitsubishi Electric Semiconductor Application Engineering Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein. The information described here may contain technical inaccuracies or typographical errors. Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation assume no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors. Please pay attention to information published by Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation by various means, including Mitsubishi Semiconductor Homepage (http://www.mitsubishichips.com/) and Mitsubishi Tool Homepage (http://www.tool-spt.maec.co.jp/index_e.htm).
* When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Mitsubishi Electric Corporation and Mitsubishi Electric Semiconductor Application Engineering Corporation assume no responsibility for any damage, liability or other loss resulting from the information contained herein.
* Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation, Mitsubishi Electric Semiconductor Application Engineering Corporation, or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
* The prior written approval of Mitsubishi Electric Corporation or Mitsubishi Electric Semiconductor Application Engineering Corporation is necessary to reprint or reproduce in whole or in part these materials.
* If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
* Please contact Mitsubishi Electric Corporation, Mitsubishi Electric Semiconductor Application Engineering Corporation or an authorized Mitsubishi Semiconductor product distributor for further details on these materials or the products contained therein.

Mitsubishi Tool Homepage http://www.tool-spt.maec.co.jp/index_e.htm

RENESAS
Renesas Technology Corp.

# Preface

The PDB38M is the debug software for Mitsubishi 8-bit Microcomputers' Simplified Emulator System.

# Product Overview

PDB38M consists of the following items:

1. Floppy disk
2. User's Manual
3. Release Notes
4. Software License Agreement (attached at the end of Release Notes)
5. User Registration Card

The Release Notes included with your software package contain supplements to the user's manual, so be sure to read it. Since the User Registration Card is needed when it is necessary to contact the customers for urgent communication, please return your Registration Card to Mitsubishi.

If any of the above items is not included with your package, please contact Mitsubishi or its distributor.

# Rights to Use the Program

The rights to use the programs included in this software package shall conform to the Software License Agreement. The PDB38M program can only be used in developing the purchaser's product, and cannot be used for any other purposes.

This manual does not convey any guarantee or license for the rights to use the software.

RENESAS
Renesas Technology Corp.

# Definition of Terms

Some terms used in the PDB38M manual are defined as follows:

| Term | Meaning |
|---|---|
| M38000T-SBI | Refers to the emulator main unit. |
| SRA74M | Refers to the assembler. |

# Contents of Body Text

RENESAS
Renesas Technology Corp.

# Contents of Figures

RENESAS
Renesas Technology Corp.

# Contents of Tables

# Part I: Setting Up

RENESAS
Renesas Technology Corp.

# 1. Contents of Software Package

This chapter describes the contents of the PDB38M software package. Check that all items are included in your package before installing PDB38M.

## 1.1. List of Package Items

Table 1-1 lists the items included in the PDB38M software package.

Table 1-1 Contents of PDB38M Software Package

| Product Name | Quantity |
|---|---|
| Floppy disk | 1 set |
| PDB38M V.1.00 User's Manual | 1 copy |
| Release Notes<br>(Software License Agreement included) | 1 copy |
| User Registration Card | 1 card |

# 2. Operating Environment

PDB38M operates on the host computers listed in the table below that run under the OS versions listed below.

Table 2-1 PDB38M's Operating Environment

| | |
|---|---|
| Host Computer | IBM PC/AT or its compatibles |
| OS | ・ Microsoft Windows Version3.11 English version<br>・ Windows95 English/Japanese version |
| CPU | i386 or more（i486DX4-100Mhz or Pentium75MHz or more is recommended） |
| Memory | 8M byte（16M bytes is recommented） |

# 3. System Configuration

Before PDB38M can be operated, the following pieces of equipment are required:

1. Host computer
2. Emulator
3. Emulation pod
4. Target system

Figure 3-1 shows a configuration of the emulator system built around the Emurator Main Unit(M38000T-SBI).

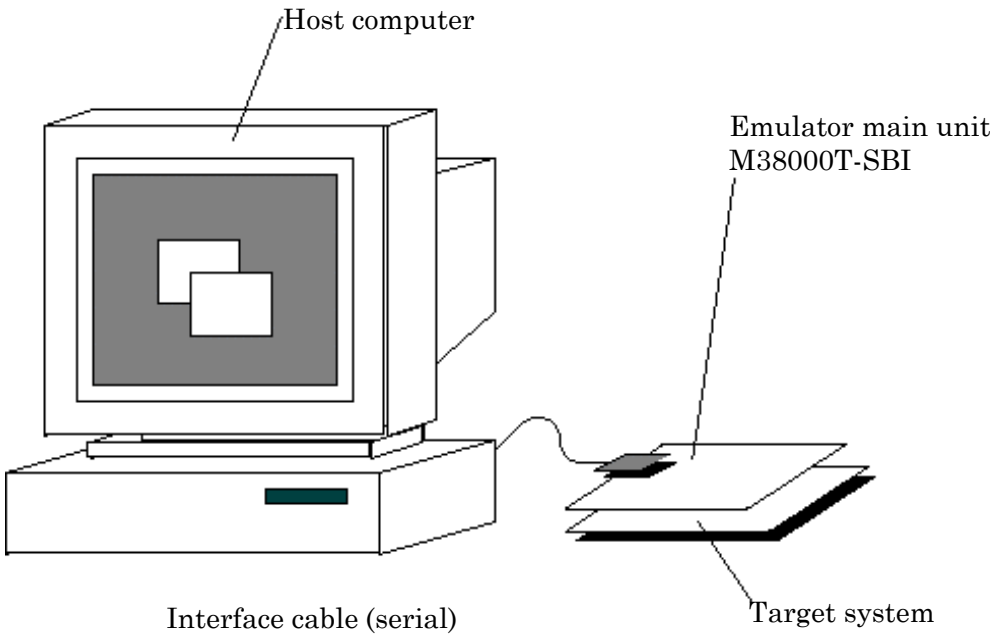



Figure 3-1 Configuration of the Emurator Main Unit(M38000T-SBI)

## 3.1. Host Computer

PDB38M operates on the host computers listed in the table below.

Table 3-1 Host Computer

| Name of Host Computer | OS |
|---|---|
| IBM PC/AT or its compatibles IBM PC/AT | ・Microsoft Windows Version3.11 English version<br>・Microsoft Windows Version3.1 Japanese version<br>・Windows95 English/Japanese version |

## 3.2. M38000T-SBI Emulator

The M38000T-SBI is the emulator main unit for Mitsubishi 8-bit Microcomputers' Simplified Emulator System. For details about this emulator, refer to the M38000T-SBI Emulator User's Manual.

## 3.3. Target System

This is the customer's target system. The simplified emulator system comes with a terminal-processed target system that can be used in place of your target system.

## 3.4. Method of Communication

The Emurator Main Unit(M38000T-SBI) and the host computer can be connected with an RS-232C serial interface.

The emulator main unit and host computer communicate via this interface at a fixed baud rate of 9,600 bps.

# 4. Files Handled by PDB38M

The files handled by PDB38M can be classified into the following two types:

Files into to PDB38M (input files)

1. Intel HEX Format File
2. Symbol File
3. Definition file
4. Script file
5. Environment setup file

Files output by PDB38M (output files)

1. Intel HEX Format File
2. Disassembled file
3. Log file
4. View file

Each of these files are detailed in the next pages.

# 4.1. Input Files

## 4.1.1. Intel HEX Format File

An Intel HEX Format File contains machine language statements. Its file attribute is ".hex" This file is generated by the linker for the SRA74M assembler, LINK74M.

If you want to debug the program at the source level, the symbol file described below must be read into the emulator system.

## 4.1.2. Symbol File

A Symbol File contains debug information such as information about symbols and line numbers. Its file attribute is ".sym" A Symbol File is generated by specifying the "-s" option when executing LINK74M.

## 4.1.3. Definition File

This file contains the information specific to the target MCU. This file is included in the software package. Its file name is M3XXXX.I38.

When starting up PDB38M (i.e., when setting up environment variables), you must specify the definition file name corresponding to the type of MCU used in your application. PDB38M cannot be started up unless this definition file is specified. Specify a definition file from the Init dialog box when Setting Up the environment variables. For details, refer to a section in this part that is titled "Using Init Dialog Box.

## 4.1.4. Script File

This file is used to automatically execute script commands from a file. Script commands are a command that can be executed in PDB38M's script window.

Use an editor you have to create a script file. Although any desired names can be used for the file name and file attribute, Mitsubishi recommends using ".scr" for the file attribute. (When script files are listed for selection in a dialog box, the files with attribute ".scr" are given priority over other files.)

To open a script file, activate the script window and choose menu [Option] → [Script] → [Open]. Then choose menu [Option] → [Script] → [Run] to execute the script file.

### 4.1.5. Environment Setup File

This file retains the information associated with PDB38M's environment settings.

An environment setup file is automatically created by PDB38M. You cannot create or edit an environment setup file for yourself.

## 4.2. Output Files

### 4.2.1. Intel HEX Format File

PDB38M allows you to save the contents of your specified address range to an Intel HEX Format File (upload).

The Intel HEX Format File thus saved can be redownloaded from the computer to the emulator.

### 4.2.2. Disassembled File

This is a text file containing the disassembled results of the program memory. To save this file, choose menu [File] → [Save Disasm].

Note that this disassembled file is a text file for reference purposes only and cannot be reassembled or re-downloaded.

Figure 4-1 shows an example of a disassembled file.

```
C000     78          RESET:    SEI
C001     58                    CLI
C002     78                    SEI
C003     32                    SET
C004     F8                    SED
C005     12                    CLT
C006     D8                    CLD
C007     3C04FB                LDM     #04H,FBH :CPUMOD
C00A     A2BF                  LDX     #BFH
C00C     9A                    TXS
C00D     3CFFC1                LDM     #FFH,C1H :P0D
C010     3C00C0                LDM     #00H :P_PORT,C0H :P0
C013     3CFFC3                LDM     #FFH,C3H :P1D
C016     3C01C2                LDM     #01H :TIME00,C2H :P1
C019     3C00C5                LDM     #00H :P_PORT,C5H :P2D
C01C     3CFFC9                LDM     #FFH,C9H :P4D
C01F     3C01C8                LDM     #01H :TIME00,C8H :P4
```

Figure 4-1 Example of disassembled file

The file contents consist of the following items, from left to right:

- Address
- Object code
- Label equivalent to address
- Disassembled result

### 4.2.3. Log File

This is a text file that contains the results of script commands executed in the script window. Its file attribute is ".log."

After activating the script window, choose menu [Option] → [Log] → [On] and specify your desired log file name. Then the results of the script commands executed thereafter can be saved to a log file in your specified name. Choose menu [Option] → [Log] → [Off] to finish saving the execution results to a log file.

### 4.2.4. View File

This is a text file that contains the script window's current display contents. All contents of the script window that can be viewed by scrolling the display are saved to this file.

A view file has the same contents as those of a log file. The difference is that while the contents of a log file are the results of commands that are executed after you have specified outputting to a log file, a view file allows you to save the results of commands that have already been executed. However, the contents saved to a view file are limited to those that can be viewed by scrolling the display (contents older than that cannot be saved because they are discarded).

Here is the procedure for saving the script window's display contents to a view file. After activating the script window, choose menu [Option] → [View] → [Save] and specify your desired view file name.

# Part II: Window Operation

RENESAS
Renesas Technology Corp.

# 5. Using PDB38M Window

## 5.1. Outline of PDB38M Window

The PDB38M window is the main window of PDB38M. It appears on the screen when you start up PDB38M. It is from this window that you can use PDB38M's various debug functions to, for example, call up various other windows, execute the target program, and set breakpoints.

The following lists the functions of the PDB38M window:

- Window control
  The PDB38M window controls various other windows such as the program and register windows.
- Emulator control
  - Downloading the target program
  - Executing and halting the target program
  - Single-stepping the target program
  - Setting and clearing breakpoints

## 5.2. Structure of PDB38M Window

Figure 5-1 shows the structure of the PDB38M window.

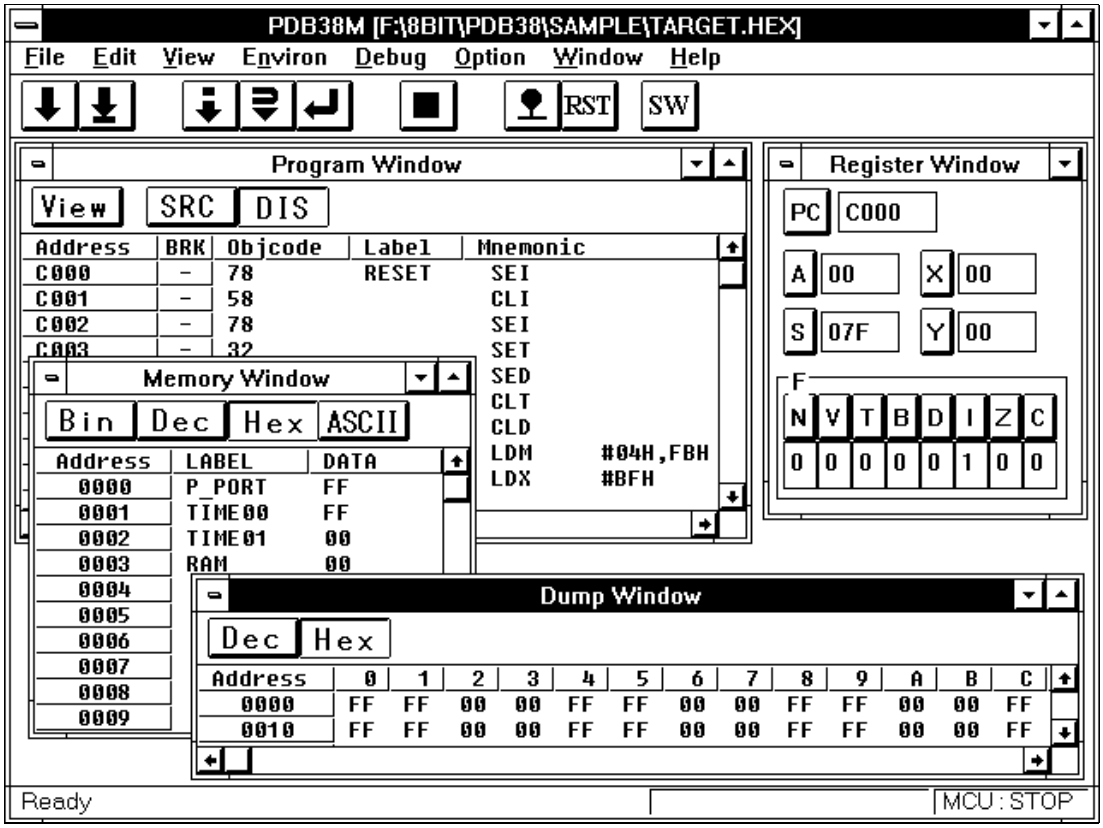



Figure 5-1 Structure of the PDB38M window

Each part of the PDB38M window are detailed in the next pages.

### 5.2.1. Menus

The PDB38M window has menus necessary to issue commands. Figure 5-2 shows an example of menus displayed in the PDB38M window.

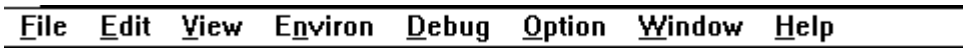



Figure 5-2 Example of menu display

The PDB38M window menus can be classified into the basic menus and extension menus.

**"Basic menus" vs. "Extension menus"**

Among the PDB38M menu items, the Option menu has its submenu items automatically switched over according to the active window displayed in the PDB38M window's main display area. In PDB38M, this Option menu is referred to as an "extension menu." Conversely, all other menus remain the same and do not change even when the active window switches from one to another. These are called the "basic menus." The basic menus provide menus required for the basic operation of PDB38M, as well as to perform debug operations.

- For details about the function and usage of each basic menu item, refer to "Using Basic Menus" in this chapter.
- For details about the function and usage of each extension menu item, refer to "Using Extension Menu" in pages where each window's operating method is explained.

### 5.2.2. Tool Bar

PDB38M has a tool bar to help you execute frequently used commands easily. Figure 5-3 shows an example of a tool bar displayed in the PDB38M window.


Figure 5-3 Example of tool bar displayed in the PDB38M window

For details about the function of each tool bar button, refer to Section 5.4, "Tool Bar Composition."

### 5.2.3. Main Display Area

The main display area is where various windows started up from the PDB38M window are displayed. None of these windows can be placed outside this area.

### 5.2.4. Various Windows

This refers to the windows that are called up from PDB38M's Window menu. These windows are displayed in the main display area.

### 5.2.5. Status Bar

The status bar displays information to assist you in operating PDB38M and the target's execution status.
Figure 5-4 shows an example of a status bar displayed on the screen.

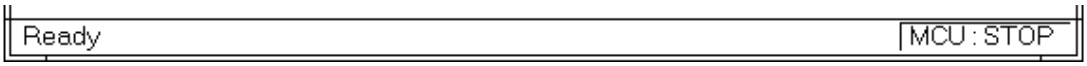



Figure 5-4 Example of status bar displayed on the screen

The following explains the contents of information displayed in each area of the status bar:

a. Explanation display area
This area displays an explanation about window operation and about menus.

b. Execution status display area
This area displays the target's current execution status.
When the target is executing (including execution of Come, Step, Over, and Return), a message "RUN" is displayed on the screen. When the target is idle, a message "STOP" is displayed.

## 5.3. Using Basic Menu

Table 5-1 and Table 5-2 show the menu structure of the PDB38M window.

Table 5-1 PDB38M Window's Menu Structure (1)

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| File | Download | Downloads the target program. | |
| | Load Module... | Machine language data and debug information are downloaded. | Shift + F.1 |
| | Memory Image... | Only machine language data is downloaded. | |
| | Symbol... | Only debug information are downloaded. | |
| | Upload... | Uploads the target program. | |
| | Save Disasm... | Saves a disassembled result. | |
| | Exit | Terminates PDB38M. | |
| Edit | Copy | Copies a selected character string to the clipboard. | Ctrl + C |
| | Paste | Pastes a character string from the clipboard into position. | Ctrl + V |
| | Find... | Searches for a character string. | |
| View | Tool Bar | Turns on or off the tool bar display. | |
| | Status Bar | Turns on or off the status bar display. | |
| Environ | Init... | Sets environment variables. | |
| | Path... | Sets the source file's search path. | |
| | Start Up... | Sets the startup functions. | |
| Debug | Go | Executes the target program. | |
| | Go | Executed beginning with the current PC. | F.1 |
| | Go Option... | Executed beginning with a specified address. | |
| | Come | Executes the target program up to the cursor position. | F.2 |
| | Step | Single-steps the target program. | |
| | Step | Single-stepped one time. | F.3 |
| | Step Option... | Single-stepped a specified number of times. | |
| | Over | Oversteps the target program. | |
| | Over | Overstepped one time. | F.4 |
| | Over Option... | Overstepped a specified number of times. | |

| | | | |
|---|---|---|---|
| | Return | Executes the target program until the current subroutine returns. | F.5 |
| | Break Point | Sets a breakpoint. | |
| | S/W Break Point... | Opens a software breakpoint setup dialog box. | F.7 |
| | Break | Sets or clears a software break at the cursor position. | |
| | Reset | Resets the target system. | F.8 |

Table 5-2 PDB38M Window's Menu Structure (2)

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Debug | Stop | Stops executing the target program. | |
| Option | ☞ | (Extension menus specific to each window are added.) | |
| Window | Cascade | Displays windows one on top of another. | |
| | Tile | Displays windows side by side. | |
| | Arrange Icon | Lines up icons. | |
| | Program Window | Activates the program window. | |
| | Source Window | Opens the source window. | |
| | Register Window | Opens the register window. | |
| | Memory Window | Opens the memory window. | |
| | Dump Window | Opens the dump window. | |
| | ASM Watch Window | Opens the watch window. | |
| | Script Window | Opens the script window. | |
| Help | About... | Displays version information. | |

The following pages explain how to use the basic menus.

☞ For details about the Option menu's extension menu items, refer to the table of extension menus in pages where each window's operating method is explained.

### 5.3.1. File Menu

The File menu is assigned menu commands for reading a file, saving a file, and terminating PDB38M. The following explains the function of each menu command assigned under the File menu.

❒ **Download**

This menu item has commands to download machine language data and debug information from the computer to the emulator.
When you choose **Download**, the following three submenu commands appear. So choose your desired submenu command.

Load Module

This submenu command downloads both machine language data and debug information.
In normal download operation, use this submenu.

Memory Image

This submenu command downloads only machine language data.
Use this submenu command when you want to re-download a machine language data file you have saved using the Upload command (menu [File] → [Upload]). In this case, note that labels, symbols, and source file cannot be displayed.

Symbol

This submenu command downloads only debug information.

When you choose one of the three submenus, a file selection dialog box pops up. Figure 5-5 shows the structure of this dialog box. When this dialog box appears, choose the file name you want to be downloaded.

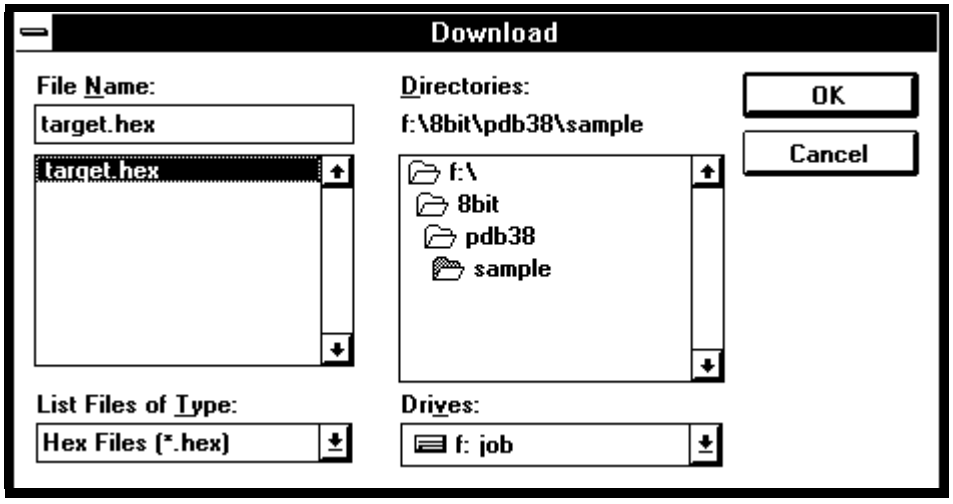



Figure 5-5 File selection dialog box

- In the Load Module submenu's dialog box, choose an Intel HEX Format File name (whose extension is ".hex").
- In the Memory Image submenu's dialog box, choose an Intel HEX Format File name (whose extension is ".hex").
- In the Symbol submenu's dialog box, choose a Symbol File name (whose extension is ".sym").
- When you click on the OK button, PDB38M starts downloading your specified file. When downloading a file, it brings up a dialog box to show a message as in Figure 5-6. If you click on the Cancel button in this dialog box, PDB38M stops downloading a file. When PDB38M finishes downloading a file, this dialog box is automatically closed.

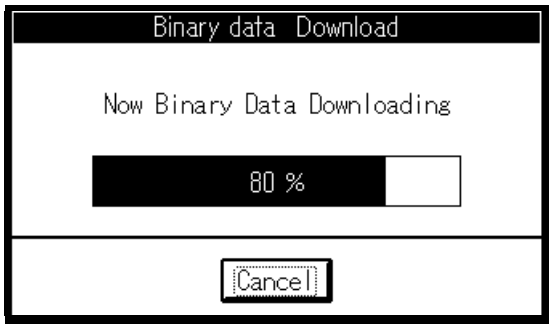



Figure 5-6 Example of dialog box displayed when downloading a file

***★★Precaution★★***

If you start downloading a file while the source window is displayed in the PDB38M window's main display area, it is only this source window that is closed when PDB38M finishes downloading the file.

## ❒ Upload

This menu command saves the program's memory contents to a machine language data file (i.e., uploading from emulator to computer).
When you choose **Upload**, an Upload dialog box pops up. Figure 5-7 shows the structure of this dialog box. When this dialog box appears, specify a file name in which you want memory contents to be saved and the start and end addresses of the memory area you want to be saved.

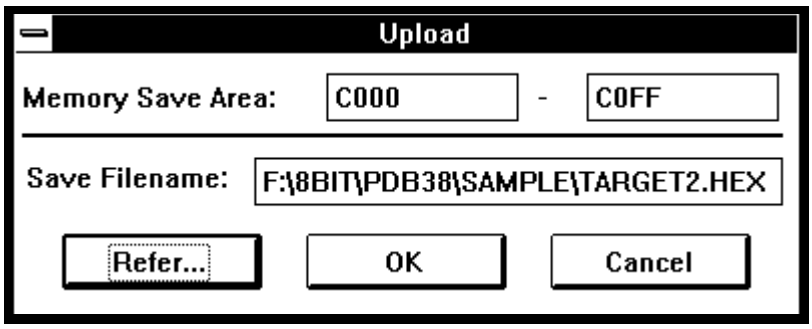



Figure 5-7 Upload dialog box

- Input the start and end addresses of the memory area you want to be saved in the **Memory Save Area:** field. Input the file name in which you want memory contents to be saved in the **Save Filename:** field.
- Expressions can be used in writing the start and end addresses of the memory area. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal.
- When specifying a file name, add a path name.
- For the file name input here, specify ".hex" as its file attribute. An Intel HEX Format File will be generated.
- If you specify an existing file name here, the existing file will be written over.

When you click on the **Refer** button, a file selection dialog box pops up. You can use this dialog box to specify a file name, instead of entering it from the keyboard. Figure 5-8 shows the structure of this dialog box.

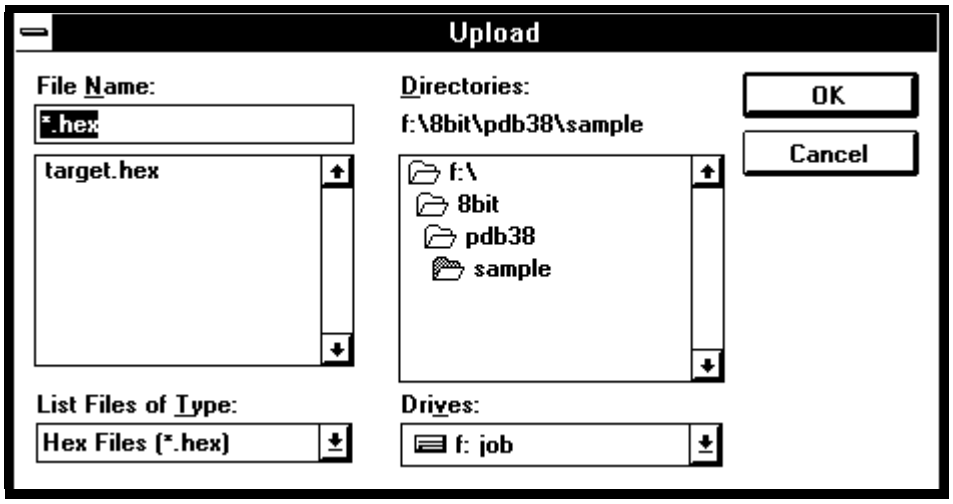



Figure 5-8 File selection dialog box

- When you click on the **OK** button, PDB38M starts saving memory contents to your specified file (uploading). PDB38M brings up a dialog box like the one shown in Figure 5-9 when it is uploading. This dialog box is automatically closed when PDB38M finishes uploading.

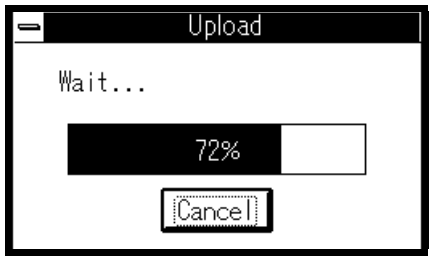



Figure 5-9 Example of dialog box displayed during uploading

- If the range of memory locations to be saved is large, PDB38M requires a large disk space and a long time. In such a case, you can stop saving memory contents by clicking on the **Cancel** button in the dialog box shown in Figure 5-9. In this case, a file is created for the memory contents immediately before you have stopped saving.

### ❒ Save Disasm

This menu command saves the program memory's disassembled result to a text file as an disassembled file. This disassembled file is a text file for reference purposes only and cannot be reassembled or re-downloaded.
When you choose **Save Disasm**, a Save dialog box pops up. Figure 5-10 shows the structure of this dialog box. When this dialog box appears, specify a file name in which you want memory contents to be saved and the start and end addresses of the memory area you want to be saved.

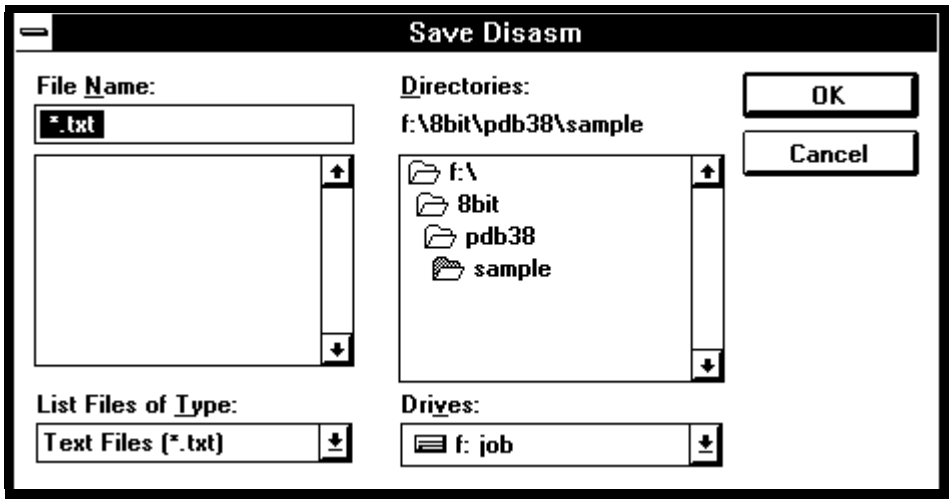



Figure 5-10 Save dialog box

- Input the start and end addresses of the memory area you want to be saved in the Memory Save Area: field. Input the file name in which you want memory contents to be saved in the Save Filename: field.
- Expressions can be used in writing the start and end addresses of the memory area. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal.
- When specifying a file name, add a path name. For this file name, you can specify any desired file name and file attribute.
- If you specify an existing file name here, the existing file will be written over.
  When you click on the **Refer** button, a file selection dialog box pops up. You can use this dialog box to specify a file name, instead of entering it from the keyboard. Figure 5-11 shows the structure of this dialog box.

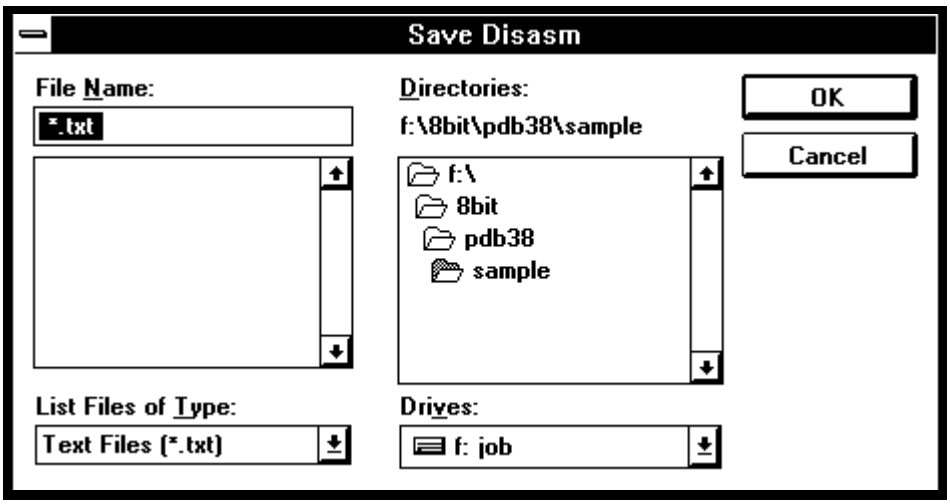



Figure 5-11 File selection dialog box

- When you click on the OK button, PDB38M starts saving the disassembled result to your specified file. PDB38M brings up a dialog box like the one shown in Figure 5-12 when it is saving. This dialog box is automatically closed when PDB38M finishes saving.

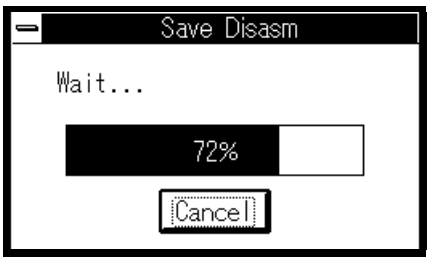



Figure 5-12 Example of dialog box displayed when saving disassembled result

- If the range of memory locations to be saved is large, PDB38M requires a large disk space and a long time. In such a case, you can stop saving memory contents by clicking on the Cancel button in the dialog box shown in Figure 5-12. In this case, a file is created for the memory contents immediately before you have stopped saving.

❒ **Exit**

This menu command quits PDB38M.

When you choose **Exit**, a confirmation dialog box pops up, requesting your confirmation of whether you are sure you want to quit. When this dialog box appears, click on the **OK** button. If you click on the **Cancel** button here, PDB38M is not terminated. Figure 5-13 shows the structure of the confirmation dialog box.


Figure 5-13 Confirmation dialog box

### 5.3.2. Edit menu

The Edit menu is assigned menu commands for copying, pasting, and searching for a character string.

The following explains the function of each menu command assigned under the Edit menu.

❒ **Copy**

This menu command copies a character string you have chosen with the mouse to the clipboard.

❒ **Paste**

This menu command pastes a character from the clipboard into the script window.

❒ **Find**

This menu command searches for a character string you have specified from the source file that is displayed in both program and source windows.
When you choose **Find**, a Find dialog box pops up. Figure 5-14 shows the structure of this dialog box. When this dialog box appears, specify a character string you want to be searched and the direction of search.

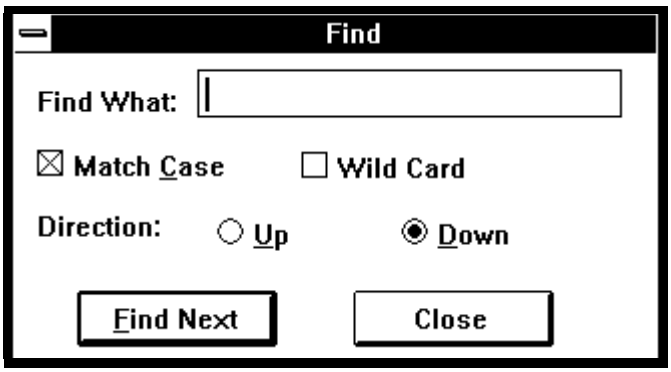



Figure 5-14 Find dialog box

- Input the character string you want to be searched in the Find What: field.
- Use the Match Case check box to specify whether you want the character string to be distinguished between uppercase and lowercase letters. When you turn on this check box, the character string is distinguished between uppercase and lowercase letters as it is searched. If you specify "ABC" as the character string to be searched, for example, "abc," "AbC," etc. are excluded from the search. If you want "abc," "AbC," etc. to be included in the search, turn off the check box.
- Use the Wild Card check box to specify whether or not you use wild cards. When this check box is turned on, the character string can be searched using a wild card you specify. The wild cards that can be used here are an asterisk (*) and a question mark (?).

| Wild Card | Explanation |
|---|---|
| * | Indicates an arbitrary character string (more than 0 character). |
| ? | Indicates one arbitrary character. |

If you want to use these special characters as ordinary characters in a character string to be searched, add the character '¥' before the special

character.
If do not use any wild card, turn off the Wild Card check box.

- To specify the direction of search, use the **Up** radio button (searched upward from the current cursor position toward the beginning) or **Down** radio button (searched downward from the current cursor position toward the end) in the **Direction:** field.
- To specify the search start position (cursor position), point to somewhere of the program display area in the program or source window and click the mouse button. Figure 5-15 shows an example of the cursor position in the program display area.

```
Line  | BRK | Source
00050 |  -  | RESET:                ;
00051 |  -  |    SEI              ;INTERRUPT DISABLE
00052 |  -  |    CLI              ;
00053 |  -  |    SEI              ;INTERRUPT DISABLE
00054 |  -  |    SET              ;INTERRUPT DISABLE
00055 |  -  |    SED              ;INTERRUPT DISABLE
00056 |  -  |    CLT              ;
00057 |  -  |    CLD              ;
00058 |  -  |    LDM #04H,CPUMOD  ;CPU MODE REGISTER
00059 |  -  |    LDX #0BFH        ;STACK POINT SET
```

Figure 5-15 Example of cursor position in program display area

If you do not specify any cursor position, PDB38M assumes the first line of the file is the search start position.

- When you click on the **Find Next** button, PDB38M starts searching sequentially in the specified direction.

### 5.3.3. View Menu

The View menu is assigned menu commands for turning on or off the tool bar and status bar display. The following explains the function of each menu command assigned under the View menu.

❒ Tool Bar

This menu command turns on or off the tool bar display of the active window in the main display area.

Note that the tool bar of the PDB38M window cannot be turned on or off.

❒ Status Bar

This menu command turns on or off the status bar display of the PDB38M window.

### 5.3.4. Environ Menu

The Environ menu is assigned menu commands for setting up PDB38M's operating environment. The following explains the function of each menu command assigned under the Environ menu.

❒ Init

This menu command sets up PDB38M's operating environment.
The Init command allows you to set the following environment variables:

- Setting of definition file
  Specify the definition file that corresponds to the target MCU.
- Setting of script file executed at startup
  Specify the script file that is executed when PDB38M starts up. Once this setting is made, PDB38M opens a script window as it starts up and executes the specified script file.
- Setting of serial communication interface
  Set parameters associated with serial communication.
- Setting of default tab value
  Specify the tab value you want to be taken for source display in the program and source windows when PDB38M starts up.
- Setting of default font and size
  Specify the font and fond size you want to be used for display of each window when PDB38M starts up.

When you choose **Init**, an Init dialog box pops up. Figure 5-16 shows the structure of this dialog box. When this dialog box appears, set each operating environment variable.

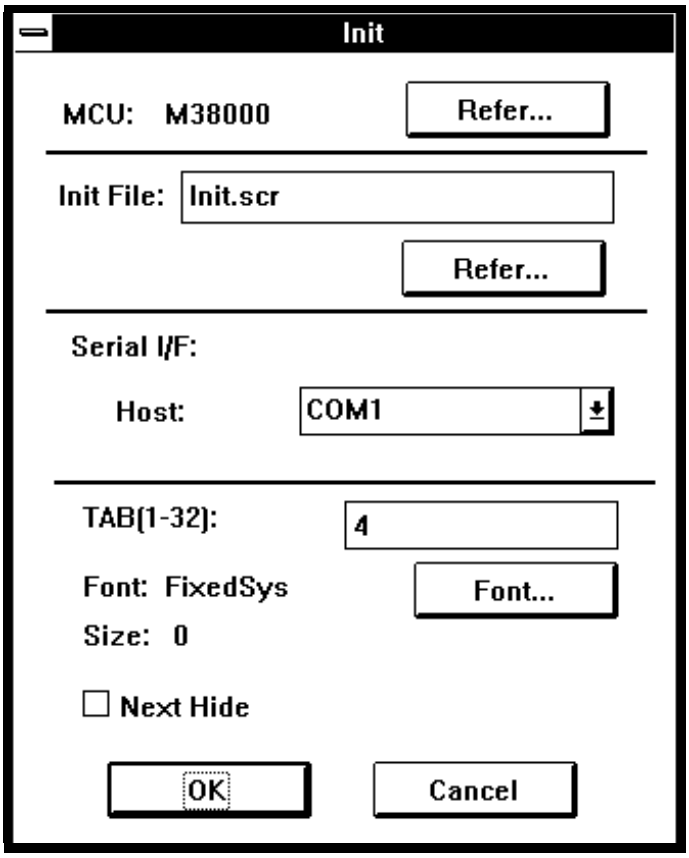



Figure 5-16 Init dialog box

- This is the same Init dialog box that appears when PDB38M starts up. For details on how to use the Init dialog box, refer to "Using Init dialog box" in Part I: Setting Up .
- If you open the Init dialog box and set environment variables using this menu command, the contents you have set become effective when you start up PDB38M next time. However, the changes made to the tag value and font size become effective immediately after you click on the **OK** button.

## ❒ Path

This menu command specifies a source file's search path name. If the source file of the program to be debugged is not found in the current directory or stored separately in two or more subdirectories, you must specify a search path name. Multiple search path names can be specified.
When displaying the source file in the program and source windows, PDB38M

searches the directories in order of priorities as follows:

1. Path names included in debug information
2. Path names specified by Path (in the order they are specified)
3. Current directory

When you choose **Path**, a Path dialog box pops up. Figure 5-17 shows the structure of this dialog box.

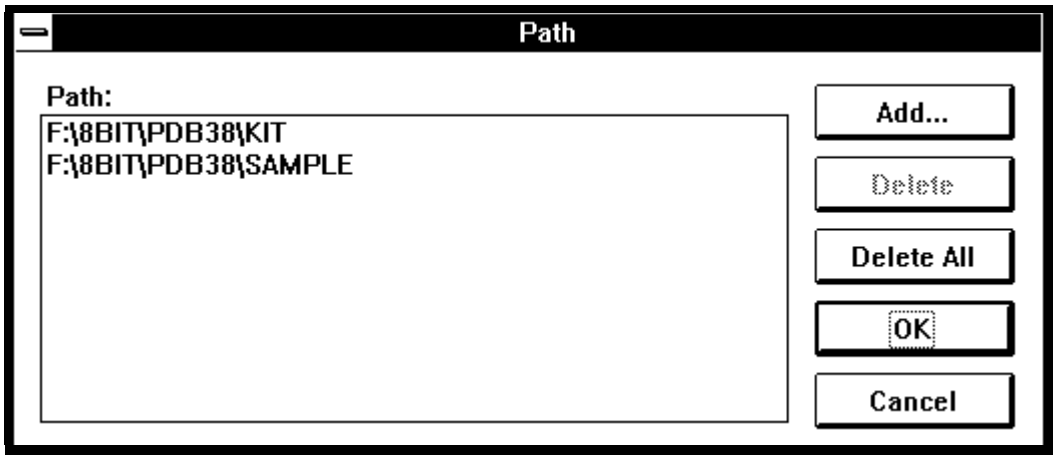



Figure 5-17 Path dialog box

- The Path: field lists all search path names currently set.
- To add a search path, click on the Add... button.
  When you click on this button, a file selection dialog box pops up. When this dialog box appears, choose your desired file name. The directory where your specified file exists is added to the search paths.
  Figure 5-18 shows the file selection dialog box.

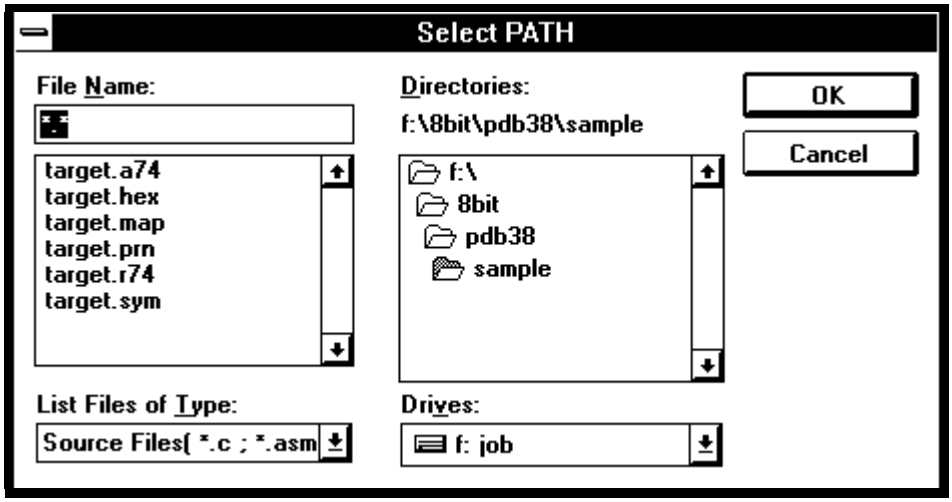



Figure 5-18 File selection dialog box

- To delete a search path, point to one of the search path names displayed in the Path: field and click the mouse button. Then click on the Delete button to delete the selected search path.
- If you want to delete all search paths, click on the Delete All button. When you click on this button, a confirmation dialog box pops up, requesting your confirmation of whether you are sure you want to delete all. When this dialog box appears, click on the OK button. If you click on the Cancel button here, no search path is deleted. Figure 5-19 shows the structure of the confirmation dialog box.

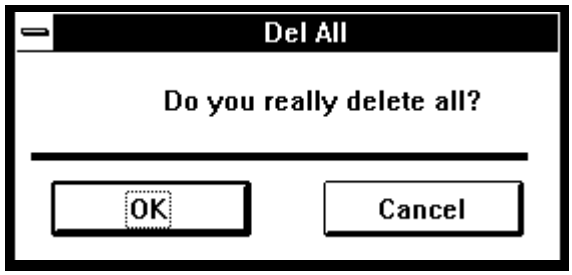



Figure 5-19 Confirmation dialog box

### ❒ StartUp

This menu command sets the position from which PDB38M starts debugging a program.

If there is no source line whose address corresponds to the PC value immediately after the target program is downloaded (e.g., if the module where the startup program is written does not have debug information), PDB38M displays the source line in the program window's program display area that corresponds to the label indicated by this StartUp menu command.

However, even when the source line specified by the StartUp menu command is displayed in the program window, the PC value remains the same as the initial PC value immediately after the target program is downloaded. In this case, if you want the PC value to be incremented to match the source line being displayed, execute Come. (Refer to the explanation of menu [Debug] → [Come].)

***★★Precaution★★***

If a source line whose address corresponds to the PC value immediately after the target program is downloaded does exist, settings made by this StartUp menu command are ignored.

When you choose **StartUp**, a StartUp dialog box pops up. Figure 5-20 shows the structure of this dialog box. When this dialog box appears, set a label or symbol that specifies the position from which PDB38M starts debugging a program.

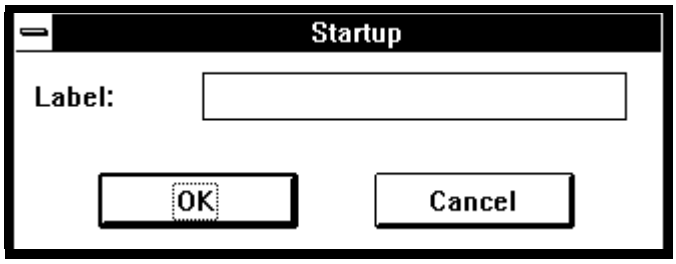



Figure 5-20 StartUp dialog box

- Input the label or symbol in the **Label:** field.
- Once the debug start position is set in the StartUp dialog box, the information is saved to the environment setup file. The content of this information also becomes effective when you start up PDB38M from next time on.

### 5.3.5. Debug Menu

The Debug menu is assigned menu commands for executing or stopping the target program, single-stepping through instructions, and for performing other debug-related operations. The following explains the function of each menu command assigned under the Debug menu.

❒ Go

This menu item has commands to execute the target program.
When you choose **Go**, the following submenu commands are displayed. So choose your desired submenu command.

Go

This submenu command executes the target program beginning with the current program counter.

Go Option

This submenu command executes the target program beginning with a specified address.

When you choose **Go Option**, a Go dialog box pops up. Figure 5-21 shows the structure of this dialog box. When this dialog box appears, set the position at which the target program starts executing.

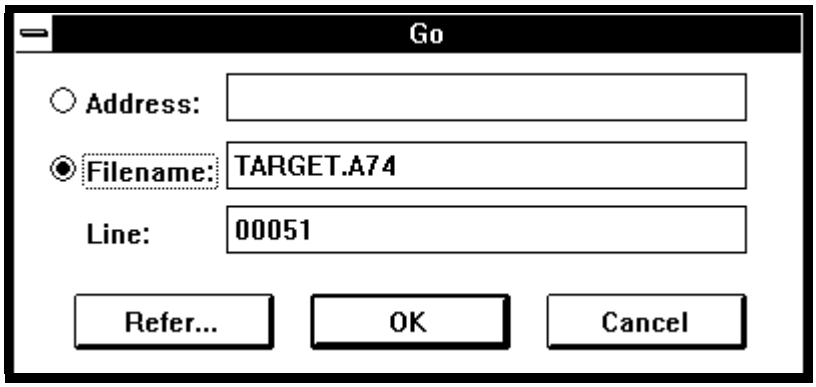



Figure 5-21 Go dialog box

- An address or a source file name plus line number can be used to specify the position at which the target program starts executing.
- When using an address to specify the target program's execution start position, click on the **Address** radio button and input your desired address in the **Address:** field.
- When using a source file name plus line number to specify the target program's execution start position, click on the **Filename** radio button and input your desired source file name in the **Filename:** field and a line number in the **Line:** field.
- When you click on the **Refer** button, a file selection dialog box pops up. You can use this dialog box to specify a file name, instead of entering it from the keyboard. Figure 5-22 shows the structure of this dialog box.

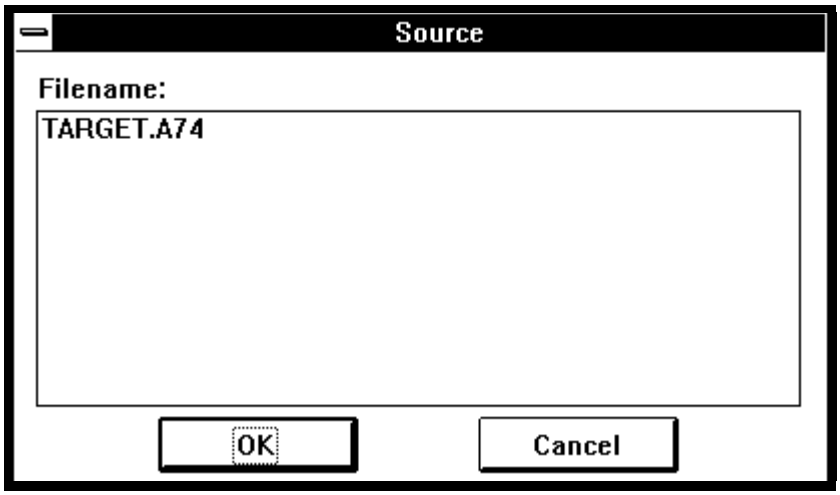



Figure 5-22 File selection dialog box

- For the line number, specify a line number with address information added.
- Expressions can be used in writing the address and line number here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants for addresses written here must be hexadecimal, and that the radix of constants for line numbers must be decimal.

The target program executed by the Go menu command stops running when a breakpoint is reached or you execute a stop command (e.g., menu [Debug] → [Stop]).

### ❒ Come

This menu command executes the target program from the current program counter to the cursor position in the program or source window's program display area. The program stops running when the program counter reaches the cursor position. However, if the cursor position is specified at a line where software breakpoints cannot be set (e.g., a comment line or data definition line), you cannot execute Come.
The currently set breakpoints are effective. Therefore, if break condition is met before reaching the cursor position, the program stops running at that point in time. If you want to halt the program in the middle of execution, execute a stop command (e.g., menu [Debug] → [Stop]).

To specify a cursor position, point to somewhere of the program or source window's program display area and click the mouse button. Figure 5-23 shows an example of the cursor position in the program display area.

```
Line  | BRK | Source
00050 |  -  | RESET:              ;
00051 |  -  |    SEI          ;INTERRUPT DISABLE
00052 |  -  |    CLI          ;
00053 |  -  |    SEI          ;INTERRUPT DISABLE
00054 |  -  |    SET          ;INTERRUPT DISABLE
00055 |  -  |    SED          ;INTERRUPT DISABLE
00056 |  -  |    CLT          ;
00057 |  -  |    CLD          ;
00058 |  -  |    LDM #04H,CPUMOD ;CPU MODE REGISTER
00059 |  -  |    LDX #0BFH       ;STACK POINT SET
```

Figure 5-23 Example of cursor position in program display area

## ❒ Step

This menu item has commands to single-step the target program by executing one instruction or line at a time.
When the source program is displayed in the program window, the program is executed one source line at a time. When the disassembled result is displayed, the program is executed one instruction at a time.

If a subroutine call instruction is encountered when single-stepping, the program is single-stepped after entering the subroutine.

When you choose **Step**, the following submenu commands are displayed. So choose your desired submenu command.

Step

This submenu command single-steps the target program one time.

Step Option

This submenu command single-steps the target program a specified number of times.
When you choose **Step Option**, a Step dialog box pops up. Figure 5-24 shows the structure of this dialog box. When this dialog box appears, set a step count or a number of times you want the target program to be single-stepped.

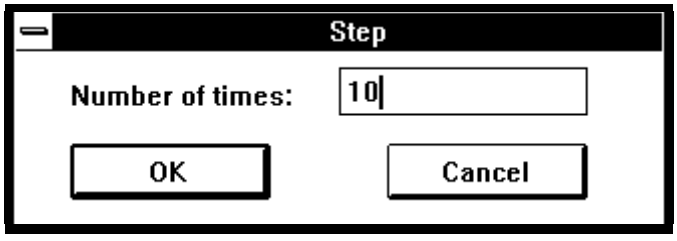



Figure 5-24 Step dialog box

- Input your desired step count in the **Number of times:** field.
- Expressions can be used to specify this step count. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be decimal.

To stop single-stepping the program in the middle, execute a stop command (e.g., menu [Debug] → [Stop]).

### ❒ Over

This menu item has commands to overstep the target program by executing one instruction or line at a time.

When the source program is displayed in the program window, the program is executed one source line at a time. When the disassembled result is displayed, the program is executed one instruction at a time. The difference with Step is that if a subroutine call instruction is encountered when overstepping, a program section from this instruction to the next is executed as one step without single-stepping the body of the subroutine.

When you choose **Over**, the following submenu commands are displayed. So choose your desired submenu command.

Over

This submenu command oversteps the target program one time.

Over Option

This submenu command oversteps the target program a specified number of times.
When you choose **Over Option**, an Over dialog box pops up. Figure 5-25 shows the structure of this dialog box. When this dialog box appears, set an overstep count or a number of times you want the target program to be overstepped.

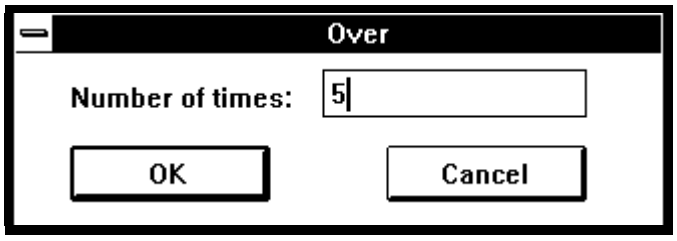



Figure 5-25 Over dialog box

- Input your desired overstep count in the **Number of times:** field.
- Expressions can be used to specify this overstep count. For details about the format of an expression, refer to "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be decimal.

To stop overstepping the program in the middle, execute a stop command (e.g., menu [Debug] → [Stop]).

### ❒ Return

This menu command returns from the current subroutine. Until control returns from the current subroutine to the position from which it has been called, the program is single-stepped repeatedly one line or instruction after another.
When the source program is displayed in the program window, control is returned one source line at a time. When the disassembled result is displayed, control is returned one instruction at a time.

To stop returning in the middle, execute a stop command (e.g., menu [Debug] → [Stop]).

### ❒ Break Point

This menu item has commands to set hardware and software breakpoints.
When you choose **Break Point**, the following submenu commands are displayed.

S/W Break Point

This submenu command opens a software breakpoint setup dialog box.

This dialog box allows you to perform the following operations:

- Referencing software breakpoints
- Setting software breakpoints
- Clearing software breakpoints
- Enabling software breakpoints
- Disabling software breakpoints

A software breakpoint means a breakpoint that causes program execution to break before executing the instruction at an address specified by the breakpoint.

When you choose **S/W Break Point**, a S/W breakpoint setup dialog box pops up. Figure 5-26 shows the structure of this dialog box.

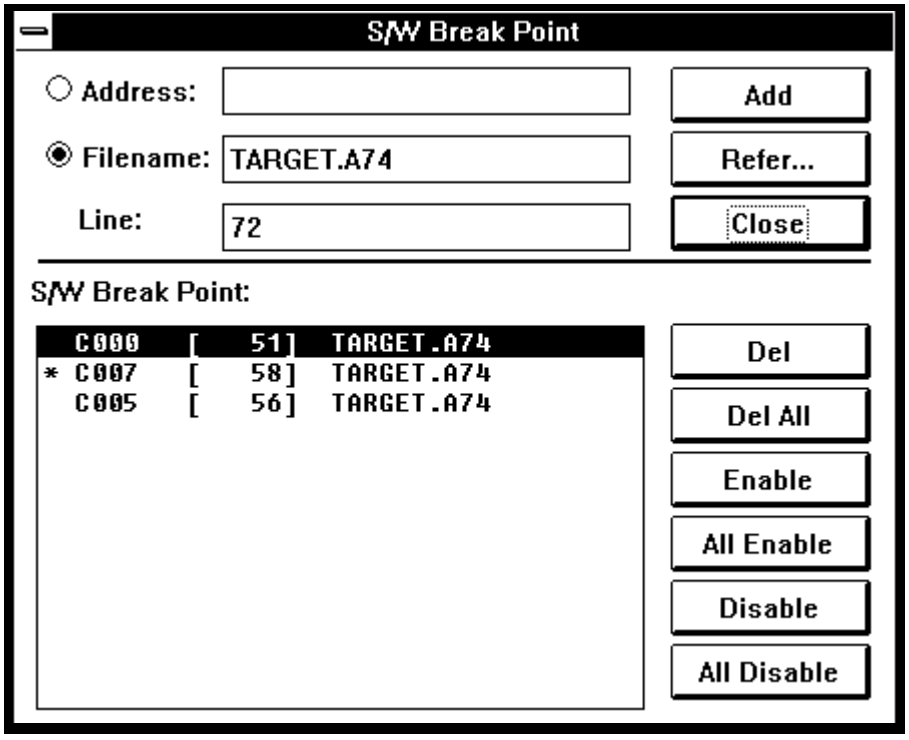
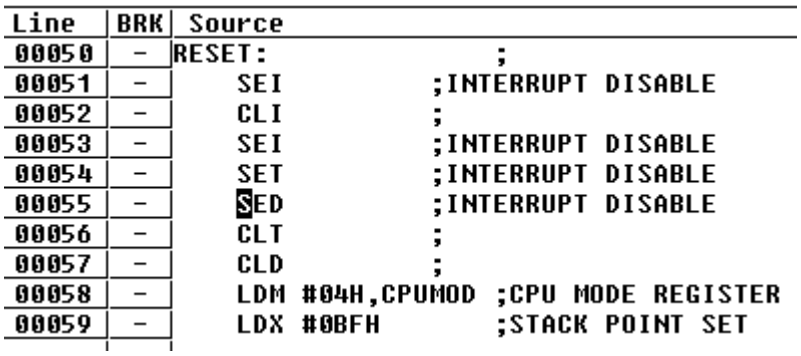



Figure 5-26 S/W breakpoint setup dialog box

- For details on how to use this dialog box, refer to Section 13, "Using S/W Breakpoint Setup Dialog Box.

Break

This submenu command sets or clears a software breakpoint at the cursor position in the program or source window's program display area.
To specify a cursor position, point to somewhere of the program or source window's program display area and click the mouse button. Figure 5-27 shows an example of the cursor position in the program display area.



| Line | BRK | Source |
|---|---|---|
| 00050 | - | RESET: ; |
| 00051 | - | SEI ;INTERRUPT DISABLE |
| 00052 | - | CLI ; |
| 00053 | - | SEI ;INTERRUPT DISABLE |
| 00054 | - | SET ;INTERRUPT DISABLE |
| 00055 | - | SED ;INTERRUPT DISABLE |
| 00056 | - | CLT ; |
| 00057 | - | CLD ; |
| 00058 | - | LDM #04H,CPUMOD ;CPU MODE REGISTER |
| 00059 | - | LDX #0BFH ;STACK POINT SET |

Figure 5-27 Example of cursor position in program display area

- Move the cursor to a line where no software breakpoint is set and choose this submenu command to set a software breakpoint in the line.
- When you move the cursor to a line where some software breakpoint is set and choose this submenu command, the currently set software breakpoint is cleared.
- If you move the cursor to a line where no software breakpoint can be set (e.g., a comment line or data definition line), PDB38M disables this submenu command from being selected.

❒ Reset

This menu command resets the target system.

❒ Stop

This menu command stops executing the target program.

### 5.3.6. Option Menu

The Option menu is assigned menu commands (extension menu) that are specific to the active window displayed in the PDB38M window's main display area. When the program window is active, for example, this menu is assigned menu commands specific to the program window. When the source window is active, this menu is assigned menu commands specific to the source window.

For details about each menu command assigned under the Option menu, refer to "Using Extension Menu" in pages where each window's operating method is explained.

### 5.3.7. Window Menu

The Window menu is assigned menu commands for changing the display form of PDB38M's main display area. The following explains the function of each menu command assigned under the Window menu.

❒ Cascade

This menu command places windows one on top of another as they are displayed in the main display area.

❒ Tile

This menu command places windows side by side as they are displayed in the main display area.

❒ Arrange Icon

This menu command lines up icons as they are displayed in the main display area.

❒ Program Window

This menu command activates the program window.
For details on how to use the program window, refer to Section 6, "Using Program Window" in Part II: Window Operation .

❒ Source Window

This menu command opens the source window in the main display area.
For details on how to use the source window, refer to Section 7, "Using Source Window" in Part II: Window Operation .

❒ Register Window

This menu command opens the register window in the main display area. If the register window is already open, this command activates the register window.
For details on how to use the register window, refer to Section 8, "Using Register Window" in Part II: Window Operation .

❒ Memory Window

This menu command opens the memory window in the main display area.
For details on how to use the memory window, refer to Section 9, "Using Memory Window" in Part II: Window Operation .

❒ Dump Window

This menu command opens the dump window in the main display area.
For details on how to use the dump window, refer to Section 10, "Using Dump Window" in Part II: Window Operation .

❒ ASM Watch Window

This menu command opens the watch window (assembler level watch window) in the main display area. If the watch window is already open, this command activates the watch window.
For details on how to use the watch window, refer to Section 11, "Using Watch Window" in Part II: Window Operation .

❒ Script Window

This menu command opens the script window in the main display area. If the script window is already open, this command activates the script window.
For details on how to use the script window, refer to Section 12, "Using Script Window" in Part II: Window Operation .

### 5.3.8. Help Menu

The Help menu is assigned menu commands for displaying information on how to use PDB38M and about version numbers. The following explains the function of each menu command assigned under the Help menu.

❒ Index

This menu command opens a help window, displaying a table of contents of help files.
A help window is used to reference PDB38M's help files.

❒ About

This menu command displays the monitor version of PDB38M and that of the Emurator Main Unit(M38000T-SBI).
When you choose **About**, an About dialog box pops up. Figure 5-28 shows the structure of this dialog box.

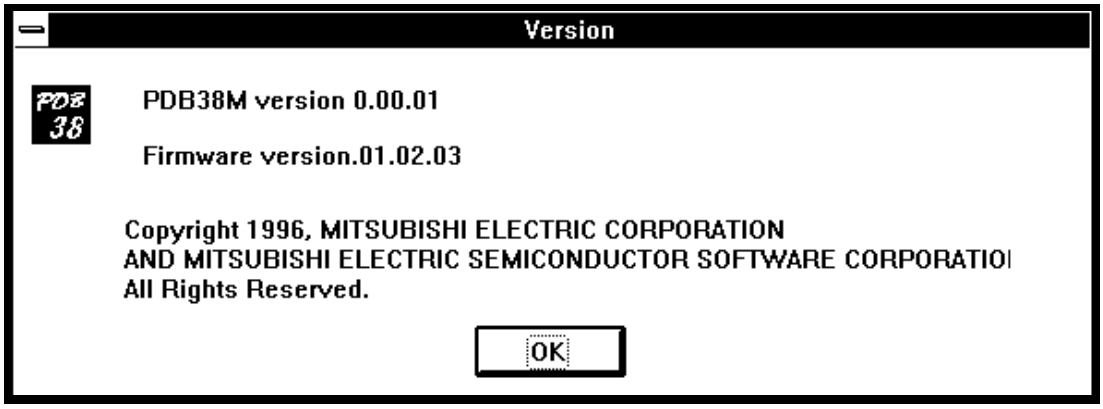



Figure 5-28 About dialog box

## 5.4. Tool Bar Composition

The PDB38M window's tool bar is comprised of the buttons shown in Figure 5-29.


Figure 5-29 Composition of PDB38M window tool bar

The following explains how to use each button.

❍ Go button

This button executes the target program beginning with the current program counter.
The operation performed by this button is the same as you choose menu [Debug] → [Go] → [Go].

❍ Come button

This button executes the target program from the current program counter to the cursor position in the program display area.
The operation performed by this button is the same as you choose menu [Debug] → [Come].

❍ Step button

This button single-steps the target program.
The operation performed by this button is the same as you choose menu [Debug] → [Step] → [Step].

❍ Over button

This button oversteps the target program.
The operation performed by this button is the same as you choose menu [Debug] → [Over] → [Over].

❍ Return button

This button returns from the current subroutine.
The operation performed by this button is the same as you choose menu [Debug] → [Return].

❍ Stop button

This button stops executing the target program.
The operation performed by this button is the same as you choose menu [Debug] → [Stop].

❍ Break button

This button sets or clears a software breakpoint at the cursor position in the program display area.
The operation performed by this button is the same as you choose menu [Debug] →

[Break Point] → [Break].

❍ **Reset button**

This button resets the target system.
The operation performed by this button is the same as you choose menu [Debug] → [Reset].

❍ **S/W button**

This button opens a S/W breakpoint setup dialog box.

The operation performed by this button is the same as you choose menu [Debug] → [Break Point] → [S/W Break Point].

# 6. Using Program Window

## 6.1. Outline of Program Window

The program window displays the program whose address corresponds to the current program counter. This window opens automatically when you start up PDB38M. It allows you to execute the target program up to the cursor position, set or clear software breakpoints in the program window, and display the program whose address corresponds to the current program counter. To set or clear software breakpoints here, point to the address display area or line number display area and double-click the mouse button.

The following explains the functions of the program window:

1. Displaying the program whose address corresponds to the current program counter
   - The program is displayed in source display or disassemble display mode.
   - The line corresponding to the program counter is marked by a yellow line.
   - A line that has a software breakpoint set is marked by the letter '**B**' in the breakpoint display area.
   - Line number and address are displayed for each line of the source file. (By default, only the line numbers are displayed.)
   - Address and object code are displayed for each line of disassemble display.
   - The display is updated as the program counter is incremented after executing a command.
   - The source file name being displayed is shown in the program window's title bar.
2. Setting and clearing software breakpoints

## 6.2. Structure of Program Window

The program window has two display modes: source and disassemble. Figure 6-1 shows the structure of the program window in source display mode. Figure 6-2 shows the structure of the program window in disassemble display mode.

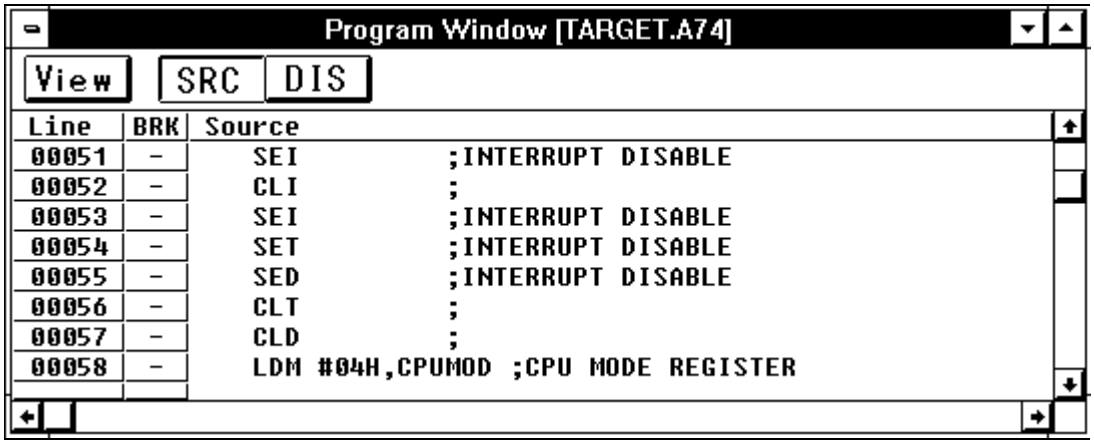



Figure 6-1 Structure of program window in source display mode

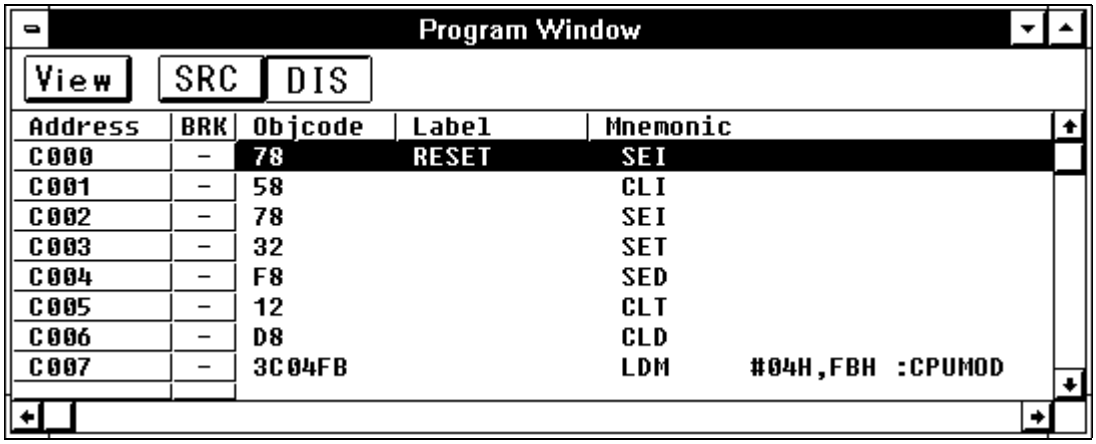



Figure 6-2 Structure of program window in disassemble display mode

Each part of the program window is explained below.

### 6.2.1. Tool Bar

The program window has buttons to change the display area and switch between two display modes.

For details about the function of each button, refer to Section 6.6, "Tool Bar Composition."

### 6.2.2. Program Display Area

This area is where a program is displayed.
In source display mode, the contents of the source file are displayed here. In disassemble display mode, the disassembled results of the source program (Object code, Label, and Mnemonic) are displayed here. The line corresponding to the current program counter is marked by a yellow line. Figure 6-3 shows an example of program display with the program counter indicated.

| Address | BRK | Objcode | Label | Mnemonic |
|---|---|---|---|---|
| C000 | - | 78 | RESET | SEI |
| C001 | - | 58 | | CLI |
| C002 | - | 78 | | SEI |
| C003 | - | 32 | | SET |
| C004 | - | F8 | | SED |
| C005 | - | 12 | | CLT |
| C006 | - | D8 | | CLD |
| C007 | - | 3C04FB | | LDM #04H,FBH :CPUMOD |

Figure 6-3 Example of program display with program counter indicated

### 6.2.3. Line Number Display Area

This area is where line numbers are displayed.
In source display mode, the line numbers of the source file are displayed in decimal here. In disassemble display mode, the line number display area is nonexistent.

PDB38M allows you to change the contents displayed in the program display area from the line number display area easily. Point to somewhere in the line number display area and double-click the mouse button. A Source dialog box will pop up. For details on how to use this dialog box, refer to the explanation of menu [Option] → [View] → [Source] in Section 6.5, "Using Extension Menu."

### 6.2.4. Address Display Area

This area is where addresses are displayed. The address of each line in the program display area is displayed in hexadecimal.

In source display mode, this area by default is not displayed. If you want this area to be displayed in source display mode, choose menu [Option] → [Layout] → [Address Area].
PDB38M allows you to change the contents displayed in the program display area from the address display area easily. Point to somewhere in the address display area and double-click the mouse button. An Address dialog box will pop up. For details on how to use this dialog box, refer to the explanation of menu [Option] → [View] → [Address] in Section 6.5, "Using Extension Menu."

### 6.2.5. Breakpoint Display Area

This area is where the lines that have software breakpoints set are displayed. The lines that have software breakpoints set are marked by the letter 'B'. The lines where you can set a software breakpoint are marked by the bar '-'. Nothing is shown for the lines where you cannot set a software breakpoint (e.g., a blank line or comment line).
Figure 6-4 shows an example of a breakpoint display area shown on the screen.

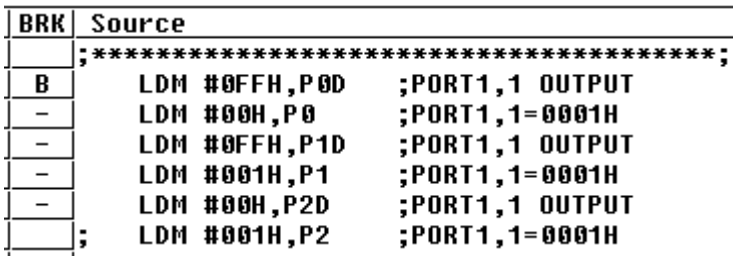



Figure 6-4 Example of breakpoint display area

Point to somewhere in this area and double-click the mouse button to set or clear a software breakpoint.

- Point to the bar '–' shown in the breakpoint display area and double-click the mouse button to set a software breakpoint there. In this case, the bar '-' changes to the letter 'B'.
- Point to the letter 'B' shown in the breakpoint display area and double-click the mouse button to clear a software breakpoint there. In this case, the letter 'B' returns to the bar '–'.

### 6.2.6. Object Code Display Area

This area is where object code is displayed.
In disassemble display mode, the object code of the disassembled lines are displayed in hexadecimal. In source display mode, this display area is nonexistent.

### 6.2.7. Vertical Scroll Bar

This bar scrolls the entire display area (program, line number, address, breakpoint, and object code display areas) in the vertical direction.

In source display mode, the display can be scrolled within the range of the source file being displayed on the screen. In disassemble display mode, the display cannot be scrolled in reverse unless you first scrolled it in forward. Scrolling the display in reverse functions as a means for returning the display several lines back while scrolling forward. When scrolling forward, the previous display addresses are saved in the internal buffer. Backward scroll is accomplished by using this address information.

### 6.2.8. Horizontal Scroll Bar

This bar scrolls the program display area in the horizontal direction.

## 6.3. Display Modes of Program Window

The program window supports two display modes: source and disassemble. Each display mode is explained below.

### 6.3.1. Source Display Mode

The source display mode is used to debug the target program at the source level. In this mode you can reference the source file of the target program.
Figure 6-5 shows an example of program display in source display mode.

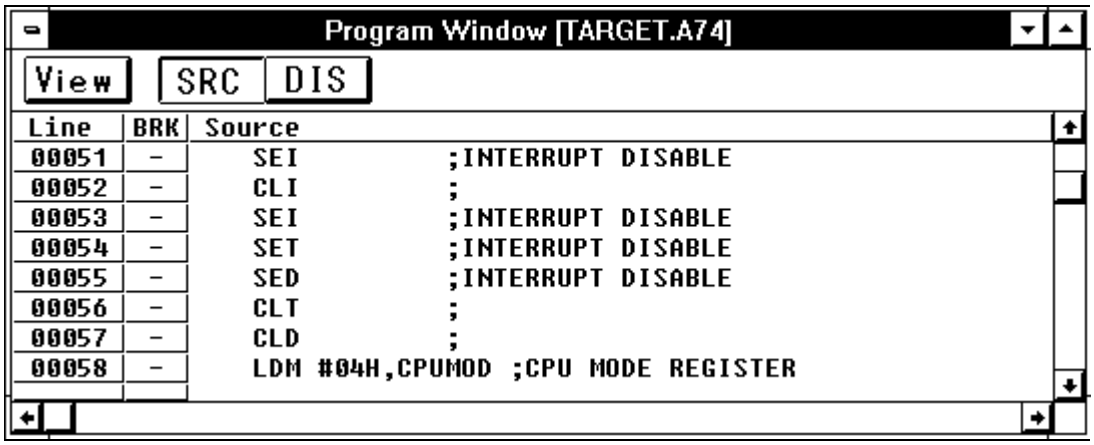



Figure 6-5 Example of program display in source display mode

- In source display mode, PDB38M allows you to debug your program at the source level.
- In source display mode, the program window has the following display areas:
  a. Line number display area
  b. Address display area (by default, not displayed)
  c. Breakpoint display area
  d. Program display area
- Displayed in the program display area are the contents of the source file. Use menu [Option] → [TAB] to set the number of columns for tabs in the source file.
- The line number and the address display areas can be turned on or off by selecting or deselecting menu [Option] → [Layout] → [Line Area] and [Option] → [Layout] → [Address Area], respectively.

- The source file being displayed in the program display area can be scrolled from the top line to the bottom line by using the vertical scroll bar. The associated other display areas are also scrolled simultaneously.
- The program display area can be scrolled in the horizontal direction by using the horizontal scroll bar.
- When the program window is in source display mode, the target program is single-stepped, overstepped, or returned one source line at a time by issuing the Step, Over, or Return command from the PDB38M window menus or tool bar buttons.

### 6.3.2. Disassemble Display Mode

The disassemble display mode is used to debug the target program at the instruction level. In this mode, PDB38M allows you to reference the disassembled result of the target program.
Figure 6-6 shows an example of program display in disassemble display mode.

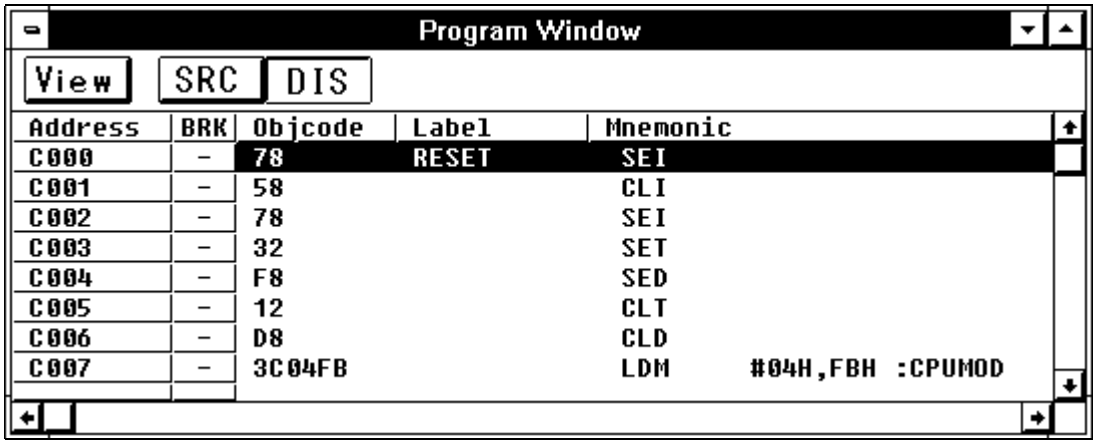
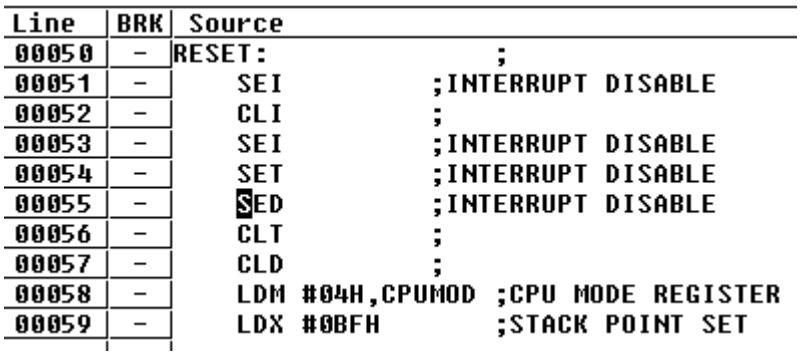



Figure 6-6 Example of program display in disassemble display mode

- The disassemble display mode is the default display mode that appears when PDB38M starts up.
- In disassemble display mode, the program window has the following display areas:
  a. Address display area
  b. Breakpoint display area
  c. Object code display area
  d. Program display area
- Object code is displayed in the object code display area (**Objcode**).
- The disassembled result is displayed in the program display area. The display contents, from left to right, are **Label** to display labels and **Mnemonic** to display mnemonics.

- The relative sizes of the object code display area (**Objcode**) and two fields in the program display area (**Label** and **Mnemonic**) can be adjusted easily by following the procedure described below.




- The address and the object code display areas can be turned on or off by selecting or deselecting menu [Option] → [Layout] → [Address Area] and [Option] → [Layout] → [Code Area], respectively.
- The object code being displayed in the object code display area can be scrolled in the vertical direction by using the vertical scroll bar. The associated other display areas are also scrolled simultaneously.
  - The display cannot be scrolled in reverse unless you first scrolled it in forward.
  - When scrolling the display in forward, the previous display addresses are saved in the internal buffer. Scroll in reverse is accomplished by using this address information.
  - If the first line address is modified by command execution, the content of the internal buffer is cleared.
- The program display area can be scrolled in the horizontal direction by using the horizontal scroll bar.
- If sections other than the program proper (e.g., data or blank sections) are disassembled, the memory contents of those sections are assumed to be instruction code as PDB38M displays the disassembled target program. At this time, undefined instructions and undefined operands are marked by "???."
- When the program window is in disassemble display mode, the target program is single-stepped, overstepped, or returned one instruction line at a time by issuing the Step, Over, or Return command from the PDB38M window menus or tool bar buttons.

## 6.4. Updating of Display Contents and Display Modes

Since the program window keeps tracking the target program by displaying the lines corresponding to the current program counter, the display contents and display modes are automatically updated after command execution. This automatic updating is explained below.

### 6.4.1. Automatic Updating of Display Contents

If after executing a command that causes the program counter to change (e.g., break in target execution, Step, or Reset), there is no line in the program display area that corresponds to the program counter, display contents are automatically updated in order to display the program counter line.

- If this updating occurs in source display mode, the line corresponding to the program counter is placed at the second line of the program display area.
- If this updating occurs in disassemble display mode, the line corresponding to the program counter is placed at the first line of the program display area.

Note that since the target system is reset after the target program is downloaded, display contents are updated in the same way as you execute Reset.

### 6.4.2. Automatic Updating of Display Modes

Automatic updating of display modes occurs in two ways: updating from source display mode to disassemble display mode, and updating from disassemble display mode to source display mode. In either case, automatic updating takes place under specific conditions as described below.

- Automatic updating from source display mode to disassemble display mode
  When target program execution is made to break (as Stop command is executed or a breakpoint is reached) or when the program counter enters an area where there is no source line information, the display mode is changed to the disassemble display mode.
- Automatic updating from disassemble display mode to source display mode
  If when the target system is reset after downloading the target program, there is any source line whose address corresponds to the program counter, the display mode is changed to the source display mode.

### 6.4.3. Manual Updating of Display Contents and Display Modes

The program window's display contents or display modes can be changed by manual operation as described below.

- Changing display contents
  – Display contents can be changed by using the horizontal or vertical scroll bar, tool bar's display area change button, or window menu.
  – Display contents are changed when you specify a source file name or address or perform character string search.

- Changing display modes
  - Display modes can be switched over by using window menus or tool bar buttons.

## 6.5. Using Extension Menu

The program window's extension menu provides the commands necessary to change the contents of the program window display areas and switch over the display modes. The program window's extension menu is added to the PDB38M window's basic menu [Option] when the program window is active.
Table 6-1 lists the composition of the program window's extension menu.

Table 6-1 Program Window's Extension Menu

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Option | Font... | Changes font. | |
| | TAB... | Sets tabs for source file display. | |
| | View | Changes display contents. | |
| | Source... | Displays beginning with specified source file or function. | |
| | Address... | Displays beginning with specified address or line number. | |
| | Program Counter | Displays beginning with current program counter. | |
| | Mode | Changes display modes. | |
| | Source mode | Changes to source display mode. | Ctrl + R |
| | Disasm mode | Changes to disassemble display mode. | Ctrl + R |
| | Layout | Sets layout. | |
| | Line Area | Turns on or off line number display area. | |
| | Address Area | Turns on or off address display area. | |
| | Code Area | Turns on or off object code display area. | |

The following explains how to use each extension menu command of the program window.

❒ Font

This menu command changes the font used in the program window display areas. When you choose **Font**, a font specification dialog box pops up. Figure 6-7 shows the structure of this dialog box. When this dialog box appears, specify your desired font.

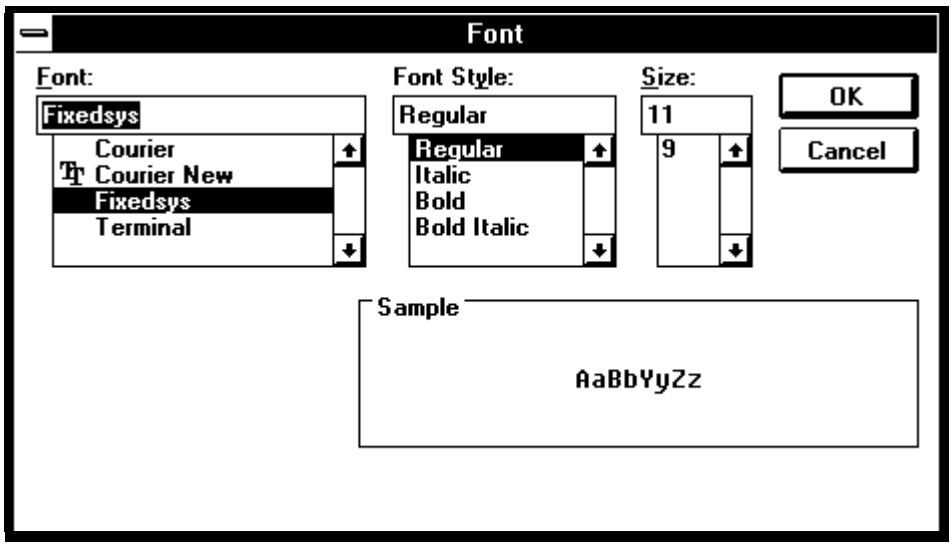



Figure 6-7 Font specification dialog box

❒ TAB

This menu command sets the tab value used for source file display.
When you choose **TAB**, a TAB dialog box pops up. Figure 6-8 shows the structure of this dialog box. When this dialog box appears, specify your desired tab value.

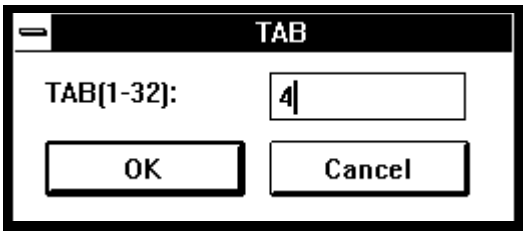



Figure 6-8 TAB dialog box

- Input your desired tab value in the **TAB(1-32):** field.
- This tab value can be specified using a value from 1 to 32.

## ❒ View

This menu item has commands to change the display contents of the program window display areas.
When you choose **View**, the following submenu commands are displayed. So choose your desired submenu command.

Source

This submenu command allows you to specify the source file or function you want to be displayed in the program display area.

When you choose **Source**, a Source dialog box pops up. Figure 6-9 shows the structure of this dialog box. When this dialog box appears, choose your desired source file name.

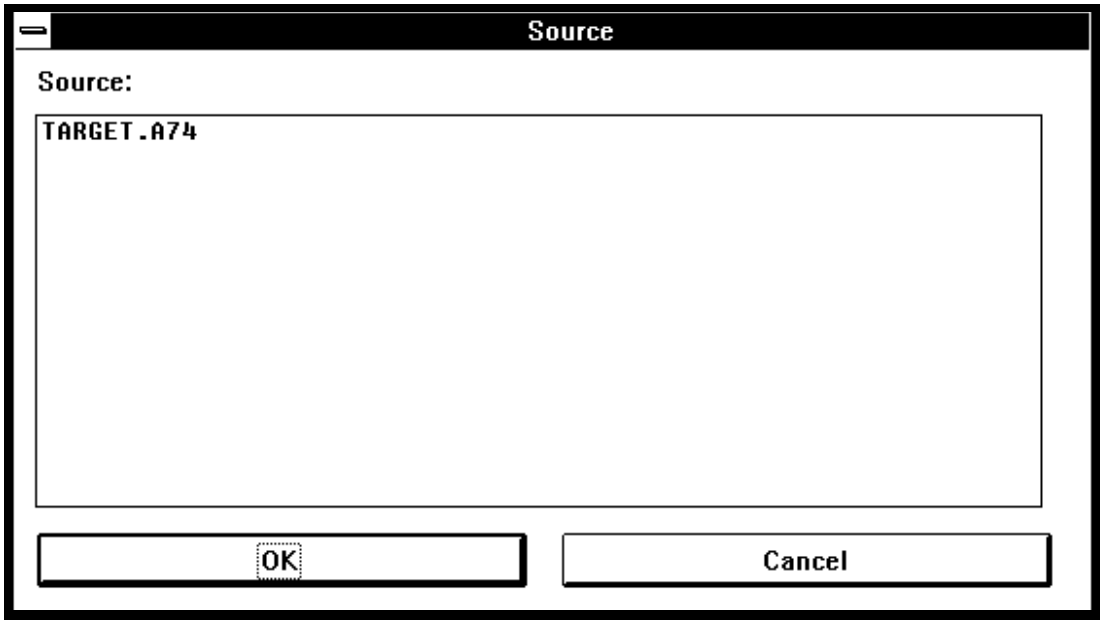



Figure 6-9 Source dialog box

- Displayed in the **Source:** field is a list of modules that constitute the downloaded load module file and source files that belong to each module (including header files).
- Point to your desired source file name in the **Source:** field and double-click the mouse button (or click the mouse button and press the **OK** button). Your specified source file will be displayed in the program display area.
- Note that this submenu command can only be used when the display mode is the source mode.

Address

This submenu command allows you to specify the address or line number of the program you want to be displayed in the program display area.

When you choose **Address** in source mode, an Address dialog box for use in source mode pops up. Figure 6-10 shows the structure of this dialog box. When this dialog box appears, specify your desired address or line number.

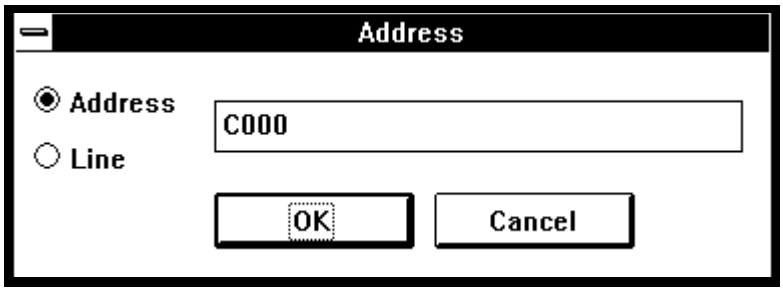



Figure 6-10 Address dialog box (when in source mode)

- To specify an address, click on the **Address** radio button and input your desired address in the field located to the right.
- To specify a line number, click on the **Line** radio button and input your desired line number in the field located to the right. The line number you input here is recognized as that of the source file currently being displayed in the program display area.
- Expressions can be used to specify an address or line number here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants for addresses written here must be hexadecimal, and that the radix of constants for line numbers must be decimal.

If you choose **Address** in disassemble mode, an Address dialog box for use in disassemble mode pops up. Figure 6-11 shows the structure of this dialog box. When this dialog box appears, specify your desired address.

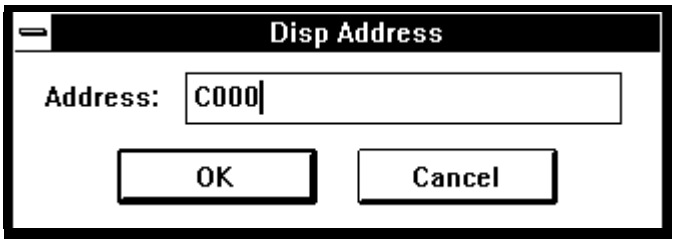



Figure 6-11 Address dialog box (when in disassemble mode)

- Input your desired address in the **Address:** field.
- Expressions can be used to specify an address here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal.

Program Counter

This submenu command changes the source file contents being displayed in the program display area to begin with the current program counter position.
When in source display mode, however, the source file contents being displayed here start one line before the program counter position.

## ❒ Mode

This menu item has commands to change the program display area's display mode.
When you choose **Mode**, the following submenu commands are displayed. So choose your desired submenu command.

Source mode

This submenu command changes the program display area's display mode to the source display mode.
If when you choose **Source mode**, there is source line information at the display start address, the contents of the source file are displayed.
However, if there is no source line information, the display mode is not changed to the source display mode.

Disasm mode

This submenu command changes the program display area's display mode to the disassemble display mode.
When you choose **Disasm mode**, the program contents are always displayed in disassemble mode regardless of whether or not there is source line information.

## ❒ Layout

This menu item has commands to set the program window layout. When you choose **Layout**, the following submenu commands are displayed. So choose your desired submenu command.

Line Area

This submenu command turns on or off the line number display area. When you choose **Line Area**, PDB38M brings up a line number display area. When you deselect it, the line number display area goes out.

Note that this submenu command can only be used when the program window is in source display mode.

Address Area

This submenu command turns on or off the address display area. When you choose **Address Area**, PDB38M brings up an address display area. When you deselect it, the address display area goes out.

Code Area

This submenu command turns on or off the object code display area. When you choose **Code Area**, PDB38M brings up an object code display area. When you deselect it, the object code display area goes out.

Note that this submenu command can only be used when the program window is in disassemble display mode.

## 6.6. Tool Bar Composition

The program window's tool bar is comprised of the buttons shown in Figure 6-12.

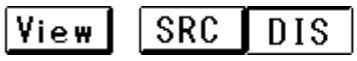



Figure 6-12 Program window's tool bar composition

The following explains how to use each button.

❍ View button

This button changes the program contents displayed in the program display area. When you click on the **View** button, a Disp Area dialog box pops up. Figure 6-13 shows the structure of this dialog box.

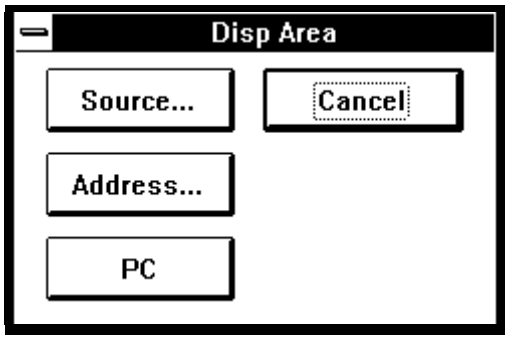



Figure 6-13 Disp Area dialog box

- When you choose **Source...**
  The program contents now are displayed beginning with the specified line of a specified source file.
  The operation procedure after you choose **Source...** is the same as you choose menu [Option] → [View] → [Source].
- When you choose **Address...** The program contents now are displayed beginning with a specified address.
  The operation procedure after you choose **Address...** is the same as you choose menu [Option] → [View] → [Address].

- When you choose **PC**
  The program contents now are displayed beginning with the current program counter value.
  The operation procedure after you choose **PC** is the same as you choose menu [Option] → [View] → [Program Counter].

### ❍ SRC button

This button changes the display mode to the source display mode. If there is source line information at the display start address, the contents of the source file are displayed. However, if there is no source line information, the display mode is not changed to the source display mode.
The operation performed by this button is the same as you choose menu [Option] → [Mode] → [Source mode].

### ❍ DIS button

This button changes the display mode to the disassemble display mode. The program contents are always displayed in disassemble mode regardless of whether or not there is source line information.
The operation performed by this button is the same as you choose menu [Option] → [Mode] → [Disasm mode].

# 7. Using Source Window

## 7.1. Outline of Source Window

The source window is used to reference a specific program area successively one line after another. The program contents displayed in this window are similar to those displayed in the program window. The difference is that whereas the contents displayed in the program window are automatically updated according to the program counter, those displayed in the source window are not updated unless you manually update them. Consequently, the source window allows you to reference a specific subroutine or task successively one line after another. Up to 10 source windows can be opened at a time. The method for operating the source window is the same as for the program window.

For details on how to use the source window, refer to Section 6, "Using Program Window."

# 8. Using Register Window

## 8.1. Outline of Register Window

The register window is used to display the contents of registers and flags. This window can also be used to set the contents of registers and flags.

## 8.2. Structure of Register Window

Figure 8-1 shows the structure of the register window.

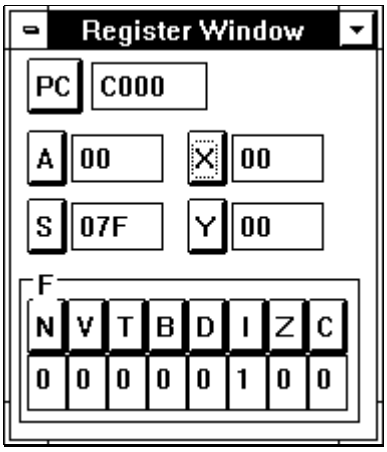



Figure 8-1 Structure of register window

Each part of the register window is explained below.

### 8.2.1. Register Area

This area is where register and flag values are displayed. This area consists of buttons indicating registers (or flags) and fields displaying the value of each register or flag.

PDB38M's register window has the register (or flag) buttons listed in Table 8-1 and a display field for each register (or flag).

Table 8-1 Types of Registers

| Type of Register | Available Button |
|---|---|
| Program counter | PC |
| Accumulator | A |
| Index registers, X and Y | X, Y |
| Stack pointer | S |
| Processor status register | PS (processor interrupt priority level)<br>N, V, T, B, D, I, Z, C (flags) |

Values for processor status register flags (N, V, T, B, D, I, Z, and C) are indicated by numerals 0 or 1. Values for other registers are indicated in hexadecimal.

These register and flag values can be changed. For details on how to change, refer to Section 8.3, "Setting Register Values."

# 8.3. Setting Register Values

## 8.3.1. Setting Processor Status Register

As you click on the flag button (N, V, T, B, D, I, Z, or C), its value toggles between 0 and 1. So use these flag buttons to set a flag to your desired value.

## 8.3.2. Setting Other Registers

When you click on the register button (PC, A, X, Y, or S) or IPL button, a Set Register dialog box like the one shown in Figure 8-2 pops up.

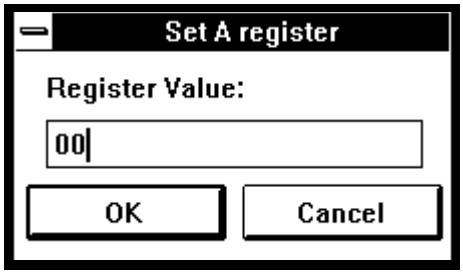



Figure 8-2 Set Register dialog box

- Input your desired register value in the **Register Value:** field.
- An expression can be used to input a register value. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal.

# 8.4. Updating of Register Window's Display Contents

The memory contents displayed in the register window are automatically updated under the following conditions:

- When a command is executed that causes the register contents to change

When a command is executed that causes the register contents to change (e.g., Stop or Step), the register contents being displayed are automatically updated. Therefore, you can check a change of register contents after executing Step or similar other commands.

# 9. Using Memory Window

## 9.1. Outline of Memory Window

The memory window displays memory contents in the forms of "Address," "Label," and "Data (memory contents)." This window can also be used to change memory contents. Up to 10 memory windows can be opened at a time.

## 9.2. Structure of Memory Window

Figure 9-1 shows the structure of the memory window.

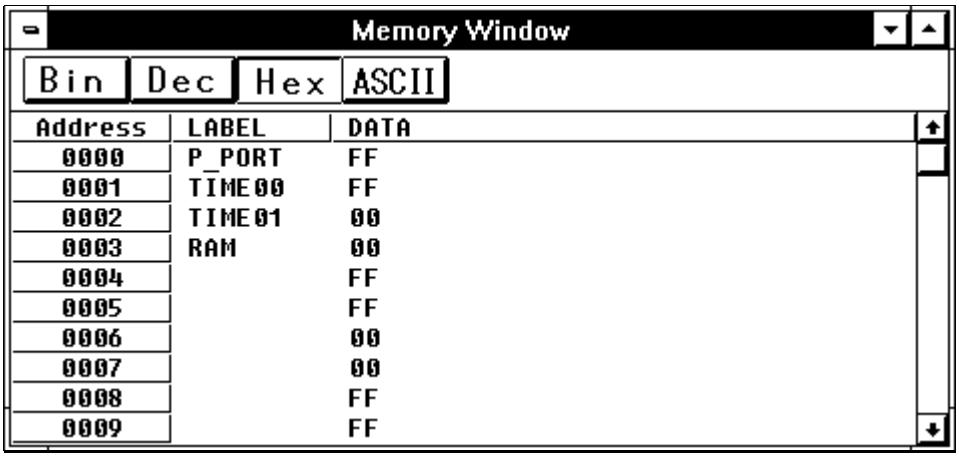



Figure 9-1 Structure of memory window

Each part of the memory window is explained below.

### 9.2.1. Tool Bar

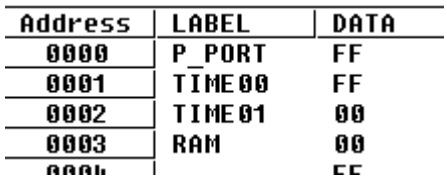
The memory window tool bar has the buttons necessary to change memory contents or change display contents.

For details about the function of each tool bar button, refer to Section 9.4, "Tool Bar Composition."

### 9.2.2. Memory Display Area

This area is where memory contents are displayed.

- The contents displayed in this area consist of **LABEL** to display labels and **DATA** to display memory contents.
- The relative sizes of these LABEL and DATA fields can be adjusted easily by following the procedure below.



| Address | LABEL | DATA |
|---|---|---|
| 0000 | P_PORT | FF |
| 0001 | TIME00 | FF |
| 0002 | TIME01 | 00 |
| 0003 | RAM | 00 |

- Memory contents are displayed in hexadecimal, decimal, binary, or ASCII characters.
- One data entry is displayed in one line.
- The data length can be selected from 1 byte or 2 bytes long.
- Memory contents can be changed from the memory display area easily. Point to your desired data displayed in the memory display area and double-click the mouse button. A Set dialog box will pop up. When you only input a data value in this dialog box, the data content at your selected position (address) is changed. For details on how to use the Set dialog box, refer to the explanation of menu [Option] → [Debug] → [Set] in Section 9.3, "Using Extension Menu."

### 9.2.3. Address Display Area

This area displays the address of each data displayed in the memory display area.

- The address display area is located to the left of the memory display area. The display in this area is scrolled up or down along with the memory display area as you manipulate the vertical scroll bar.
- The start address of memory display can be changed from the address display area easily. Point to somewhere in the address display area and double-click the mouse button. An Address dialog box will pop up. For details on how to use the Address dialog box, refer to the explanation of menu [Option] → [View] → [Address] in Section 9.3, "Using Extension Menu."

## 9.3. Using Extension Menu

The memory window's extension menu provides menu commands for changing the contents of the display area and input memory data. The memory window's extension menu is added to the PDB38M window's basic menu [Option] when the memory window is active.
Table 9-1 lists the composition of the memory window's extension menu.

Table 9-1 Memory Window's Extension Menu

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Option | Font... | Changes font. | |
| | View | Changes display contents. | |
| | Scroll Area... | Specifies scroll range. | |
| | Address... | Specifies display start address. | |
| | S | Changes display start address to S register value. | |
| | Data Length | Specifies display data length. | |
| | Byte | Displays in units of 1 byte. | |
| | Word | Displays in units of 2 bytes. | |
| | Radix | Specifies display radix. | |
| | Bin | Displays in binary. | |
| | Dec | Displays in decimal | |
| | Hex | Displays in hexadecimal | |
| | Ascii | Displays in ASCII characters. | |
| | Refresh | Redisplays data. | |
| | Debug | Sets memory contents. | |
| | Set... | Sets data at specified address. | |
| | Fill... | Fills specified memory block with data. | |

The following explains how to use the memory window's extension menu commands.

## ❒ Font

This menu command changes the font used in the memory window's display areas.

When you choose **Font**, a font specification dialog box pops up. Figure 9-2 shows the structure of this dialog box. When this dialog box appears, specify your desired font.

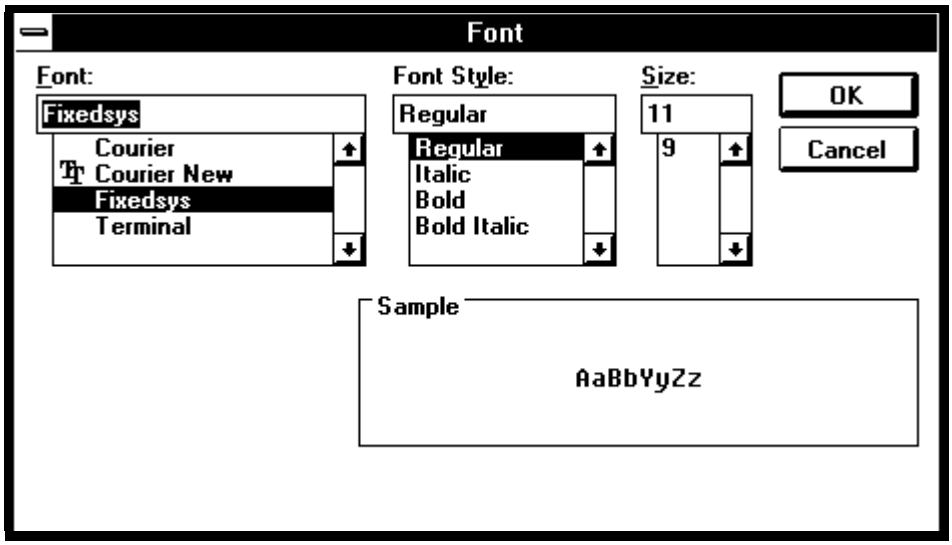



Figure 9-2 Font specification dialog box

## ❒ View

This menu item has commands to change the contents displayed in the memory window's display area.

When you choose **View**, the following submenu commands are displayed. So choose your desired submenu command.

Scroll Area

This submenu command specifies the memory display area's scroll range. When you choose **Scroll Area**, a Scroll Area dialog box pops up. Figure 9-3 shows the structure of this dialog box. When this dialog box appears, input your desired start and end addresses.

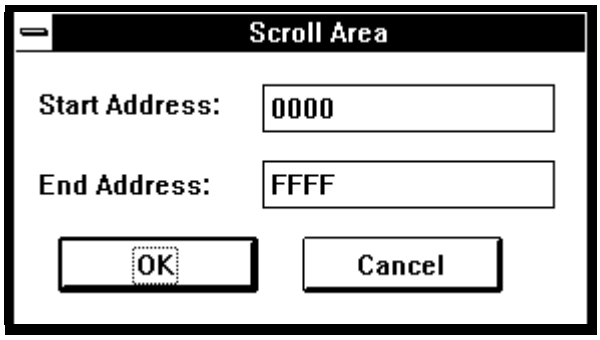



Figure 9-3 Scroll Area dialog box

- Input the start address of the scroll range you want in the **Start Address:** field and the end address in the **End Address:** field. The memory display area can be scrolled in this specified range.
- Expressions can be used to specify the start and end addresses here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants written here must be hexadecimal.
- If your specification of a scroll range results in the current display address getting out of the scroll range, memory contents are redisplayed beginning with the first address of your specified scroll range.

Address

This submenu command specifies the start address of memory display. When you specify this start address, the contents shown in the memory display area are redisplayed beginning with your specified address. When you choose **Address**, an Address dialog box pops up. Figure 9-4 shows the structure of this dialog box. When this dialog box appears, input your desired address.

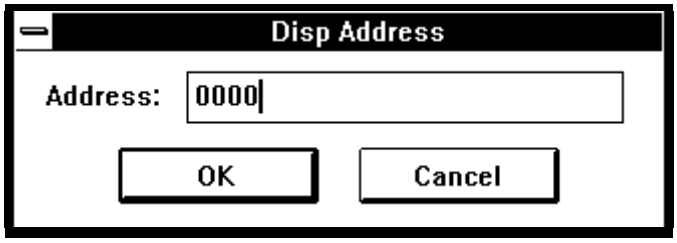



Figure 9-4 Address dialog box

- Input your desired address in the **Address:** field. Memory contents are redisplayed beginning with your specified address.
- An expression can be used to specify this address. For details about the format of an expression, refer to Section 15, "Method for Writing

Expressions" in Part III: Script Specification .
Note that the radix of constants written here must be hexadecimal.

- If you specify an address that is outside the scroll range, an error is assumed.

S

This submenu command sets the content of register S at the display start address.

Data Length

This submenu command specifies the data length of display data.
When you choose **Data Length**, the following subsubmenu commands are displayed. So choose your desired subsubmenu command.

Byte

This subsubmenu command displays memory contents in units of 1 byte.

Word

This subsubmenu command displays memory contents in units of 2 bytes.

Radix

This submenu command specifies the radix of display data.
When you choose **Radix**, the following subsubmenu commands are displayed. So choose your desired subsubmenu command.

Bin

This subsubmenu command displays memory contents in binary.

Dec

This subsubmenu command displays memory contents in decimal.

Hex

This subsubmenu command displays memory contents in hexadecimal.

Ascii

This subsubmenu command displays memory contents in ASCII characters.

Refresh

This submenu command refreshes data display.
The contents displayed in the memory window normally are updated automatically when you execute a command that causes memory contents to change (e.g., Set or Fill, Stop, or Step). The Refresh submenu command allows you to update the displayed contents at any time you want.

## ❒ Debug

This menu item has commands to set memory contents.
When you choose **Debug**, the following submenu commands are displayed. So choose your desired submenu command.

Set

This submenu command sets specified data at a specified address.
When you choose **Set**, a Set dialog box pops up. Figure 9-5 shows the structure of this dialog box. When this dialog box appears, input your desired address, data, and data length.

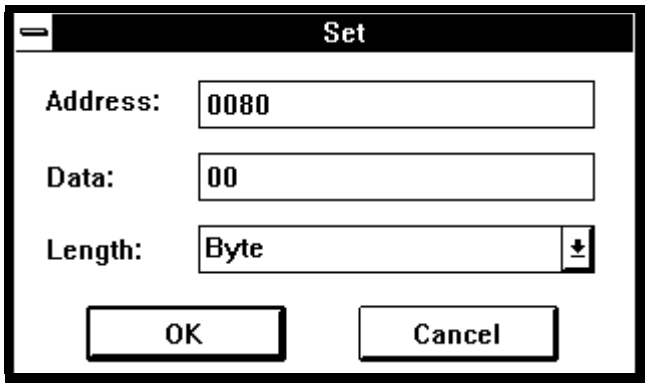



Figure 9-5 Set dialog box

- Input your desired address in the **Address:** field, data in the **Data:** field, and data length in the **Length:** field.
  Your specified data is written at your specified address.
- Expressions can be used to specify the address and data here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants for addresses written here must be hexadecimal, and that the radix of constants for data must be decimal if the radix shown in the memory window is decimal; otherwise, hexadecimal.
- The data length can be selected from Byte (1 byte long) or Word (2 bytes long). Use the dropdown list located at the right of the **Length:** field to choose Byte or Word (displayed as you click on the down arrow). If this specification is omitted, the data length in which contents are being displayed in the memory window is selected by default.

Fill

This submenu command fills memory with data.
When you choose **Fill**, a Fill dialog box pops up. Figure 9-6 shows the structure of this dialog box. When this dialog box appears, specify the start and end addresses, data, and data length.

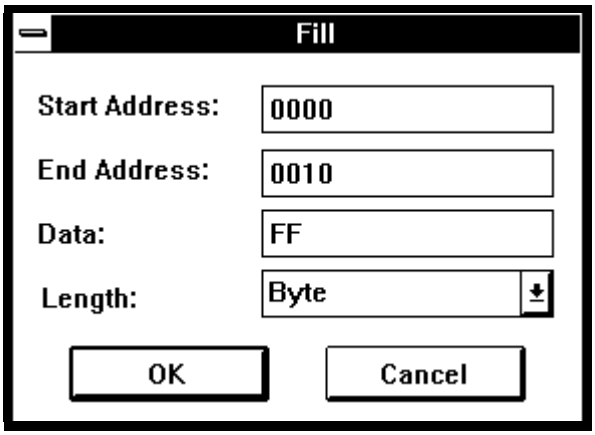



Figure 9-6 Fill dialog box

- Specify your desired start and end addresses in the **Start Address:** and **End Address:** fields, data in the **Data:** field, and data length in the **Length:** field.
  A range of memory from the start address to the end address you have specified is filled with your specified data.
- Expressions can be used to write the start and end addresses and data here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants for start and end addresses written here must be hexadecimal, and that the radix of constants for data must be decimal if the radix shown in the memory window is decimal; otherwise, hexadecimal.
- The data length can be selected from Byte (1 byte long) or Word (2 bytes long). Use the dropdown list located at the right of the **Length:** field to choose Byte or Word (displayed as you click on the down arrow). If this specification is omitted, the data length in which contents are being displayed in the memory window is selected by default.

## 9.4. Tool Bar Composition

The memory window's tool bar is comprised of the buttons shown in Figure 12-9 Composition of script window tool bar

.


Figure 9-7 Composition of memory window tool bar

The following explains how to use each button.

❍ **Bin button**

This button displays memory contents in binary.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Bin].

❍ **Dec button**

This button displays memory contents in decimal.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Dec].

❍ **Hex button**

This button displays memory contents in hexadecimal.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Hex].

❍ **ASCII button**

This button displays memory contents in ASCII characters.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Ascii].

## 9.5. Updating of Memory Window's Display Contents

The memory contents displayed in the memory window are automatically updated under the following conditions:

- When a command is executed that causes the memory contents to change

When a command is executed that causes the memory contents to change (e.g., Set or Fill, Stop, or Step), the memory contents being displayed are automatically updated. Therefore, you can check a change of memory contents after executing Step or similar other commands.

# 10. Using Dump Window

## 10.1. Outline of Dump Window

The dump window displays memory contents in dump form. This window can also be used to change memory contents. Up to 10 dump windows can be opened at a time.

## 10.2. Structure of Dump Window

Figure 10-1 shows the structure of the dump window.

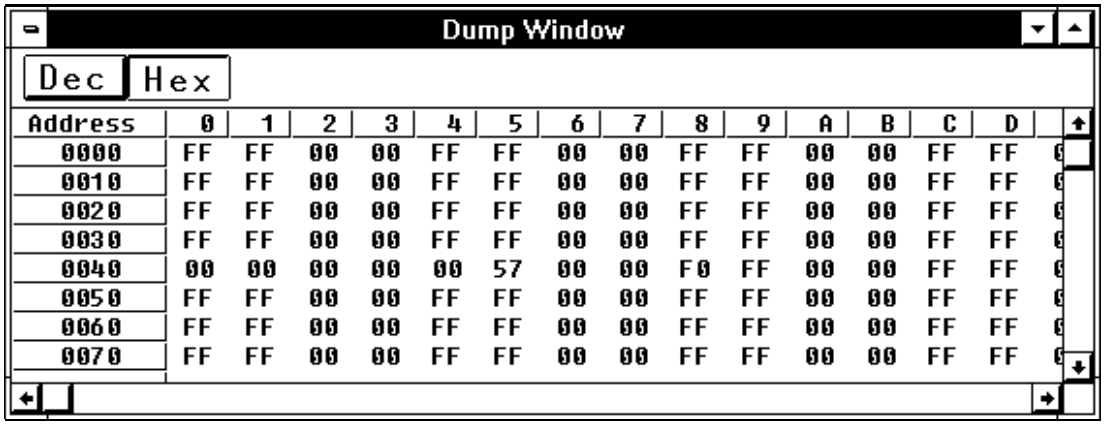



Figure 10-1 Structure of dump window

Each part of the dump window is explained below.

### 10.2.1. Tool Bar

The dump window tool bar has the buttons necessary to change memory contents or change display contents.
For details about the function of each tool bar button, refer to Section 10.4, "Tool Bar Composition."

### 10.2.2. Memory Display Area

This area displays memory contents in dump form.

- Memory contents are displayed in hexadecimal or decimal.
- 16 bytes of memory contents are displayed in one line.
- The data length can be selected from 1 byte or 2 bytes long.
- Shown above the memory display area are values indicating offsets from the top-of-the-line address. This field is scrolled left or right along with the memory display area as you manipulate the horizontal scroll bar.
- An ASCII code string (for 16 bytes) corresponding to the memory content on each line is displayed at the right side of the memory content.
- Memory contents can be changed from the memory display area easily. Point to your desired data shown in the memory display area and double-click the mouse button. A Set dialog box will pop up. When you only input a data value in this dialog box, the data content at your selected position (address) is changed. For details on how to use the Set dialog box, refer to the explanation of menu [Option] → [Debug] → [Set] in Section 10.3, "Using Extension Menu."

### 10.2.3. Address Display Area

This area displays the address at the beginning of each line displayed in the memory display area.

- The address display area is located to the left of the memory display area. The display in this area is scrolled up or down along with the memory display area as you manipulate the vertical scroll bar.
- The first address of each line in the memory display area resides on the 16-bite boundary (addresses whose LSB is 0).
- The start address of memory display can be changed from the address display area easily. Point to somewhere in the address display area and double-click the mouse button. An Address dialog box will pop up. For details on how to use the Address dialog box, refer to the explanation of menu [Option] → [View] → [Address] in Section 10.3, "Using Extension Menu."

## 10.3. Using Extension Menu

The dump window's extension menu provides menu commands for changing the display area or entering memory data. The dump window's extension menu is added to the PDB38M window's basic menu [Option] when the dump window is active.
Table 10-1 lists the composition of the dump window's extension menu.

Table 10-1 Dump Window's Extension Menu

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Option | Font... | Changes font. | |
| | View | Changes display contents. | |
| | Scroll Area... | Specifies scroll range. | |
| | Address... | Specifies display start address. | |
| | Data Length | Specifies display data length. | |
| | Byte | Displays in units of 1 byte. | |
| | Word | Displays in units of 2 bytes. | |
| | Radix | Specifies display radix. | |
| | Dec | Displays in decimal | |
| | Hex | Displays in hexadecimal | |
| | Refresh | Redisplays data. | |
| | Debug | Sets memory contents. | |
| | Set... | Sets data at specified address. | |
| | Fill... | Fills specified memory block with data. | |

The following explains how to use the dump window's extension menu commands.

## ❒ Font

This menu command changes the font used in the dump window's display areas. When you choose **Font**, a font specification dialog box pops up. Figure 10-2 shows the structure of this dialog box. When this dialog box appears, specify your desired font.

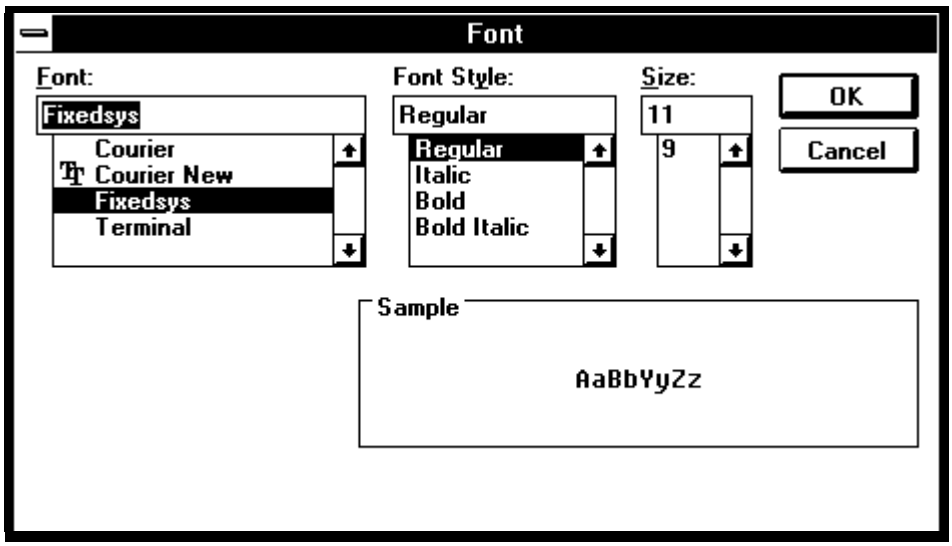



Figure 10-2 Font specification dialog box

## ❒ View

This menu item has commands to change the contents displayed in the dump window's display area.
When you choose **View**, the following submenu commands are displayed. So choose your desired submenu command.

Scroll Area

This submenu command specifies the memory display area's scroll range. When you choose **Scroll Area**, a Scroll Area dialog box pops up. Figure 10-3 shows the structure of this dialog box. When this dialog box appears, input your desired start and end addresses.

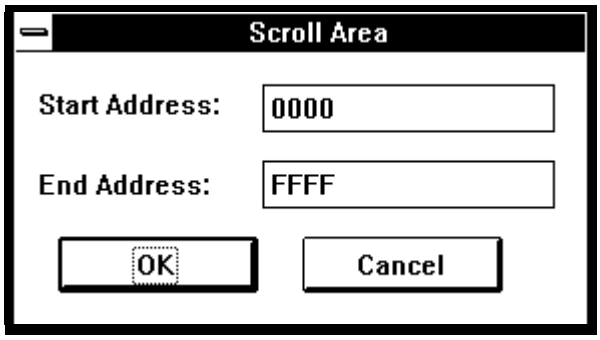



Figure 10-3 Scroll Area dialog box

- Input the start address of the scroll range you want in the **Start Address:** field and the end address in the **End Address:** field. The memory display area can be scrolled in this specified range.
- Expressions can be used to specify the start and end addresses here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification. Note that the radix of constants written here must be hexadecimal.
- If your specification of a scroll range results in the current display address getting out of the scroll range, memory contents are redisplayed beginning with the first address of your specified scroll range.

Address

This submenu command specifies the start address of memory display. When you specify this start address, the contents shown in the memory display area are redisplayed beginning with your specified address. When you choose **Address**, an Address dialog box pops up. Figure 10-4 shows the structure of this dialog box. When this dialog box appears, input your desired address.

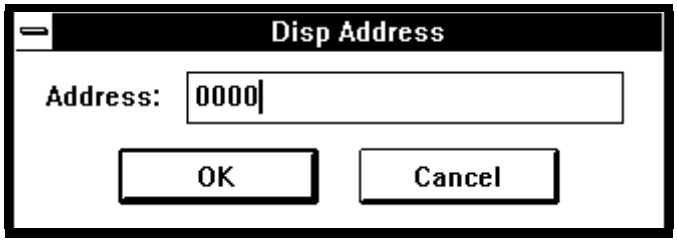



Figure 10-4 Address dialog box

- Input your desired address in the **Address:** field. Memory contents are redisplayed beginning with your specified address.
- However, if your specified address overlaps the 16-byte boundary (i.e., the LSB digit is not $0_{16}$), the LSB digit of your specified address is discarded and the resultant value is assumed to be your specified

address.

- An expression can be used to specify the address here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal
- If you specify an address that is outside the scroll range, an error is assumed.

Data Length

This submenu command specifies the data length of display data.
When you choose **Data Length**, the following subsubmenu commands are displayed. So choose your desired subsubmenu command.

Byte

This subsubmenu command displays memory contents in units of 1 byte.

Word

This subsubmenu command displays memory contents in units of 2 bytes. In this case, 8 data are displayed in one line.

Radix

This submenu command specifies the radix of display data.
When you choose **Radix**, the following subsubmenu commands are displayed. So choose your desired subsubmenu command.

Dec

This subsubmenu command displays memory contents in decimal.

Hex

This subsubmenu command displays memory contents in hexadecimal.

Refresh

This submenu command refreshes data display.
The contents displayed in the dump window normally are updated automatically when you execute a command that causes memory contents to change (e.g., Set or Fill, Stop, or Step). The Refresh submenu command allows you to update the displayed contents at any time you want.

## ❒ Debug

This menu item has commands to set memory contents.
When you choose **Debug**, the following submenu commands are displayed. So choose your desired submenu command.

Set

This submenu command sets specified data at a specified address.
When you choose **Set**, a Set dialog box pops up. Figure 10-5 shows the structure of this dialog box. When this dialog box appears, input your desired address, data, and data length.

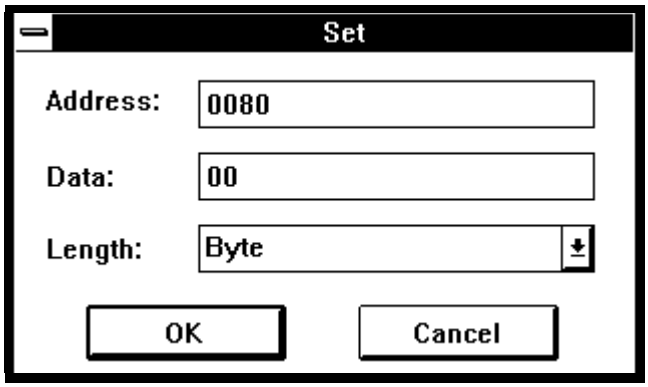



Figure 10-5 Set dialog box

- Input your desired address in the **Address:** field, data in the **Data:** field, and data length in the **Length:** field.
  Your specified data is written at your specified address.
- Expressions can be used to specify the address and data here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants for addresses written here must be hexadecimal, and that the radix of constants for data must be one in which contents are currently displayed in the dump window.
- The data length can be selected from Byte (1 byte long) or Word (2 bytes long). Use the dropdown list located at the right of the **Length:** field to choose Byte or Word (displayed as you click on the down arrow). If this specification is omitted, the data length in which contents are currently displayed in the dump window is selected by default

Fill

This submenu command fills memory with data.
When you choose **Fill**, a Fill dialog box pops up. Figure 10-6 shows the structure of this dialog box. When this dialog box appears, specify the start and end addresses, data, and data length.

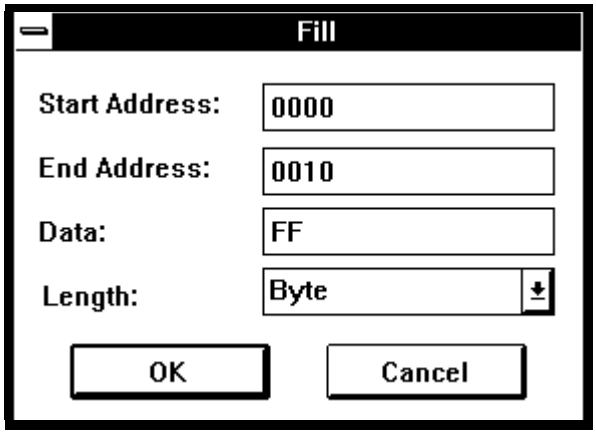



Figure 10-6 Fill dialog box

- Specify your desired start and end addresses in the **Start Address:** and **End Address:** fields, data in the **Data:** field, and data length in the **Length:** field.
  A range of memory from the start address to the end address you have specified is filled with your specified data.
- Expressions can be used to write the start and end addresses and data here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants for start and end addresses written here must be hexadecimal, and that the radix of constants for data must be decimal if the radix shown in the dump window is decimal; otherwise, hexadecimal.
- The data length can be selected from Byte (1 byte long) or Word (2 bytes long). Use the dropdown list located at the right of the **Length:** field to choose Byte or Word (displayed as you click on the down arrow). If this specification is omitted, the data length in which contents are currently displayed in the dump window is selected by default.

## 10.4. Tool Bar Composition

The dump window's tool bar is comprised of the buttons shown in Figure 12-9 Composition of script window tool bar

.


Figure 10-7 Composition of dump window tool bar

The following explains how to use each button.

❍ **Dec button**

This button displays memory contents in decimal.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Dec].

❍ **Hex button**

This button displays memory contents in hexadecimal.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Radix] → [Hex].

## 10.5. Updating of Dump Window's Display Contents

The memory contents displayed in the dump window are automatically updated under the following conditions:

- When a command is executed that causes the memory contents to change (e.g., Set or Fill, Stop, or Step), the memory contents being displayed are automatically updated. Therefore, you can check a change of memory contents after executing Step or similar other commands.

# 11. Using Watch Window

## 11.1. Outline of Watch Window

The watch window is used to display address expressions and corresponding memory contents. A specified address expression and the memory content corresponding to it are displayed side by side. The memory locations to be displayed in this watch window are called "watchpoints." The following can be used for watchpoints:

- Address expression (including symbol)
  The memory content of a specified address is displayed.
- Bit symbol
  A bit symbol has address and bit number information, indicating one arbitrary bit. If the watchpoint is a bit symbol, the value of a specified bit is indicated by 0 or 1.
- Address expression and bit number
  The value of a specified bit is indicated by 0 or 1.

The watchpoint memory contents being displayed are updated after command execution. So you can check a change of memory contents after executing Step or similar other commands.

## 11.2. Structure of Watch Window

Figure 11-1 shows the structure of the watch window.

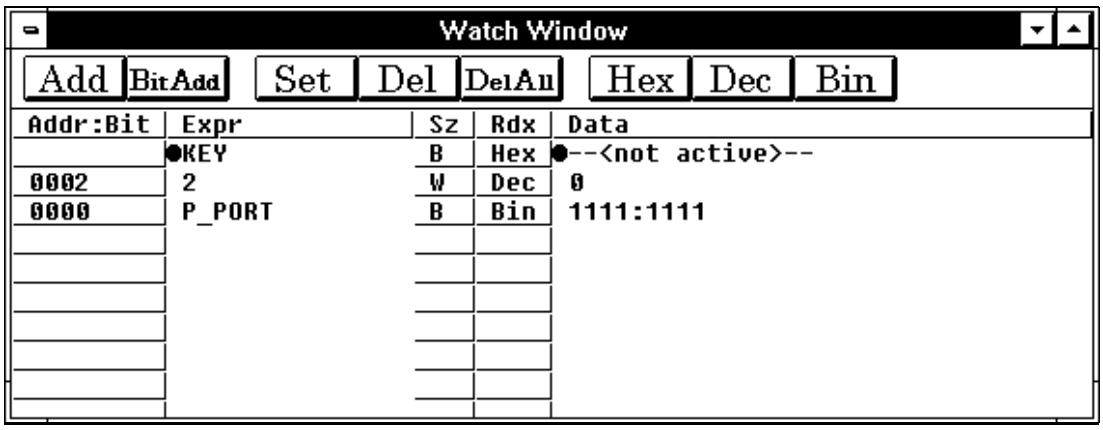



Figure 11-1 Structure of watch window

Each part of the watch window is explained below.

### 11.2.1. Tool Bar

The watch window tool bar has the buttons necessary to register and delete watchpoints.
For details about the function of each tool bar button, refer to Section 11.4, "Tool Bar Composition."

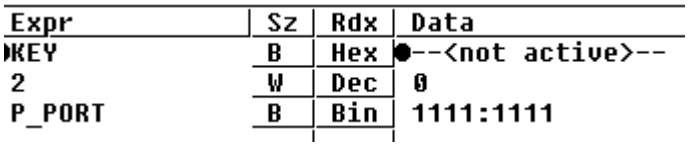
### 11.2.2. Address/Bit Number Display Area

This area displays watchpoint addresses in hexadecimal.
If the watchpoint is a bit (one bit position in memory), the address is followed by a colon (:) and a bit number (0 to 7).

### 11.2.3. Address Expression Display Area

This area displays watchpoint address expression.

- The width of this display area can be adjusted easily by following the procedure below.




- In the address expression and the data display areas the cursor positions are indicated by a red mark. The cursor position can be moved up or down by clicking somewhere in either area or using the ↑ and ↓ keys.

  Figure 11-2 shows an example of the cursors shown in the address expression and the data display areas.

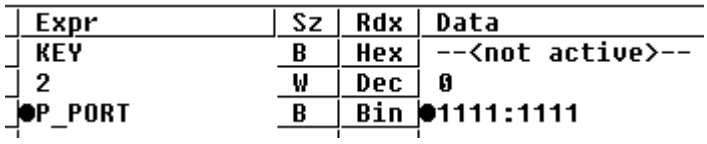



Figure 11-2 Example of cursor display

- The address/bit number display area can be turned on or off by selecting or deselecting menu [Option] → [Layout] → [Address Area]. (By default, this display area is turned on.)

### 11.2.4. Size Display Area

This area displays the length of data at watchpoints.
The letter '**B**' denotes Byte (1 byte long) and '**W**' denotes Word (2 bytes long).

- The size display area can be turned on or off by selecting or deselecting menu [Option] → [Layout] → [Size Area]. (By default, this display area is turned on.)

### 11.2.5. Radix Display Area

This area displays the radix of the watchpoint data being displayed in the window.
The word '**Hex**' denotes hexadecimal, '**Dec**' denotes decimal, and '**Bin**' denotes binary.

- The radix of the data being displayed can be changed from the radix display area easily. Point to somewhere in the radix display area and double-click the mouse button. The radix of the data displayed at that position is changed from the current radix to other radices cyclically as follows: Hexadecimal → Decimal → Binary → Hexadecimal

### 11.2.6. Data Display Area

This area displays memory contents at watchpoints.
If the radix is '**Bin**' (binary), the data is separated by a colon (:) every 4 bits and one blank character every 8 bits.

- In the address expression and the data display areas the cursor positions are indicated by a red mark. The cursor position can be moved up or down by clicking somewhere in either area or using the [↑] and [↓] keys.

  Figure 11-3 shows an example of the cursors shown in the address expression and the data display areas.

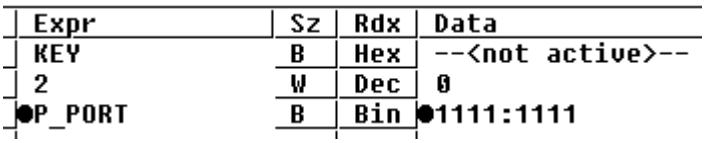



Figure 11-3 Example of cursor display

## 11.3. Using Extension Menu

The watch window's extension menu provides menu commands for registering and deleting watchpoints. The watch window's extension menu is added to the PDB38M window's basic menu [Option] when the watch window is active.

Table 11-1 lists the composition of the watch window's extension menu.

Table 11-1 Watch Window's Extension Menu

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Option | Font... | Changes font. | |
| | Watch | Registers/deletes watchpoint. | |
| | Add... | Registers watchpoint. | Ctrl + A |
| | Bitadd... | Registers bit-level watchpoint. | Ctrl + B |
| | Set... | Sets memory content at selected watchpoint. | Ctrl + S |
| | Del | Deletes watchpoint at selected position. | Ctrl + D |
| | DelAll | Deletes all watchpoints. | |
| | Radix | Changes display radix. | |
| | Bin | Displays watchpoint value at selected position in binary. | Alt + B |
| | Dec | Displays watchpoint value at selected position in decimal. | Alt + D |
| | Hex | Displays watchpoint value at selected position in hexadecimal. | Alt + H |
| | Layout | Sets layout. | |
| | Address Area | Turns on or off address/bit display area. | |
| | Size Area | Turns on or off size display area. | |

The following explains how to use the watch window's extension menu commands.

### ❒ Font

This menu command changes the font used in the watch window's display areas. When you choose **Font**, a font specification dialog box pops up. Figure 11-4 shows the structure of this dialog box. When this dialog box appears, specify your desired font.

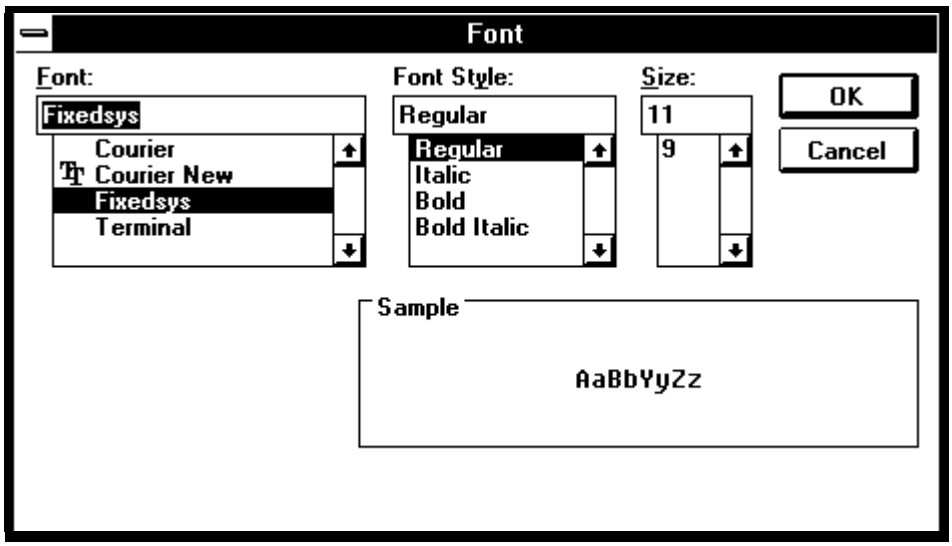



Figure 11-4 Font specification dialog box

### ❒ Watch

This menu item has commands to register/delete watchpoints and set watchpoint values.

When you choose **Watch**, the following submenu commands are displayed. So choose your desired submenu command.

Add

This submenu command registers a watchpoint

When you choose **Add**, an Add dialog box pops up. Figure 11-5 shows the structure of this dialog box. When this dialog box appears, specify the address expression that constitutes a watchpoint and your desired data length and radix.

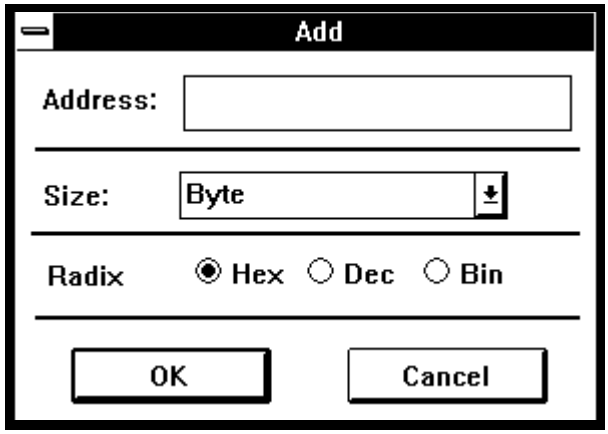



Figure 11-5 Add dialog box

- Specify the address expression in the **Address:** field and data length in the **Size:** field. To specify a radix, choose your desired radix from the radio buttons in the **Radix:** field.
  The address expression is registered as a watchpoint and memory contents of your specified data length are displayed using your specified radix.
- For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be hexadecimal.
- The data length can be selected from Byte (1 byte long) or Word (2 bytes long). Use the dropdown list located at the right of the **Size:** field to choose Byte or Word (displayed as you click on the down arrow).
- The radix can be selected from Hex (hexadecimal), Dec (decimal), or Bin (binary). For this selection, click on your desired radio button in the **Radix:** field.
- If the address expression cannot be calculated correctly (e.g., symbols are undefined), the warning dialog box shown in Figure 11-6 appears. When you click on the **Yes** button in this dialog box, the address expression is registered as an invalid watchpoint. No memory contents are displayed for invalid watchpoints. (The address/bit display area is left blank and the data display area is always marked by a message "--<not active>--.")
  Invalid watchpoints change to valid watchpoints if the address expression can be calculated correctly when it is recalculated upon re-downloading, Step execution, or break.

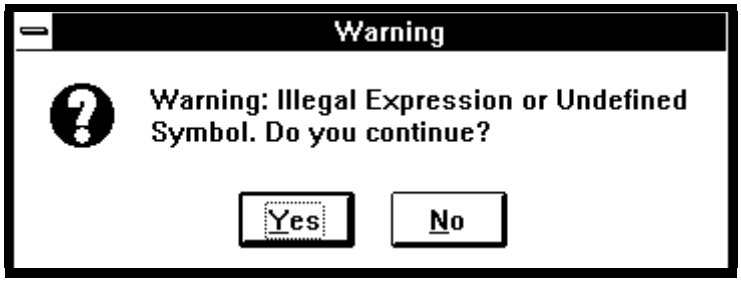



Figure 11-6 Warning dialog box

- The watchpoint added is inserted at the current cursor position. The cursor position is indicated by a red mark in the address expression and the data display areas.
  The cursor position can be moved up or down by clicking somewhere in either area or using the ↑ and ↓ keys.

Bitadd

This submenu command registers bit-level watchpoints.
When you choose **Bitadd**, a BitAdd dialog box pops up. Figure 11-7 shows the structure of this dialog box. When this dialog box appears, specify the bit symbol that constitutes a watchpoint or your desired address and bit number.

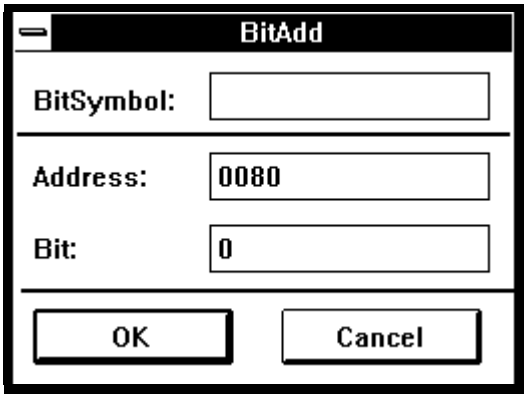



Figure 11-7 BitAdd dialog box Bit

- When using a bit symbol , input the bit symbol in the **BitSymbol:** field. When using an address and bit number, input the address in the **Address:** field and bit number (0 to 7) in the **Bit:** field. When using an address and bit number for watchpoint specification, you must specify both an address and a bit number.

- Expressions can be used to specify the addresses and bit numbers here. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants written here must be hexadecimal.
- The watchpoint added is inserted at the current cursor position. The cursor position is indicated by a red mark in the address expression and the data display areas.
  The cursor position can be moved up or down by clicking somewhere in either area or using the [↑] and [↓] keys.

Set

This submenu command sets a memory content at the watchpoint indicated by the cursor.
When you choose **Set**, a Set dialog box pops up. Figure 11-8 shows the structure of this dialog box. When this dialog box appears, specify the data you want to be placed at a watchpoint.

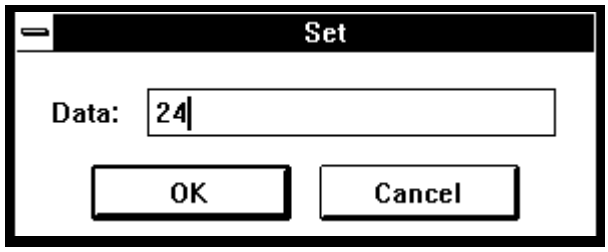



Figure 11-8 Set dialog box

- Input your desired data in the **Data:** field. The memory content at the watchpoint indicated by the cursor is changed to the data you have input.
- Expressions can be used to input this data. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification .
  Note that the radix of constants written here must be decimal if watchpoints are displayed in decimal or hexadecimal if other radices are used for watchpoint display.

Del

This submenu command deletes a watchpoint at the cursor position.

Del All

This submenu command deletes all watchpoints that are currently registered.

When you choose **Del All**, a confirmation dialog box pops up. When this dialog box appears, click on the **OK** button. If you click on the **Cancel** button here, no watchpoints will be deleted. Figure 11-9 shows the structure of the confirmation dialog box.

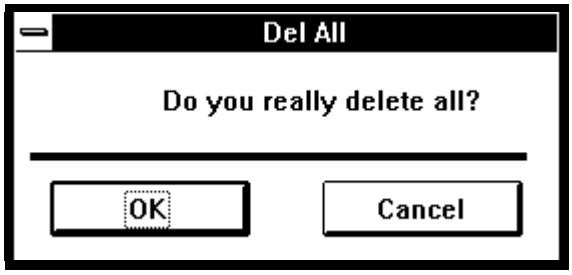



Figure 11-9 Confirmation dialog box

## ❒ Radix

This menu item has commands to specify the radix of watchpoints displayed in the window.

When you choose **Radix**, the following submenu commands are displayed. So choose your desired submenu command.

Bin

This submenu command displays the memory content at the watchpoint indicated by the cursor in binary.

Dec

This submenu command displays the memory content at the watchpoint indicated by the cursor in decimal.

Hex

This submenu command displays the memory content at the watchpoint indicated by the cursor in hexadecimal.

## ❒ Layout

This menu item has commands to set the watch window's layout.

When you choose **Layout**, the following submenu commands are displayed. So choose your desired submenu command.

Address Area

This submenu command turns on or off the address/bit number display area.

When you select **Address Area**, the address/bit number display area is displayed (turned on). When you deselect it, the address/bit number display area is not displayed (turned off).

Size Area

This submenu command turns on or off the size display area.

When you select **Size Area**, the size display area is displayed (turned on). When you deselect it, the size display area is not displayed (turned off).

## 11.4. Tool Bar Composition

The watch window's tool bar is comprised of the buttons shown in Figure 12-9 Composition of script window tool bar

.


Figure 11-10 Composition of watch window tool bar

The following explains how to use each button.

❍ Add button

This button registers watchpoints.
The operation performed by this button is the same as you choose menu [Option] → [Watch] → [Add].

❍ BitAdd button

This button registers bit-level watchpoints.
The operation performed by this button is the same as you choose menu [Option] → [Watch] → [BitAdd].

❍ Set button

This button sets a memory content at the watchpoint you have selected by clicking the mouse button.
The operation performed by this button is the same as you choose menu [Option] → [Watch] → [Set].

❍ Del button

This button deletes the watchpoint you have selected by clicking the mouse button.
The operation performed by this button is the same as you choose menu [Option] → [Watch] → [Del].

❍ DelAll button

This button deletes all watchpoints.
The operation performed by this button is the same as you choose menu [Option] → [Watch] → [DelAll].

❍ Hex button

This button displays the memory content at the watchpoint you have selected by clicking the mouse button in hexadecimal.
The operation performed by this button is the same as you choose menu [Option] → [Radix] → [Hex].

❍ **Dec button**

This button displays the memory content at the watchpoint you have selected by clicking the mouse button in decimal.
The operation performed by this button is the same as you choose menu [Option] → [Radix] → [Dec].

❍ **Bin button**

This button displays the memory content at the watchpoint you have selected by clicking the mouse button in binary.
The operation performed by this button is the same as you choose menu [Option] → [Radix] → [Bin].

## 11.5. Updating of Watch Window's Display Contents

The watchpoint memory contents displayed in the watch window are automatically updated under the following conditions:

- When a command is executed that causes the memory contents to change
  When a command is executed that causes the memory contents to change (e.g., Set or Fill, Stop, or Step), the memory contents being displayed are automatically updated. Therefore, you can check a change of memory contents after executing Step or similar other commands.

## 11.6. Function for Recalculating Watchpoint Address

In the watch window, PDB38M recalculates the address expressions of registered watchpoints after downloading the target program and uses the new addresses as it references memory contents. Therefore, even when the watchpoint addresses change as the program is modified, you do not need to set the addresses back again.

- Invalid watchpoints (whose memory values are displayed as "--<not active>--") become valid watchpoints if their address expressions are calculated correctly when recalculated.

## 11.7. Function for Saving Watchpoints

All watchpoints registered in the watch window are saved to the environment setup file when you close the watch window or quit PDB38M. Consequently, when you open the watch window or start up PDB38M next time, the previous watchpoints can be registered automatically.

# 12. Using Script Window

## 12.1. Outline of Script Window

The script window is used to execute and display the script commands in text format. Script commands can be executed by using a script file or through man-machine dialog. When script commands are executed from a script file, the commands can be executed automatically. Script command execution results can be saved to a log file.

The script commands that can be executed by PDB38M are detailed in Section 14, "Outline of Script Commands" in Part III: Script Specification . The method for writing a script file is detailed in Section XX, "Method for Writing Script File" in Part III: Script Specification .

Script Window Composition

Figure 12-1 shows the structure of the script window.

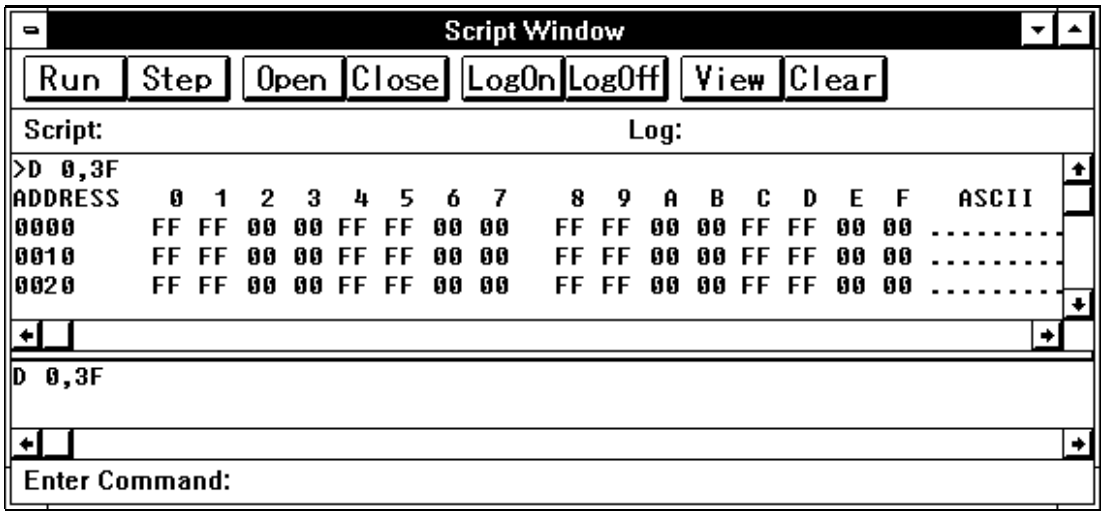



Figure 12-1 Structure of script window

Each part of the script window is explained below.

### 12.1.1. Tool Bar

For details about the function of each tool bar button, refer to Section 12.4, "Tool Bar Composition."

### 12.1.2. File Name Display Area

This area displays the name of a script file or log file when a script file or log file is open.

### 12.1.3. Command Display Area

This area displays the results of the script commands executed.

### 12.1.4. Command History Display Area

This area saves script command character strings specified through man-machine dialog. (Up to 100 instances of the latest script commands are saved.)
A command character string can be copied from this area into the command input area simply by clicking the mouse button. Furthermore, you can execute a command character string by selecting your desired one and double-clicking the mouse button.

### 12.1.5. Command Input Area

This area is where you input a script command through man-machine dialog.

### 12.1.6. Script File Display Area

This area appears only when the script window is open, displaying the contents of a script file. When script files are open that are nested, it is the last-opened script file whose contents are displayed here. Also, in the script file display area, the currently executed script file line is displayed in inverse video. Figure 12-2 shows the structure of the script window when a script file is open.

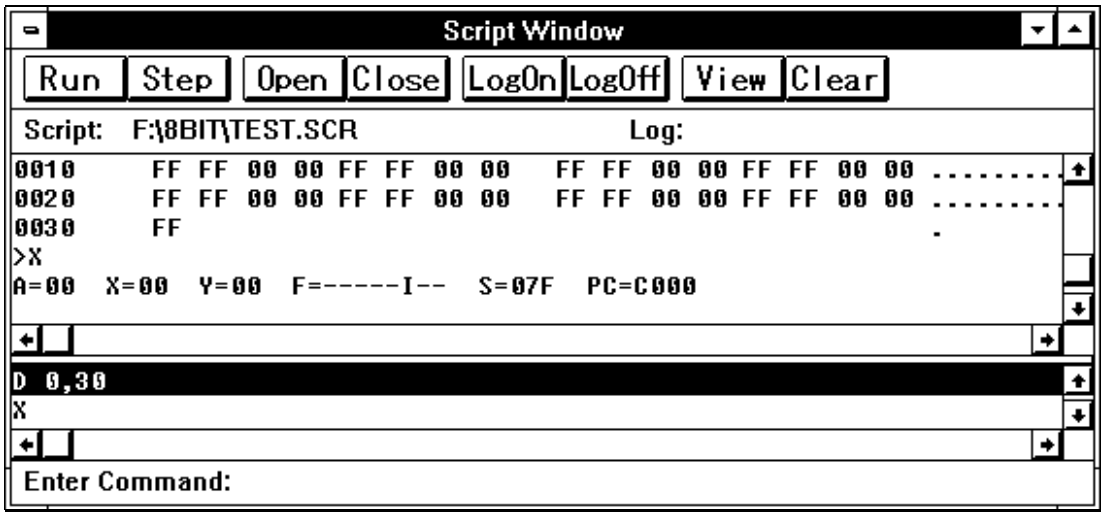



Figure 12-2 Structure of script window when script file is open

When you open a script file, the window's command history area is erased.

If while in this state you choose menu [Option] → [Script] → [Run] (or script window's **Run** button) or menu [Option] → [Script] → [Step] (or script window's **Step** button), the script file is executed. And when you choose menu [Option] → [Script] → [Close] (or script window's **Close** button), the script file is closed. For more information, refer to the explanation of how to use each menu command.

Even when a script file is open, you can issue a script command from the command input area providing that PDB38M has stopped executing the script file.

## 12.2. Stopping Execution of Script Command

Once you execute a script command from the command input area or a script file, PDB38M does not accept any other operation until it finishes processing the command. When PDB38M is executing a script command, a dialog box like the one shown in Figure 12-3 is displayed. (This dialog box is automatically closed when command execution is completed.) While this dialog box is on, any other operation of PDB38M cannot be performed. If want to stop command execution in the middle, click on the **STOP** button in this dialog box.


Figure 12-3 Example of dialog box that appears when executing script command

## 12.3. Using Extension Menu

The script window's extension menu provides menu commands for executing and stopping a script file, as well as single-stepping and closing a script file. The script window's extension menu is added to the PDB38M window's basic menu [Option] when the script window is active.
Table 12-1 lists the composition of the script window's extension menu.

Table 12-1 Script Window's Extension Menu

| Menu | Menu Item | Function | Keyboard Shortcut |
|---|---|---|---|
| Option | Font.... | Changes font. | |
| | Script | Manipulates script file. | |
| | Open... | Opens script file. | |
| | Run | Executes script file. | |
| | Stop | Stops execution of script file. | |
| | Step | Single-steps script file. | |
| | Close | Closes script file. | |
| | View | Manipulates view buffer. | |
| | Save... | Saves view buffer file. | |
| | Clear | Clears view buffer. | |
| | Log | Manipulates log file. | |
| | On... | Opens log file (starts outputting). | |
| | Off | Closes log file (terminates outputting). | |

The following explains how to use the script window's extension menu commands.

### ❒ Font

This menu command changes the font used in the script window's display areas. When you choose **Font**, a font specification dialog box pops up. Figure 12-4 shows the structure of this dialog box. When this dialog box appears, specify your desired font.

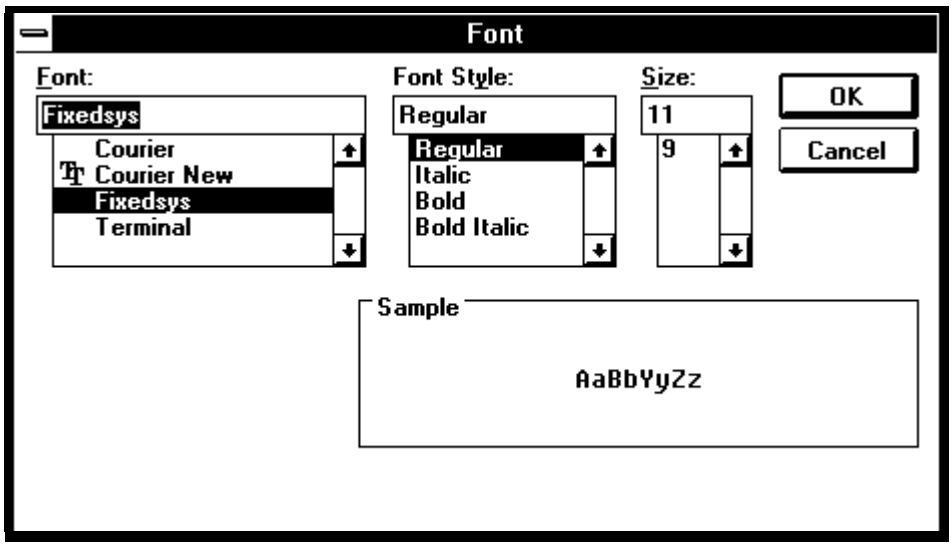



Figure 12-4 Font specification dialog box

### ❒ Script

This menu item has commands to open, close, and execute a script file. When you choose **Script**, the following submenu commands are displayed. So choose your desired submenu command.

Open

This submenu command opens a script file.
When you choose **Open**, a file selection dialog box pops up. Figure 12-5 shows the structure of this dialog box. When this dialog box appears, choose the script file you want to open.

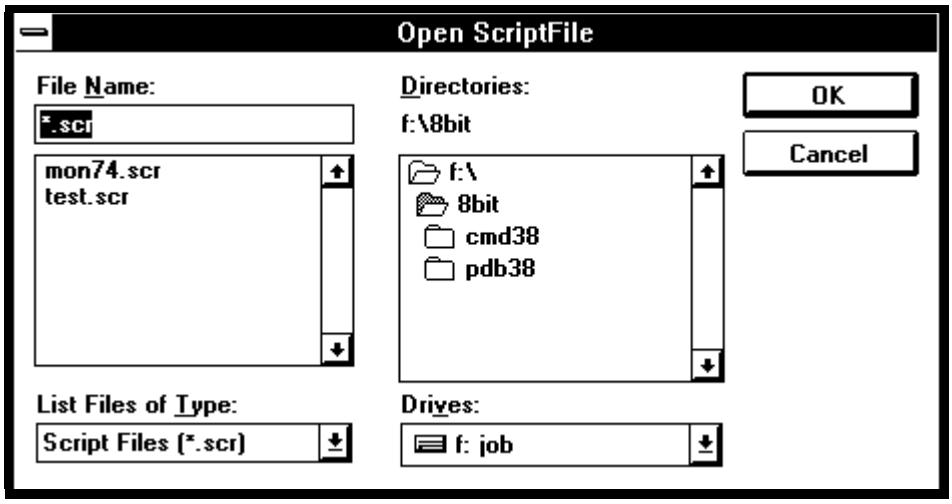



Figure 12-5 File selection dialog box

- For the script file to be opened, normally specify a file name whose extension is ".scr." (Script files whose filename extension is ".scr" are listed preferentially over other types of files.) However, you can open other type of file whose file attribute is not ".scr" by entering its full file name in the file name input field.
- When you click on the **OK** button after selecting a script file name, your specified script file is opened. When a script file is open, the window's command history display area goes out and a script file display area appears.
- Multiple script files can be opened by nesting them one in another. Script files can be nested up to five levels.

Run

This submenu command executes a script file.

Execution of a script file can be halted by clicking on the **STOP** button in the dialog box shown in Figure 12-6 when executing the script file.
Note that this is the same dialog box that appears when you execute a script command. If the command you stopped executing from this dialog box was one of those which were successively executed from a script file, the script file is stopped executing simultaneously when command execution is stopped, before the next line in the file is executed.


Figure 12-6 Example of dialog box that appears when executing script file

Stop

This submenu command stops execution of a script file.

Step

This submenu command single-steps a script file by executing one command at a time. This allows you to execute a script file step by step while checking the execution result of each script command.
Execution of a script file is halted after executing the line that is displayed in inverse video in the script file display area by one command.

Close

This submenu command closes a script file.
When you close a script file, the window's script file display area goes out and a command history display area appears again.

## ❒ View

This menu item has commands to save and clear the contents of the view buffer in your computer.
When you choose **View**, the following submenu commands are displayed. So choose your desired submenu command.

Save

This submenu command saves the contents of the current view buffer to a file. This file is called a "view file."
When you choose **Save**, a file selection dialog box pops up. Figure 12-7 shows the structure of this dialog box. When this dialog box appears, specify a view file name.

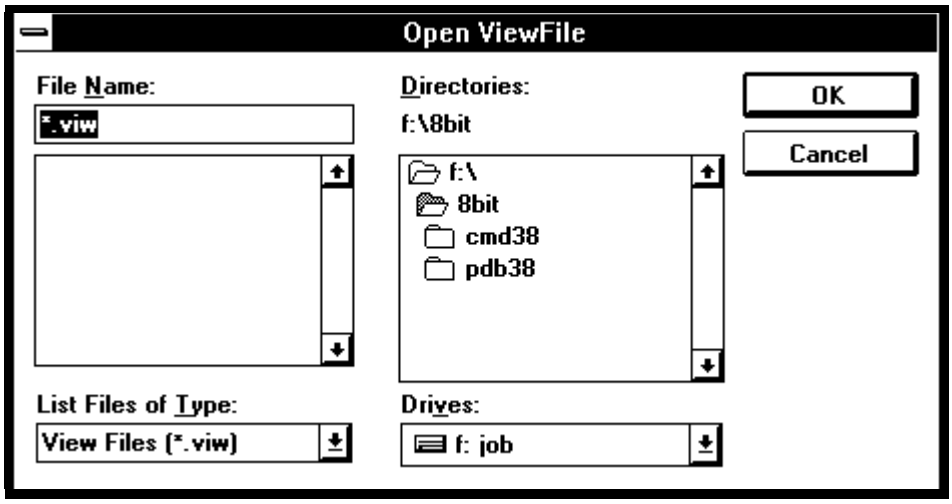



Figure 12-7 File selection dialog box

- You can specify any desired file name for the view file name.
- If you specify an existing file name for this file name, the contents of the view buffer are added to the existing file beginning with the end of the file.
- The view buffer contains command execution results for the latest 1,000 lines.

Clear

This submenu command clears the contents of the view buffer.

## ❒ Log

This menu item has commands to open and close a log file.
When you choose **Log**, the following submenu commands are displayed. So choose your desired submenu command.

On

This submenu command opens a log file and then outputs script command execution results to the log file.
When you choose **On**, a file selection dialog box pops up. Figure 12-8 shows the structure of this dialog box. when this dialog box appears, specify a log file name.

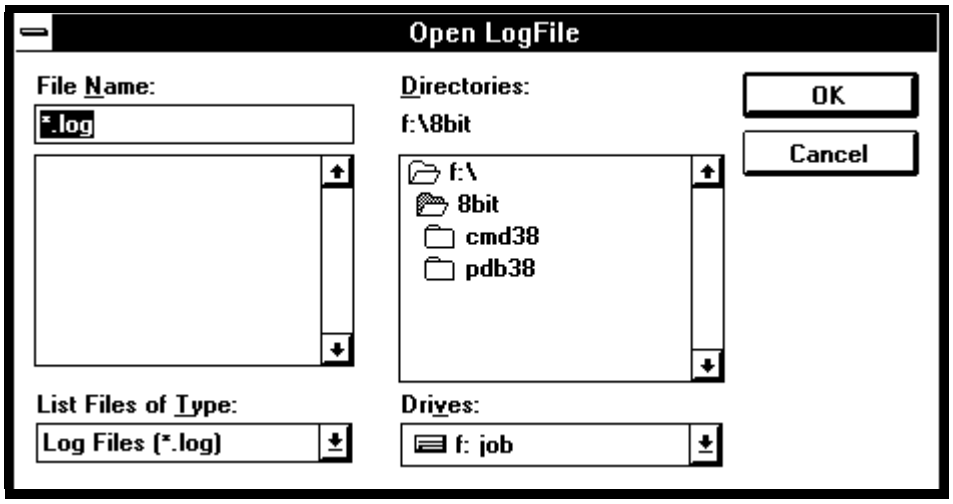



Figure 12-8 File selection dialog box

- For this log file, specify a file name whose extension is ".log."
- If you open a log file that you have once opened and closed after starting up PDB38M, new contents are added to that file beginning with the end of the file.
  However, if you re-open a log file that was already created before you started up PDB38M this time, this file is overwritten.
- Multiple log files can be opened by nesting them one in another. Log files can be nested up to eight levels.

Off

This submenu command closes the currently open log file.
Once you execute this command, execution results are no longer output to a log file. However, if log files are nested, although output to the current log file is finished, PDB38M restarts outputting to a log file that immediately precedes the closed file.

## 12.4. Tool Bar Composition

The script window's tool bar is comprised of the buttons shown in Figure 12-9.


Figure 12-9 Composition of script window tool bar

The following explains how to use each button.

❍ **Run button**

This button executes a script file.
The operation performed by this button is the same as you choose menu [Option] → [Script] → [Run].

❍ **Step button**

This button single-steps a script file by executing one command at a time.
The operation performed by this button is the same as you choose menu [Option] → [Script] → [Step].

❍ **Open button**

This button opens a script file.
The operation performed by this button is the same as you choose menu [Option] → [Script] → [Open].

❍ **Close button**

This button closes a script file.
The operation performed by this button is the same as you choose menu [Option] → [Script] → [Close].

❍ **LogOn button**

This button opens a log file. Once a log file is open, PDB38M outputs script command execution results to that file.
The operation performed by this button is the same as you choose menu [Option] → [Log] → [On].

❍ **LogOff button**

This button closes the currently open log file.
The operation performed by this button is the same as you choose menu [Option] → [Log] → [Off].

❍ **View button**

This button saves the contents of the current view buffer to a file.
The operation performed by this button is the same as you choose menu [Option] → [View] → [Save].

❍ **Clear button**

This button clears the contents of the view buffer.

The operation performed by this button is the same as you choose menu [Option] → [View] → [Clear].

# 13. Using S/W Breakpoint Setup Dialog Box

## 13.1. Outline of S/W Breakpoint Setup Dialog Box

This dialog box is used to set software breakpoints. This dialog box appears when you choose the PDB38M window's menu [Debug] → [Break Point] → [S/W Break Point].

A software breakpoint means a breakpoint that causes program execution to break before executing the instruction at an address specified by the breakpoint.

- PDB38M allows you to set up to four software breakpoints in one program.
- If you set multiple software breakpoints, a combination of these points to cause a break is the OR condition. Namely, program execution is halted when any one of software break addresses is reached.
- The software break function causes program execution to break before executing the instruction at the software breakpoint encountered.

## 13.2. Structure of S/W Breakpoint Setup Dialog Box

Figure 13-1 shows the structure of the S/W breakpoint setup dialog box.

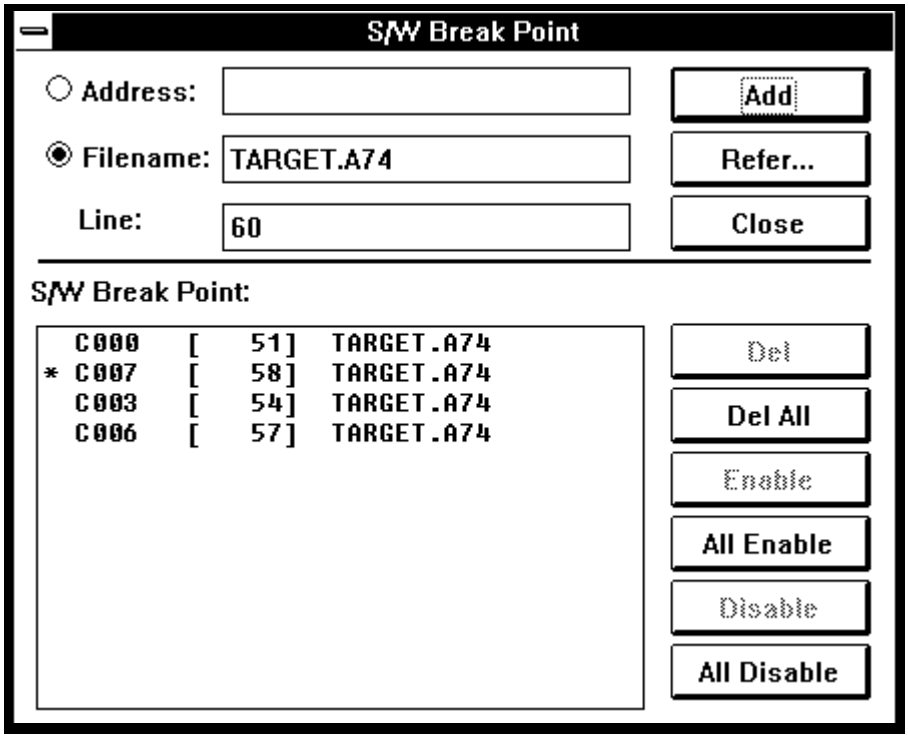



Figure 13-1 Structure of S/W breakpoint setup dialog box

## 13.3. Using S/W Breakpoint Setup Dialog Box

The S/W breakpoint setup dialog box allows you to perform the following operations:

1. Reference software breakpoints.
2. Set software breakpoints.
3. Delete software breakpoints.
4. Disable software breakpoints.
5. Enable software breakpoints.

These software breakpoint-manipulating operations can be specified successively one by one until you close the S/W breakpoint setup dialog box by clicking on its **Close** button.

The following sections explain how to specify each of the above operations from this dialog box.

### 13.3.1. Referencing Software Breakpoints

The **S/W Break Point:** field lists the software breakpoints that are currently set. By manipulating the vertical scroll bar of the **S/W Break Point:** field you can see all software breakpoints set.

- The contents of the list shown in this field consist of software breakpoint addresses, line numbers at the beginning of source lines corresponding to these addresses, and the source file name.

```
Display example: 0F0047 [   62] TEST1.c     ← Address [Line number]
                                               File name
```

- If there is no source line number corresponding to any address, only the address value is displayed.

```
Display example: 0F00CD                     ← Address
```

- Furthermore, if any software breakpoint is invalid (disabled), an asterisk (*) denoting invalidity is displayed at the left edge of that line.

```
Display example: * 0F0069 [  90] TEST1.c    ← Address [Line number]
                                               File name
```

### 13.3.2. Setting Software Breakpoints

A software breakpoint can be set by specifying an address or a source file name plus a line number.

- When specifying an address to set a software breakpoint
  After clicking on the **Address** radio button, input your desired address in the **Address:** field and then click on the **Add** button.
  - An expression can be used to specify this address. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants written here must be hexadecimal.

- When specifying a source file name plus line number to set a software breakpoint
  After clicking on the **Filename** radio button, input the source file name in the **Filename:** field and the line number in the **Line:** field and then click on the **Add** button.
  - When you click on the **Refer** button, a file selection dialog box pops up. You can use this dialog box to choose the source file you want to specify.

Figure 13-2 shows the structure of this dialog box.

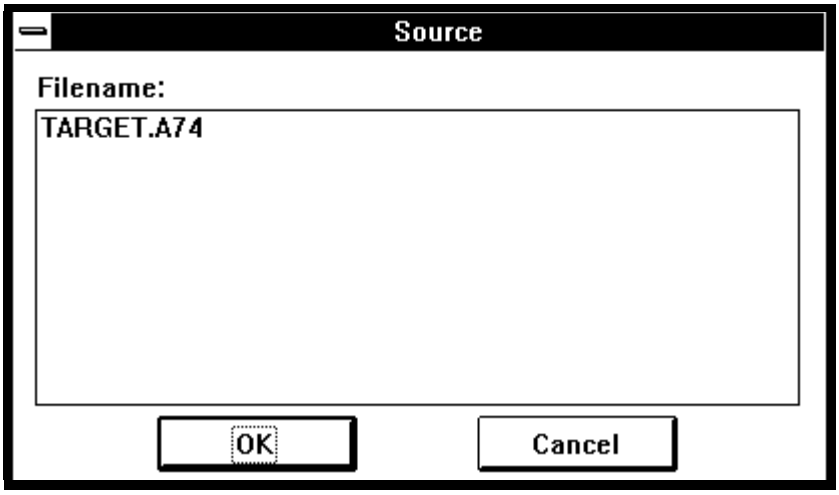



Figure 13-2 File selection dialog box

– An expression can be used to specify this line number. For details about the format of an expression, refer to Section 15, "Method for Writing Expressions" in Part III: Script Specification . Note that the radix of constants written here must be hexadecimal.

### 13.3.3. Deleting Software Breakpoints

There are two methods for deleting software breakpoints: one for deleting a selected software breakpoint and one for deleting all software breakpoints collectively.

- When deleting a selected software breakpoint
  Choose the software breakpoint you want to delete and click on the **Del** button.
  To choose a software breakpoint, point to your desired software breakpoint displayed in the **S/W Break Point:** field and click the mouse button.

- When deleting all software breakpoints currently set
  Click on the **DelAll** button.

  When you click on the **DelAll** button, a confirmation dialog box pops up, requesting your confirmation of whether you are sure you want to delete all. When this dialog box appears, click on the **OK** button. If you click on the **Cancel** button here, no software breakpoints are deleted. Figure 13-3 shows the structure of this dialog box.

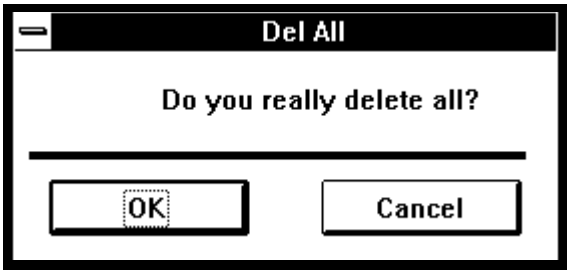



Figure 13-3 Confirmation dialog box

### 13.3.4. Disabling Software Breakpoints

There are two methods for disabling software breakpoints: one for disabling a selected software breakpoint and one for disabling all software breakpoints collectively.

- When disabling a selected software breakpoint
  Choose the software breakpoint you want to disable and click on the **Disable** button.
  To choose a software breakpoint, point to your desired software breakpoint displayed in the **S/W Break Point:** field and click the mouse button.

  – An asterisk (*) denoting invalidity will be displayed at the left edge of the software breakpoint you have selected.

  – If you performed Disable on an already invalid software breakpoint, the operation is ignored.

- When disabling all software breakpoints currently set
  Click on the **All Disable** button.

### 13.3.5. Enabling Software Breakpoints

There are two methods for enabling software breakpoints: one for enabling a selected software breakpoint and one for enabling all software breakpoints collectively.

- When enabling a selected software breakpoint
  Choose the software breakpoint you want to enable and click on the **Enable** button.
  To choose a software breakpoint, point to your desired software breakpoint displayed in the **S/W Break Point:** field and click the mouse button.

  - The asterisk (*) denoting invalidity that is displayed at the left edge of your selected software breakpoint will go out.
  - If you performed Enable on an already valid software breakpoint, the operation is ignored.

- When enabling all software breakpoints currently set

Click on the **All Enable** button.

# Part III: Script Specifications

# 14. Outline of Script Commands

## 14.1.Command List

The commands that can be executed in PDB38M's script window (i.e., script commands) are listed in Table 14-1 on the next page.

Table 14-1 Command List (Execution and Stopping Execution)

| Command Name | Outline Function of Command |
|---|---|
| A | Assembles program one line at a time. |
| BIT | References and modifies bits corresponding to bit symbols. |
| CD | Modifies/references current directory. |
| D | Displays memory contents in bytes. |
| F | Writes specified byte data to specified memory block. |
| G | Executes target program. |
| GB | Executes target program. (Breakpoints set by PB are valid.) |
| I | Reads HEX and SYM files. |
| L | Displays program memory contents after disassembling. |
| LOGOFF | Stops outputting PDB38M's execution results to log file. |
| LOGON | Outputs PDB38M's execution results to log file. |
| O | Outputs current memory contents to file (HEX file). |
| PB | Sets program breakpoints. |
| RADIX | Displays and modifies default values of numeric input. |
| S | Displays and modifies memory contents in bytes. |
| SCOPE | Specifies effective range of local labels and symbols. |
| SCRIPT | Opens script file. |
| SI | Displays section information. |
| SRC | Specifies source file name to be displayed in program window. |
| STOP | Stops execution of target program. |
| T | Executes program one instruction at a time and displays register contents, etc. |
| U | Executes program a specified number of instructions. (Register contents are displayed before execution and not displayed during execution.) |
| VER | Displays PDB38M's version number. |
| WAIT | Waits until target program breaks. |
| X | Displays and modifies register contents. |
| Z | Resets target MCU in hardware. |
| ; | Comment line |

## 14.2. Command Input Format

PDB38M accepts the commands that are input in the format shown in Figure 14-1.

| |
|---|
| Command name Command type Parameter 1 Subparameter 1<br><br>, Parameter 2, Subparameter 2 ..... <RET> |

Figure 14-1 Command input format

- Insert at least one blank character or tab between the command name and parameter, between the command name and command type, and between the command type and parameter.
- Uppercase and lowercase letters can be used to write command names and command types.
- The meaning of command type, meaning of each parameter, and whether or not a parameter can be omitted vary with each command.

# 14.3. Detailed Explanation of Commands

## 14.3.1. Description Format

Detailed explanation of each command here is written following the description format below.

- Title
  Indicates a command name and the command's outline function.

> **Command name**
>
> **Outline command function**

- Body text
  Explains a command about following seven items:

***Input format***

Indicates a command's input format.
The notation of input format is explained in Section 14.3.2, "Notational Conventions of Notational Conventions of Command Input Format."

***Run-time execution***

Indicates whether or not a command can be executed during program execution.

***Function***

Explains the function of the command.

***Parameter***

Explains the parameters shown in the input format.
If no parameter is used for the command, this section is written as "None."

***Command execution example***

Shows a command execution example.

### 14.3.2. Notational Conventions of Command Input Format

The notational conventions shown below are followed when explaining each command's input format in this chapter.

Table 14-2 Notational Conventions

| Convention | Meaning |
|---|---|
| **XXXX** | XXXX must be input. |
| **[ XXXX ]** | Input can be omitted or XXXX can be input. |
| **{ X1 \| X2 \| X3 }** | Any one of X1, X2, or X3 must be input. |
| **[ { X1 \| X2 \| X3 } ]** | Input can be omitted or any one of X1, X2, or X3 can be input. |

### 14.3.3. Detailed Explanation of Each Command

The following explains details of each command that is executed in the script window in alphabetical order.

# A (Assemble)

**Assembles program one line at a time.**

## *Input format*

Assemble [ address ]

## *Run-time execution*

Cannot be executed.

## *Function*

- This command assembles a program one line at a time beginning with a specified address or the address indicated by a label and writes the corresponding machine language to program memory. (Example 1)
- If an error is found in the input mnemonic or operand, the command displays an assemble error message and waits for input at the same address. (Example 2)
- If a jump address extends beyond the range of a specified addressing mode, the command displays an error message and waits for input at the same address. (Example 3)
- The label, symbol, or bit symbol written in the label column is registered as a global label or global symbol.
- When defining a label, be sure to add a colon (:) after the label name. (Example 4)
- If an address is omitted when entering the command, the program is assembled beginning with the last address processed by the command A in the previous execution.
  However, if address FFFFh was reached in the previous execution, the command displays an error message.
  Note that at startup of PDB38M, the program is assembled beginning with the program counter.
- Following four instructions can be used as pseudo-instructions:

| Pseudo-instruction | Function |
|---|---|
| . BLKB | Allocates 1 byte of RAM area (range: 1 to 0x10000). (Example 8) |
| . BYTE | Sets byte data (can be set up to 8 bytes). |
| . EQU | Defines symbol. (Example 8)<br>Defines bit symbol. (Example 9) |
| . WORD | Sets word data (can be set up to 4 bytes). (Example 10) |

- To stop the command A, enter only the RET key and nothing else. (Example 11) Also, if you press the RET key after entering only blank characters, the command A is halted. (Example 14)
- If the name you specified in defining a label, symbol, or bit symbol is already defined, the command displays an error message for duplicate definitions. (Example 12)
- If the name you specified in defining a label, symbol, or bit symbol is incorrect, the command displays an error message. (Example 13)
- If address FFFFh is exceeded, the command displays an assemble error even though the command syntax may be all right. (Example 15)
- When address FFFFh is reached, the command A is halted. (Example 16)
- When entering a value beginning with A, add '0' or '$' at the beginning or 'h' or 'H' at the end of the value to discriminate it against the accumulator. (Example 18)
- Bit symbols can be used. (Example 19)
- A bit symbol can be used as only a bit value or address value. (However, a bit symbol as only an address value can only be used in LDA or STA instruction.)
- The addressing mode can be specified as follows:
  - Immediate addressing (Example 20)
    When using this addressing, add '#' before the immediate data.
  - Zero-page addressing (Example 21)
    When using this addressing, specify a memory range between address 0 to address FF.
  - Absolute addressing (Example 22)
    When using this addressing, write the relevant address in the operand.
  - Relative addressing (Example 23)
    When using this addressing, write the relevant address in the operand directly as is.
  - Special page addressing (Example 24)
    When using this addressing, specify the special page address and add '¥' before the address.
  - Indirect addressing (Example 25)
    When using this addressing, add '0' before and after the address.
- When assembling a BRK instruction (1-byte instruction), the command writes op-code (0x00) and 0xEA (NOP) to memory. The assemble address is updated for 2 bytes. (Example 26)

## Parameter

- An expression can be used to write the address. If the address is omitted, the program is assembled beginning with the address where the command A finished processing in the previous execution.
- The range of values that can be specified for the address is 0000h to FFFFh.

## *Command execution example*

```
>A E0000                                          (Example 1)
ADDRESS>  LABEL      PROGRAM
E000      START:     SEI
E001                 BAT MNEMONIC                 (Example 2)
ERROR 1101: Assembly language description contains error.
E001                 BRA E083                     (Example 3)
ERROR 1102: Specified operand value is out of range.
E001                 Label:                       (Example 4)
E001      Label:
>A                                                (Example 5)
ADDRESS>  LABEL      PROGRAM
E001      Label:     .BLKB 5                      (Example 6)
E006                 .BYTE 0,'0',FF               (Example 7)
E009                 Symbol .Equ 0                (Example 8)
E009                 BitSymbol .Equ 0,0           (Example 9)
E009                 .WORD 0,'0',FFFF             (Example 10)
E00F                                              (Example 11)
>A
ADDRESS>  LABEL      PROGRAM
E00F                 Label:                       (Example 12)
ERROR 1508: Symbol/label of the same name is already registered.
E00F                 Symbol .EQU 0
ERROR 1508: Symbol/label of the same name is already registered.
E00F                 BitSymbol .Equ 0,0
ERROR 1508: Symbol/label of the same name is already registered.
E00F                 Lab-el:                      (Example 13)
ERROR 1505: Character string contains a character that cannot be used
in symbol/label.
E00F                 Sym-bol .EQU 0
ERROR 1505: Character string contains a character that cannot be used
in symbol/label.
E00F                 Bit-Symbol .EQU 0
ERROR 1505: Character string contains a character that cannot be used
in symbol/label.
E00F                                              (Example 14)
>A FFFF
ADDRESS>  LABEL      PROGRAM
FFFF      LAST_ADD: LDA #10                       (Example 15)
ERROR 1101: Assembly language description contains error.
FFFF      LAST_ADD: NOP                           (Example 16)
>A                                                (Example 17)
```

```
ERROR 1204: Maximum address reached in previous execution.
>A ACCUMULATOR
ADDRESS>  LABEL      PROGRAM
E010      ACCUMULA:  INC A                      (Example 18)
E011                 INC 0A
E013                 INC $A
E015                 INC Ah
E017                 INC AH
>A BITSYMBOL
ADDRESS>  LABEL      PROGRAM
E020      BITSYMBO: SEB 0,10                    (Example 19)
E022                 SEB BitSymbol
E024                 SEB BitSymbol,10
E026                 SEB 1,A
E027                 SEB BitSymbol,A
E028                 LDA BitSymbol
E02A                 STA BitSymbol
E02C
>A IMMEDIATE
ADDRESS>  LABEL      PROGRAM
E030      IMMEDIAT: LDA #10                     (Example 20)
E032                 LDA #Symbol
E034
>A ZEROPAGE
ADDRESS>  LABEL      PROGRAM
E040      ZEROPAGE: LDA 10                      (Example 21)
E042                 LDA ZeroPage
E044
>A ABSOLUTE
ADDRESS>  LABEL      PROGRAM
E050      ABSOLUTE: LDA E090                    (Example 22)
E053                 LDA Label
E056
>A RELATIVE
ADDRESS>  LABEL      PROGRAM
E060      RELATIVE: BRA E000                    (Example 23)
E062                 BRA Label
E064
>A SPECIAL
ADDRESS>  LABEL      PROGRAM
E070      SPECIAL:   JSR ¥FF70                  (Example 24)
E072                 JSR ¥SPECIALPAGE
E074
>A INDIRECT
ADDRESS>  LABEL      PROGRAM
E080      INDIRECT: LDA (70,X)                  (Example 25)
E082                 LDA (ZeroPage,X)
E084
>A BRK_OPERATION
ADDRESS>  LABEL      PROGRAM
E090      BRK_OPER: BRK                         (Example 26)
E092
```

# BIT (set BIT)

**References/sets bit symbols.**

## *Input format*

BIT bit symbol name

## *Run-time execution*

Can be executed.

## *Function*

- This command displays or modifies the value of a specified bit symbol.

## *Parameter*

- A bit symbol can be used to write the bit symbol name. (Expressions cannot be used.)
- An expression can be used to write the value you want to be modified. The range of values to be modified is 0 to 1.

## *Command execution example*

```
>BIT BITSYMBOL<RET>
0,0005h      1    0<RET>
>
```

- In this execution example, the bit indicated by the bit symbol BITSYMBOL (bit 0 at address 0005h) has its value changed to 0.

# CD

**Sets/references current directory.**

## *Input format*

CD [directory name]

## *Run-time execution*

Can be executed.

## *Function*

- This command sets or references the current directory.

## *Parameter*

- If the parameter is omitted, the command displays the current directory.

## *Command execution example*

```
>CD
c:¥usr
>cd ¥usr¥pdb38m
c:¥usr¥pdb38m
```

Example where the program you want to be downloaded (e.g., program.hex) exists in c:¥usr¥pdb38m¥program

```
>CD ¥usr¥pdb38m¥program                    (Example 1)
c:¥usr¥pdb38m¥program
>I program
program.hex Now Loading.
program.hex Load end.
program.sym Now Loading.
program.sym Load end.
```

**Supplements**

- When using the I, o, logon, src, or script command to specify a file name, you can use a relative path name from the directory specified by this command. (Refer to Example 1.)
- If you choose a file in the download dialog box of menu [File] -> [Download], the current directory is set to the directory where your selected file exists.

# D (Dump)

**Displays memory contents.**

## *Input format*

D [start address] [,end address]

## *Run-time execution*

Can be executed.

## *Function*

- This command displays memory contents of a specified address range along with corresponding ASCII data.
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.

## *Parameter*

- An expression can be used to write the start address. If the start address is omitted, memory contents are displayed beginning with an address next to the end address of the previous display. (If this command is executed for the first time, address E000h is the start address.)
- An expression can be used to write the end address. If the end address is omitted, 128 bytes of memory contents from the start address are displayed before finishing command execution. However, if address FFFFh is reached during this display, command execution is terminated before displaying full 128 bytes of memory contents.
- The range of values that can be specified for the start and end addresses is 0000h to FFFFh.

## *Command execution example*

```
>D E089, E0D0                                          (Example 1)
ADDRESS:   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F     ASCII
 E080:                          EA EA EA EA EA EA EA EA            ........
 E090:    EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA     ................
 E0A0:    EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA     ................
 E0B0:    EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA     ................
 E0C0:    EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA     ................
 E0D0:    EA                                                  .
>
```

# F (Fill)

**Fills memory block with data.**

### *Input format*

F start address, end address, data

### *Run-time execution*

Can be executed.

### *Function*

- This command writes specified data to a specified range of memory addresses.
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.

### *Parameter*

- Expressions can be used to write the start and end addresses and data. The specification is
- The range of values that can be specified for the start and end addresses is 0000h to FFFFh.
- The range of values that can be specified for data is 0h to FFh. Signed integers from -128 to 255 can also be specified.

### *Command execution example*

```
>F 20,5F,74<RET>                                    (Example 1)
>
DumpByte 2000, 200F<RET>
ADDRESS: +0 +1 +2 +3 +4 +5 +6 +7                    +8 +9 +A +B +C +D
+E +F    ASCII
 002000: 74 74 74 74 74 74 74 74                    74 74 74 74 74 74
74 74    tttttttttttttttt
>
```

- In Example 1, data 74h is written to a range of memory addresses from 20h to 5Fh.

# G (Go)

**Executes program.**

## *Input format*

G [start address] [,break address 1]

## *Run-time execution*

Cannot be executed.

## *Function*

- This command executes the target program beginning with a specified address.
- This command temporarily disables other break functions.
- Program execution breaks when the instruction at the specified break address is executed.

## *Parameter*

- Expressions can be used to write the start address, and break address.
- If the start address is omitted, the program is executed beginning with the address indicated by the current program counter.
- If the break address is omitted, no break occurs. (To break the target program in this case, input the STOP command.)
- The range of values that can be specified for the start and break addresses is 0000h to FFFFh.

## *Command execution example*

```
>G E000,E003                                    (Example 1)
>G                                              (Example 2)
>STOP
>
```

Example 1: The target program is executed beginning with address E000h and when the instruction at address E003h is executed, the target program is halted.

Example 2: The target program is executed without a break beginning with the address indicated by the current program counter.

# GB (Go with program)

**Executes program.**

## *Input format*

GB [start address]

## *Run-time execution*

Cannot be execute.

## *Function*

- This command executes the target program beginning with a specified start address. If the start address is omitted, the program is executed beginning with the address indicated by the current program counter.
- When executed by this command, the program is made to break at program breakpoints (alias software breakpoints) that are currently set.

## *Parameter*

- An expression can be used to write the start address.
- The range of values that can be specified for the start address is 0000h to FFFFh.

## *Command execution example*

```
>PB
ADDRESS label
E006
E009
>GB E000                                          (Example 1)
>GB                                               (Example 2)
>
```

Example 1: The target program is executed with a break beginning with address E000h.

Example 2: The target program is executed with a break beginning with the address indicated by the program counter.

# I (Input hex and sym file)

**Reads HEX and SYM files.**

## *Input format*

I file name [.hex | .sym]]

## *Run-time execution*

Cannot be executed.

## *Function*

- This command reads HEX and SYM files into the system. When you specify a file name (path name can be specified), the command reads a specified HEX file. And, if an SYM file of the same name exists in the same directory, the SYM file is also read in.
- By adding file attribute ".hex" or ".sym," you can read a HEX file or an SYM file independently.
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.

## *Parameter*

- For the file name, you can specify a HEX file and an SYM file. If the file attribute is omitted, both HEX and SYM files are read in.

## *Command execution example*

```
>I test                                          (Example 1)
test.hex Load end
test.hex Load end
>I test.hex                                      (Example 2)
test.hex Load end
>I test.sym                                      (Example 3)
test.sym Load end
```

Example 1: HEX and SYM files bearing file name "test" are read in.
Example 2: A HEX file bearing file name "test" is read in.
Example 3: An SYM file bearing file name "test" is read in.

# L (List)

**Disassembles memory contents one line at a time.**

## *Input format*

L [start address] [,end address]

## *Run-time execution*

Can be executed.

## *Function*

- This command displays memory contents from specified start address to end address after disassembling them. (Example 1)
- If there is no corresponding machine language, "???" is displayed in the program column. (Example 2)
- If there is a label, symbol, or bit symbol corresponding to the operand value, they are displayed along with the operand. (Example 5)
- When address FFFFh is reached, the command L is halted. (Example 6)
- If the operand data exceeds FFFFh, extra data beyond that is marked by "???." (Example 7)
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.
- When disassembling a BRK instruction (1-byte instruction), object code is displayed for 2 bytes and the disassemble address is updated for 2 bytes. (Example 9)

## *Parameter*

- An expression and keyword (_PC) can be used to write the start address. An expression can be used to write the end address.
  If you specify "_PC" for the start address, memory contents are displayed beginning with the current program counter. (Example 4)
- If the start address is omitted, memory contents are displayed beginning with the last address where the command L finished processing in the previous execution. (Example 3) However, if address FFFFh was reached in the previous execution, the command displays an error message. (Example 8) Note that at startup of PDB38M, memory contents are displayed beginning with the program counter.
- If the end address is omitted, memory contents are displayed for 10 minutes from the start address.

*Command execution example*

```
>L E000, E001<RET>                                  (Example 1)
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E000       A904        START_AD:   LDA #04H
>L E001, E002<RET>
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E001       04                      ???             (Example 2)
E002       3C1020                  LDM # 10H,20H
>L E006<RET>                                        (Example 3)
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E005       8D0001                  STA 0100H
>X<RET>
A=72 X=74 Y=77 F=N----I—S=0BF PC=E000 START_ADD
>L _PC,E005<RET>                                    (Example 4)
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E000       A904        START_AD:   LDA #04H
E002       3C1020                  LDM #10H,20H
E005       8D0001                  STA 0100H
>L E020,E03D<RET>
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E020       A930                    LDA #30H :Immediate (Example 5)
E022       2280                    JSR ¥FF80H :SpacialPage
E024       B704E9                  BBC 5H,04H,E010H
E027       B702E6                  BBC 5H,02H :ZeroPage,E010H
E02A       B7040D                  BBC 5H,04H,E03AH :BraAddr
>L FFFE<RET>                                        (Example 6)
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
FFFE       E6A9                    INC A9H
>L FFFF<RET>
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
FFFF       A9??        LAST_ADD: LDA #??H          (Example 7)
>L<RET>                                             (Example 8)
ERROR 1204: Maximum address reached in previous execution.
>L E080,E080<RET>
ADDRESS>   OBJ-CODE    LABEL       PROGRAM
E080       00EA        BRK                         (Example 9)
>
```

# LOGON,LOGOFF

**Outputs/stops outputting display screen to log file.**

## *Input format*

LOGON [file name [.file attribute]]
LOGOFF

## *Run-time execution*

Can be executed.

## *Function*

- The LOGON command writes the contents of the display screen to a log file.
- The LOGON command allows you to open multiple files. Note that the log file to which the display screen is written is the log file you opened last.
- The LOGOFF command disables the display screen from being output to a log file. If you execute this command when multiple log files are open, the log file that is opened last by the LOGON command is closed.
- Log files can be nested up to 8 levels.

## *Parameter*

- A log file name can be used to write the file name. Any desired name can be used for its file attribute. If the file attribute is omitted, a file attribute ".log" is automatically added to the file name.
  If the file name is omitted, the command opens the file that was logged-off immediately before. In this case, screen contents are written to the file thus opened starting after its last line. If no file is found that can be opened, a log file name is assumed after the time at which LOGON is executed. If LOGON is executed at 15:03 on December 16, for example, the log file name is "12161503.log."

## *Command execution example*

```
>LOGON            ←File name not specified ("12071333.log" is
                    opened)
>LOGON d.log      ←File name specified ("d.log" is opened)
>D E000,E100      ←――――――――――(output to "d.log")
    :
    :
>LOGON l.log      ←File name specified ("l.log" is opened)
>D E000,E100      ←――――――――――(output to "d.log")
```

```
    :
    :
>LOGOFF           ←"l.log" closed
>LOGOFF           ←"d.log" closed
>LOGON            ← File name not specified ("d.log" is opened)
>D E300,E400      ←————————————(output to "d.log" after its
                    last line)
    :
    :
>LOGON            ←File name not specified ("l.log" is opened)
>D E300,E400      ←————————————(output to "l.log" after its
                   last line)
    :
    :
>LOGOFF           ←"l.log" closed
>LOGOFF           ←"d.log" closed
>LOGOFF           ←"12071333.log" closed
>
```

# O (Output HEX file)

**Outputs HEX file.**

## *Input format*

O file name, start address, end address

## *Run-time execution*

Cannot be executed.

## *Function*

- This command stores data of a specified address range in a file name that is specified as an Intel HEX file.
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.

## *Parameter*

- An output file name can be used to write the file name. If the file attribute is omitted, the command assumes ".hex" is specified.
- Expressions can be used to write the start and end addresses.
- The range of values that can be specified for the start and end addresses is 0000h to FFFFh.

## *Command execution example*

```
>O TEST.HEX, E000,EFFF                          (Example 1)
>
```

# PB (Program Break)

**Sets/references program breakpoints.**

## *Input format*

PB
PB {SET | CLEAR} address

## *Run-time execution*

Cannot be executed.

## *Function*

- This command sets, clears, or references program breakpoints (alias software breakpoints).
- Program breakpoints, also called software breakpoints, are breakpoints set in the program area that cause program execution to break before instruction execution. Namely, a break occurs before the instruction at a specified break address is executed.
- When setting a breakpoint, input "PB SET address." To clear a breakpoint, input "PB CLEAR address." To reference a breakpoint, input simply "PB."

## *Parameter*

- An expression can be used to write the address.
- The range of values that can be specified for the address is 0000h to FFFFh.

## *Command execution example*

```
>PB SET E000                                    (Example 1)
>PB                                             (Example 2)
ADDRESS    LABEL
E000       START:
>PB CLEAR E000                                  (Example 3)
>PB
ADDRESS    LABEL
```

Example 1: A breakpoint is set at address E000h.
Example 2: The currently set breakpoint is referenced.
Example 3: The breakpoint at address E000h is cleared.

# RADIX

**Sets/references default radix of numeric input.**

## *Input format*

RADIX [ { 10 | 16 } ]

## *Run-time execution*

Can be executed.

## *Function*

- This command sets the default radix of constants input in an expression. Decimal or hexadecimal can be set. The default radix is hexadecimal.
- Even when you set a radix with this command, the numeration of constants can be individually specified by adding '@' or '$' that denotes decimal or hexadecimal numeration when entering a constant.
- The radix set by this command is effective in only script commands.

## *Parameter*

- For the default radix, you can specify either 10 (decimal) or 16 (hexadecimal).
- If the parameter is omitted, the command displays the currently set default radix.

## *Command execution example*

```
>RADIX 16                                          (Example 1)
>RADIX                                             (Example 2)
RADIX = 16
>RADIX 10                                          (Example 3)
>RADIX
RADIX = 10
>
```

Example 1: The default radix is set to hexadecimal.
Example 2: The currently set default radix is referenced.
Example 3: The default radix is set to decimal.

# S (SET)

**Displays/modifies memory contents.**

## *Input format*

| | | |
|---|---|---|
| S | [address] | (Hexadecimal representation) |
| SD | [address] | (Decimal representation) |
| SB | [address] | (Binary representation) |

## *Run-time execution*

Can be executed.

## *Function*

- This command displays or modifies the memory content at a specified address in units of bytes. The memory content can be displayed in binary, decimal, or hexadecimal as selected.
- To reference the preceding addresses, input "-" (minus) and hit the RET key.
- To terminate the command, input "." (period) and hit the RET key.
- If you do not modify data, enter only the RET key.
- If the start address is omitted, memory contents are displayed beginning with the last address referenced by the command S immediately before. (If the command is executed for the first time, memory contents are displayed beginning with address E000h.)

## *Parameter*

- An expression can be used to write the address. If the address is omitted, memory contents are displayed beginning with the last address referenced by the command S immediately before. (If the command is executed for the first time, memory contents are displayed beginning with address E000h.)
- An expression can be used to write data to be modified.
- The range of values that can be specified for the address is 0000h to FFFFh. The range of values that can be used for the data to be modified is 0 to FFh. Signed integers from -128 to 255 can also be specified.

## *Command execution example*

```
>S e000<RET>
```

```
ADDRESS    LABEL     0LD        NEW
E000       start:    EA         50<RET>
E001                 EA         30+20<RET>
E002                 EA         -<RET>         ←Returns to the
                                                 previous address.
E001                 50         20<RET>
E002                 EA         .<RET>         ←Terminated
>S<RET>
ADDRESS    LABEL     0LD        NEW
E002                 EA         00
E003                 EA         .<RET>         ←Terminated
>SD F000
ADDRESS    LABEL     0LD        NEW
E000                 210        2D
E001                 0          3E
E002                 EA         .<RET>         ←Terminated
>SB 0000
ADDRESS    LABEL     0LD        NEW
0000                 00000000   5F
0001                 00000000   -
0000                 01011111   .<RET>         ←Terminated
>
```

# SCOPE

**Sets/references effective range of local symbols.**

## *Input format*

SCOPE [relocatable file name]

## *Run-time execution*

Can be executed.

## *Function*

- This command sets a scope.
  A scope means an effective range of local labels or symbols. A scope is defined in units of object files (R74 files).
- If the relocatable file is omitted, the command displays the current scope

## *Parameter*

- A file name bearing the file attribute ".r38" can be used to write the module file name. A file attribute must always be specified.

## *Command execution example*

```
>SCOPE test.r74                                    (Example 1)
>
```

Example 1: File "test.r74" is specified for the scope.

***★★Precautions★★***

SCOPE processing

In PDB38M, the effective range is, by default, set to the scope indicated by the program counter. Furthermore, when a command is being executed, the scope is automatically switched over according to the addresses processed by the command and when command execution is completed, the current scope returns. Use the SCOPE command when you want to modify the scope temporarily.

# SCRIPT

**Opens/executes script file.**

## *Input format*

SCRIPT　script file name

## *Run-time execution*

Can be executed.

## *Function*

- This command opens a script file.
  A script file is a text file that contains a description of script commands.
  Use the editor you have at hand to create a script file.
- When after opening a script file with this command, you click on the script window's **Run** button, the script commands written in the script file are automatically executed.
- This command can be written in a script file also.　Script files can be nested up to five levels.

## *Parameter*

- Any desired name can be used for the script file name.
  A file attribute must always be specified.

## *Command execution example*

```
>SCRIPT test.scr                                    (Example 1)
```

Example 1: File "test.scr" is opened as a script file.

# SI (Section Information)

**Displays section information.**

## *Input format*

SI

## *Run-time execution*

Can be executed.

## *Function*

- This command displays information on sections.
- The section information displayed here consists of NAME, OBJECT, TYPE, START, LENGTH, SOURCE, and LIBRARY from left to right.
  - Section name (Item name: NAME)
  - Object file name (r.74) (Item name: OBJECT)
  - Physical attribute (ROM/RAM) (Item name: TYPE)
  - Start address (Item name: START)
  - Number of types in section area (Item name: LENGTH)
  - Source file name (.a74) (Item name: SOURCE)
  - Library file name (Item name: LIBRARY)

## *Parameter*

- None

## *Command execution example*

```
>SI<RET>
NAME       OBJECT         TYPE    START   LENGTH SOURCE      LIBRARY
RAMAREA    sample0.r74    RAM     0000    0100   sample0.a74
PROG1      sample1.r74    ROM     E000    0074   sample1.a74
PROG1      sample2.r74    ROM     E074    0077   sample2.aA4
PROG2      sample1.r74    ROM     F000    0072   sample1.a74
>
```

# SRC（set SouRCe file）

**Changes program window display by source file name.**

## *Input format*

SRC　source file name

## *Run-time execution*

Can be executed.

## *Function*

- This command changes the source file name displayed in the program window.

## *Parameter*

- Specify the source file name (.a74) for the parameter.

## *Command execution example*

```
>SRC sample.a74                                    (Example 1)
>
```

Example 1: The source file name displayed in the program window is changed to "sample.a74."

# STOP

**Stops program execution.**

### *Input format*

STOP

### *Run-time execution*

Can be executed.

### *Function*

- This command forcibly stops the target program being executed.

### *Parameter*

None

### *Command execution example*

```
>G E000
>STOP
>
```

# T (Trace)

**Single-steps target program.**

## *Input format*

T [step count]

## *Run-time execution*

Cannot be executed.

## *Function*

- This command single-steps the target program by executing instructions a specified number of steps beginning with the current program counter.
- If the step count is omitted, the program is executed one step.
- Shown after '*' on the last line of the T command is the program counter value.

## *Parameter*

- An expression can be used to write the step count.
- The range of values that can be specified for the step count is 1 to 65,535.

## *Command execution example*

```
>T 3                                                (Example 1)
A=04 X=7F Y=EB F=-----I-- S=07F PC=E000 START
A=04 X=7F Y=EB F=---B-I-- S=07F PC=E001
A=04 X=7F Y=EB F=---B-I-- S=07F PC=E002
*E003
>
```

Example 1: The T command is executed after specifying step count = 3.
(The program counter value after executing the T command is address E003.)

# U (Untrace)

**Single-steps target program.**

## *Input format*

U [step count]

## *Run-time execution*

Cannot be executed.

## *Function*

- After displaying the status before it starts single-stepping, the command executes instructions a specified number of steps beginning with the current program counter.
- Shown after '*' on the last line of the U command is the program counter value.

## *Parameter*

- An expression can be used to write the step count.
- The range of values that can be specified for the step count is 1 to 65,535.
- If the step count is omitted, PDB38M assumes the default value (= 1).

## *Command execution example*

```
>U 3                                                (Example 1)
A=04 X=7F Y=EB F=-----I-- S=07F PC=E000 START
*E003
>
```

Example 1: The U command is executed after specifying step count = 3.
(The program counter value after executing the U command is address E003.)

# VER（VERsion）

**Displays version.**

## *Input format*

VER

## *Run-time execution*

Can be executed.

## *Function*

- This command displays PDB38M's version number.

## *Parameter*

None

## *Command execution example*

```
>VER<RET>
PDB38M Ver.V.1.00.00
>
```

# WAIT (WAIT commands till break)

**Keeps command input waiting until program breaks.**

## *Input format*

WAIT

## *Run-time execution*

Can be executed.

## *Function*

- This command keeps command input waiting until the target program breaks.
- Use this command when you want to execute the target program with breakpoints from a script file and execute commands subsequently after the program breaks.
- This command is ignored if it is issued when the target program is idle.
- Command execution can be halted by issuing the STOP command or clicking on the PDB38M window's **Stop** button when the command is being executed.

## *Parameter*

None

## *Command execution example*

```
>WAIT<RET>
>
```

# X (eXeamine register)

**References/modifies register values.**

## *Input format*

Format 1: X
Format 2: XA set value
: XX set value
: XY set value
: XS set value
: XF set value
: XP set value
: XA set value
Format 3: X register name
(Register name: {A | X | Y | S | F | P})

## *Run-time execution*

Cannot be executed.

## *Function*

- This command references or modifies register contents.
- In Format 1, input only the command name X. In this case, the command references the contents of all registers.
- In Format 2, input the command name X and a register name in succession (e.g., XA). Then write a value you want to be set to the register in the first parameter.
- In Format 3, input X as the command name and a register name in the first parameter. (Separate the command and the register names with a blank character.) In this case, the command waits for you to input a value you want to be set to the register (interactive input).

## *Parameter*

- For the register name, any of A (A register), X (X register), Y (Y register), F (F processor status register), S (S stack pointer), or P (P program counter) can be used.
- An expression can be used to write the set value.

- The range of values that can be specified for the set value varies with each register.

The table below lists the range of set values that can be for each register.

| Register | Minimum Value | Maximum Value |
|---|---|---|
| A,X,Y,F | 0h | FFh |
| S | 000h | 1FFh |
| P | 0000h | FFFFh |

For registers A, X, Y, and F, signed integers from -128 to 255 can also be specified.

*Command execution example*

```
>X A                                                    (Example 1)
04  72
>X X                                                    (Example 2)
7F  74
>XY FF                                                  (Example 3)
>X                                                      (Example 4)
A=72       X=74    Y=FF    F=---B-I--     S=07F   PC=E006
>
```

Example 1: The content of register A is changed from to 04h to 72h.
Example 2: The content of register X is changed from to 7Fh to 74h.
Example 3: The content of register Y is changed to FFh.
Example 4: The contents of all registers are referenced.

# Z (reset)

**Resets MCU in hardware.**

## *Input format*

Z

## *Run-time execution*

Cannot be executed.

## *Function*

- This command resets the target MCU in hardware.
- The program counter value is changed to the address indicated by the reset vector.

## *Parameter*

None

## *Command execution example*

```
>Z
>
```

# ; (comment line)

**Comment line**

### *Input format*

; [character string]

### *Run-time execution*

Can be executed.

### *Function*

- This command indicates that the character string following it is a comment. PDB38M ignores comments.

### *Parameter*

- Any characters can be written in a character string.
- The number of characters that can be written in a character string depends on the maximum size of the command input line (160 columns).

### *Command execution example*

```
>; Displays data table TABLE1 <RET>
>D TABLE1,TABLE1+8<RET>
>
```

# 15. Method for Writing Expressions

PDB38M allows you to use expressions when entering commands. Figure 15-1 shows several examples where commands are input using expressions.

```
>D TABLE1
>D TABLE1+20
>F TABLE1, LABEL1, 'C'
>F TABLE, LABEL1, 10
>PB SET #10.TEST.A74
>D TABLE1+20-#10*20-5
```

Figure 15-1 Example of commands input using expressions

The following explains how to write an expression.

## 15.1. Elements of An Expression

Following can be used as elements constituting an expression.

1. Label
   The labels defined at the beginning of lines of assembler SRA74M can be used. You only can use these labels after an SYM file is read into the system. Also, the labels defined by the line assemble command (command A) can be used.

1. Symbol
   The symbols defined by assembler SRA74M's pseudo-instruction ".EQU" can be used. You only can use these symbols after an SYM file is read into the system. Also, the symbols defined by the line assemble command (command A)'s pseudo-instruction ".EQU" can be used.

1. Constant
   A binary, decimal, or hexadecimal constant can be used.

1. Operator
   Arithmetic operators (add, subtract, multiply, and divide) and bit manipulating operators can be used.

1. Line number
   Line numbers in the program area (the area containing executable instruction code) can be used.

Each element of an expression is detailed in the pages to follow.

# 15.2. Labels and Symbols

## 15.2.1. Rules for Writing Labels and Symbols

Write a label or symbol name directly as it is.

- Alphanumeric characters, underline (_), period (.), and question mark (?) can be used. However, the first character cannot be a numeral; it must be an alphabet.
- Label and symbol names are case-sensitive; uppercase and lowercase letters are discriminated.
- Register names (A, X, Y, S, PC, PS, and P) cannot be used.
- The number of characters that can be used in a label or symbol name depends on specifications of assembler SRA74M (255 characters).
- Assembler SRA74M's structured instructions, pseudo-instructions, macro instructions, and op-code cannot be used. (For example, these include .SECTION, .BYTE, .switch, if, etc.)
- Following cannot be used for labels and symbols:

| | | |
|---|---|---|
| .D0 | ～ | .D65535 |
| .F0 | ～ | .F65535 |
| .I0 | ～ | .I65535 |
| .S0 | ～ | .S65535 |
| ..0 | ～ | ..65535 |
| ??0 | ～ | ??65535 |

Indicates a command name and the outline function of each command.

## 15.2.2. Local Labels/Symbols and Scope

PDB38M supports two types of labels and symbols: global labels and symbols that can be referenced from anywhere in the program, and local labels and symbols that can be referenced in only the declared file.

The effective range of local labels and symbols is called a "scope." A scope is defined in units of object files (.R74 files). PDB38M switches over the scope depending on the situation as follows:

1. When entering a command

   The object file that contains the address indicated by the program counter is made the current scope. If you set a scope with the SCOPE command, the scope you have set becomes effective.

2. When executing a command

   The current scope is automatically switched over according to the program address handled by the command.

### 15.2.3. Priority of Labels and Symbols

Conversion from values into labels/symbols and conversion from labels/symbols into values are performed following the priority of labels and symbols shown below.

1. When converting address values

(a) Local labels
(b) Global labels
(c) Local symbols
(d) Global symbols
(e) Local labels outside scope
(f) Local symbols outside scope

2. When converting data values

(a) Local symbols
(b) Global symbols
(c) Local labels
(d) Global labels
(e) Local labels outside scope
(f) Local symbols outside scope

3. When converting bit values

(a) Local bit symbols
(b) Global bit symbols
(c) Local bit symbols outside scope

## 15.3. Constants

A binary, decimal, or hexadecimal constant can be used in an expression. The radix of a numeral is identified by adding one of marks listed in the table below at the beginning or end of the numeral.

Write a label or symbol name directly as it is.

Table 15-1 Radix of Constants

| Numeration | Notation when default radix is decimal | Notation when default radix is hexadecimal |
|---|---|---|
| Binary | %10010 or 10010B | %10010 |
| Decimal | 1234 | @1234 |
| Hexadecimal | $AB24 or 0AB24H | 0AB24, AB24, $AB24, or 0AB24H |

If the mark is omitted, numerals are processed as being hexadecimal numbers. However, when entering expressions in a script command, the default radix (decimal or hexadecimal) set by the RADIX command has priority. Note that if a hexadecimal number begins with an alphabet "A" to "F" and equals a label or symbol name, it is handled as a label or symbol.

## 15.4. Operators

The tables below list the operators that can be used in expressions.

Table 15-2 Operators Usable in Expressions (Execution and Stopping Execution)

| Operator | Meaning | Priority Level |
|---|---|---|
| ( | Left parenthesis | Level 1 |
| ) | Right parenthesis | |
| Monadic operator | | Level 2 |
| * | Multiply | Level 3 |
| / | Divide | |
| + | Add | Level 4 |
| -- | Subtract | |
| >> | Right | Level 5 |
| << | Left | |
| & | AND of bits | Level 6 |
| ^ | EOR of bits | Level 7 |
| \| | OR of bits | Level 8 |

Table 15-3 List of Monadic Operators

| Operator | Meaning | Priority Level |
|---|---|---|
| * | Positive | Level 2 |
| -- | Negative | |
| ~ | NOT of bits | |
| ! | Logical NOT | |

- Level 1 has the highest priority of operation, and Level 8, the lowest priority.
- An arithmetic expression is operated on in order of priorities beginning with the highest.

## 15.5. Line Numbers

Figure 15-2 shows a format for entering a line number.

```
# line number
# line number. "file name. file attribute
```

Figure 15-2 Input format of line number

- Decimal numbers can be used in line numbers.
- Source file line numbers can be omitted. If omitted, the source line numbers currently displayed in the program window are assumed to be the line numbers.

No blank character can be inserted between the line number, file name, and mark

# Part IV: Troubleshooting

# 16. Error Messages

The tables below list the error messages generated by PDB38M.

Table 16-1 Error Messages (No. 100 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 100 | Can't find Source File (%s). | |
| 101 | Unable to read Load Module File (%s). | |
| 102 | INTERNAL ERROR：File size information is illegal. | |
| 103 | Line number of Source File (%s) is over 65000. | |
| 104 | INTERNAL ERROR:Widget ID is illegal. | |
| 105 | INTERNAL ERROR：Can't create Bit Map. | |
| 106 | INTERNAL ERROR：Window ID is illegal. | |
| 107 | INTERNAL ERROR：Memory overflow. | |
| 108 | Name of Save File (%s) is illegal. | |
| 109 | Environment value 'MXDB38_HOME' isn't defined. | |
| 110 | Environment File is illegal. | |
| 111 | Can't write Environment File. | |
| 112 | Environment value 'MXDB38_HELP' isn't defined. | |
| 113 | Environment value 'MXDB38_MCU' isn't defined. | |
| 114 | Environment value 'MXDB38_LANG' isn't defined. | |
| 115 | Can't open more source window. | |
| 116 | Necessary environment definition isn't done. | |
| 117 | Can't execute new command during command processing. | |

Table 16-2 Error Messages (No. 150 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 150 | Can't open more %s window. | As many instances of the specified window as can be opened are already open. |
| 151 | Can't Create %s window. | The specified window cannot be opened. This is probably because memory is insufficient. Quit other applications or increase memory. |
| 152 | Can't open %s window, when the target program is running. | Stop the target program before you open the window. |
| 153 | Value is out of range. | The specified address exceeds the MCU's maximum address FFFFh |

Table 16-3 Error Messages (No. 200 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 200 | Can't change view mode. | The display start address does not match the source line address or the necessary source file cannot be found. |
| 201 | Can't find source file (%s). | The specified source file cannot be found. Specify the directory that contains the source file by using PATH command or menu [Environ] → [Path]. |
| 202 | Can't find search string (%s). | The specified character string could not be found although the entire location from the start to the end positions was searched. |
| 203 | Line number of Source File (%s) is over %d. | Since the source file contains many more lines than can be displayed, it cannot be displayed in source mode. The display mode is changed to the disassemble mode before the file contents are displayed. |

Table 16-4 Error Messages (No. 300 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 300 | Illegal endi. (%s %d line) | There is no *if* that corresponds to *endi*. |
| 301 | Illegal endw. (%s %d line) | There is no *while* that corresponds to *endw*. |
| 302 | INTERNAL ERROR : ER_BAT_EOF | |
| 303 | Script File is already exist. | |
| 304 | Can't find endi. (%s %d line) | There is no *endi* that corresponds to *if*. |
| 305 | Line length is overflow. (%s %d line) | The line contains more characters than can be written in one line. |
| 306 | Nest level is overflow. (%s %d line) | |
| 307 | Can't find Script File (%s). | |
| 308 | Can't read Script File (%s). | The remainder of the script file cannot be read. |
| 309 | Description is illegal. (%s %d line) | |
| 310 | Can't find endw. (%s %d line) | There is no *endw* that corresponds to *while*. |
| 311 | Nest level is overflow. (%d line) | |
| 312 | INTERNAL ERROR : ER_BAT_NONE | |

Table 16-5 Error Messages (No. 400 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 400 | Address value is out range for scroll area. | |

Table 16-6 Error Messages (No. 600 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 600 | Can't add new watch point because it exceeds limit of watch point number. Maximum number is (%d). | |
| 601 | Address value is out of range. | |
| 602 | Data value is out of range. | |
| 603 | Bit value is out of range | |

Table 16-7 Error Messages (No. 650 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 650 | There are no symbol information. | The load module file has not been loaded. |
| 651 | The expression is too long. | |

Table 16-8 Error Messages (No. 700 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 700 | Can't find file (%s) | |
| 701 | Message File (%s) is broken. | |
| 702 | Can't get enough memory. | |

Table 16-9 Error Messages (No. 900 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 900 | SYMBOL file is illegal. | The load module file contains a format error. |
| 901 | Loading is canceled. | |
| 902 | Can't find SYMBOL file(%s). | The load module file is nonexistent. |

Table 16-10 Error Messages (No. 1001 and over)(1/2)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1001 | Can't find symbol. | The specified symbol is nonexistent. |
| 1002 | Description of expression is illegal. | |
| 1004 | Description is illegal. | The expression contains a description error. |
| 1005 | Can't find scope. | The specified variable does not exist within the scope. |
| 1006 | Can't find symbol. | |
| 1007 | Can't find function. | The specified function is nonexistent. |
| 1008 | Right hand side of the expression is illegal. | |
| 1009 | The Type of structure(union) are not same. | |
| 1010 | Can't assign. | |
| 1011 | Can't find type. | The specified type is nonexistent. |
| 1012 | Not supported float(double) operation. | |
| 1013 | The operation does not be allowed to pointers. | |
| 1014 | The operation does not be allowed to the pointer. | |
| 1015 | Can't decrease by pointer. | |

Table 16-11 Error Messages (No. 1001 and over)(2/2)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1016 | Divided by 0. | |
| 1017 | The operator is not supported. | |
| 1018 | Type information is broken. | The load module file's symbol information contains an error. |
| 1019 | Left value must be the pointer. | |
| 1020 | Left value must be a structure or an union. | |
| 1021 | Can't find member. | |
| 1022 | Left value must be reference of a structure or an union. | |
| 1023 | Left value is illegal. | |
| 1024 | The operand must be a value. | |
| 1025 | The operand is able to be opposite sign. | |
| 1026 | Can't get address value. | |
| 1027 | The array variable is illegal. | |
| 1028 | The essential number of array is illegal. | |
| 1029 | The operand must be an address value. | |
| 1030 | Type casting for register variable is not be supported. | |
| 1031 | The type of type casting is illegal. | |
| 1032 | Type casting for that type is not be supported. | |

Table 16-12 Error Messages (No. 1070 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1078 | Can't open more memory window. | |
| 1080 | The address area is illegal. | |

Table 16-13 Error Messages (No. 1100 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1100 | Address value is out of range. | The specified address exceeds the MCU's maximum address value of FFFFh. |
| 1101 | Description of Assembly language is illegal. | |
| 1102 | Address value for JUMP is out of range. | |
| 1103 | Operand value is out of range. | |
| 1104 | Description of expression is illegal. | |

Table 16-14 Error Messages (No. 1200 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1200 | Start address is larger than end address. | |
| 1201 | Address value is out of range. | |
| 1202 | Data value is out of range. | |
| 1203 | Parameter (%d) is illegal. | |
| 1204 | Reached to maximum address in the previous execution. | The specified command cannot be used when PDB38M is executing a program. |
| 1205 | Can't find File (%s). | |
| 1206 | File (%s) is broken. | |
| 1207 | File Name is illegal. | |
| 1208 | Can't execution because it exceeds limit nest level of log file. Maximum level is (%d)." | |
| 1209 | File (%s) is already log on. | |
| 1210 | Can't open File (%s). | |
| 1211 | Can't get enough memory for LOGON command. | |
| 1212 | Can't execute because it exceeds limit nest level of script file. Maximum level is (%d). | Data cannot be saved to the specified file. |
| 1213 | Can't write File (%s) because there is not enough disk space. | An 8 bits wide bus cannot be accessed in 16 bits. |
| 1214 | Can't get enough memory. | The M38000T-SBI commands cannot be issued for M38000T-SBI. |
| 1215 | Address data is wrong. | |
| 1216 | Register value is out of range. | |
| 1217 | Can't execute this command because target type is unexpected. | |
| 1218 | Start cycle is larger than end cycle. | |
| 1219 | Can't open source file, or can't find source line. | |

Table 16-15 Error Messages (No. 1250 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1250 | The number of parameter is too many. | |
| 1251 | Unknown command. | |
| 1252 | Parameter (%d) value is out of range. | |
| 1253 | Parameter (%d) is illegal. | |
| 1254 | Input command line is too long. | |
| 1255 | Can't execute that command, when the target program is running. | |
| 1256 | Description of expression is illegal. | |
| 1257 | Target program is already stopped. | |

Table 16-16 Error Messages (No. 1300 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1300 | Line number is illegal. | |
| 1301 | Can't find right bracket ')'. | |
| 1302 | The Number of Macro constant is over the limit (%d). | |
| 1303 | Immediate value is out of range. | |
| 1304 | Prefix which gives radix of the constant is illegal. | |
| 1305 | Description of indirect reference is illegal. | |
| 1306 | Can't find end of strings (%s). | |
| 1307 | Description of expression is illegal. | |
| 1308 | Macro constant (%s) isn't defined. | |
| 1309 | Symbol (%s) isn't defined. | |
| 1310 | Immediate value is illegal. | |
| 1311 | Divide by 0. | |
| 1313 | The value is over the maximum value of which can be treated by MCU. | |

Table 16-17 Error Messages (No. 1400 and over)(1/2)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1400 | Address value is out of range. | The specified address exceeds the maximum value of FFFFh that the MCU can handle. |
| 1401 | Target program is already stopped. | |
| 1402 | The number of break point is over the limit (%d). | |
| 1403 | The break point isn't defined at that address. | |
| 1404 | Data value is out of range. | |
| 1406 | Can't read/write, because there are no memory at that area. | No reference or write can be performed on addresses where no memory is allocated. |
| 1407 | Can't get enough memory. | Memory is insufficient. Quit other applications or increase memory. |
| 1408 | Register value is out of range. | |
| 1409 | Can't execute that command, when the target program is running. | |
| 1410 | Start address is larger than end address. | |
| 1411 | STOP execution. | |
| 1412 | Can't find source lines which include that address. | There is no source line information at the specified address. |
| 1413 | That command has not yet supported. | |
| 1417 | Address value is out of range. | |
| 1418 | That baud rate has not yet supported. | |
| 1450 | Bit number is out of range. | |
| 1452 | STOP execution. | |
| 1453 | Data value is out of range. | |
| 1454 | Monitor File (%s) is broken. | |
| 1455 | Can't find File (%s). | Re-install the monitor file (filename). |
| 1456 | Target system is not constructed properly. | Make sure the monitor file is stored in the same directory as is PDB38M.EXE. |
| 1457 | INTERNAL ERROR:ER_IN2_ILLEGAL_MODE has happen. (in %s) | The combination of PDB38M, M38000T-SBI, and pod is incorrect. |
| 1459 | Mask value is out of range. | |

Table 16-18 Error Messages (No. 1400 and over)(2/2)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1460 | Counter of measurement time is overflow. | |
| 1461 | The version of PDB38M and the firmware on the target are not same. | The firmware installed in the target is newer than PDB38M. Download the firmware that matches the PDB38M version. |
| 1462 | Pass count value is out of range. | |
| 1463 | Can't execute that command, when the target program is running. | |
| 1464 | Target MCU is reset state. Please reset target systems. | Reset the target system. |
| 1465 | Target MCU is unable to reset. Please reset target systems. | Reset the target system. |
| 1466 | Target MCU is HOLD state. Please reset target systems. | |
| 1467 | Target MCU is not given clock. Please reset target systems. | |
| 1468 | Target MCU is not given power. Please reset target systems. | Reset the target system. |
| 1469 | INTERNAL ERROR:Break point number is illegal. | |
| 1470 | Please download the firmware to target. | The firmware version is automatically updated. |
| 1471 | Can't download firmware. | Start up the M38000T-SBI in maintenance mode before you start up PDB38M. The firmware version is automatically updated. |
| 1472 | Download firmware is finished. Please restart PDB38M. | Restart PDB38M. |
| 1473 | Can't find trace data which is able to refer. | |
| 1474 | Cycle value is out of range. | |
| 1475 | Target MCU is not under control. Please reset target systems. | Reset the target system. |
| 1476 | First data is larger than second data. | |
| 1477 | First address is larger than second address. | |
| 1478 | No event set on the state transition path. | |
| 1479 | Time out value is out of range. | |
| 1480 | Process ID value is out of range. | |
| 1481 | Cannot execute PDB38M because Emulator's version is old. Please version up Emulator. | |
| 1482 | The MCU file does not match POD. Please check MCU file name. | |

Table 16-19 Error Messages (No. 1500 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1500 | Address value is out of range. | The specified address exceeds the maximum value of FFFFh that the MCU can handle. |
| 1501 | Bit number is out of range. | |
| 1502 | File (%s) is broken. | |
| 1503 | Can't find File (%s). | |
| 1504 | Can't find sub routine information. | Recreate the target program after adding options to output debug information. |
| 1505 | Illegal character in the strings. | |
| 1506 | INTERNAL ERROR:ER_IN2_ILLEGAL_MODE has happen. (in %s) | |
| 1507 | Can't find that line number. | |
| 1508 | Multiple definition of symbol/label. | |
| 1509 | There are no code at that line. | No machine language is generated at the address that corresponds to the specified line number. |
| 1510 | Can't get enough memory. | |
| 1511 | Can't find scopes. | |
| 1512 | Can't find section information. | Recreate the target program after adding options to output debug information. |
| 1513 | Can't find source lines which correspond to that address. | |
| 1514 | Can't find symbol (%s). | |
| 1515 | Can't find the scopes which include that address. | |
| 1516 | Loading is canceled. | |

Table 16-20 Error Messages (No. 1700 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 1700 | INTERNAL ERROR:Already connected with the target. | |
| 1701 | INTERNAL ERROR:Fork error has happen. | |
| 1702 | Can't find Host Name (%s). | |
| 1703 | INTERNAL ERROR:The Baud rate is illegal. | |
| 1704 | The connection with the target isn't created. | |
| 1705 | Can't connect with the target. | |
| 1706 | INTERNAL ERROR:The Time of time out is out of range. | |
| 1707 | Time Out ERROR. | A time-out error has occurred when communicating with the target system. |
| 1708 | INTERNAL ERROR:Can't disconnect with the target. | |
| 1709 | INTERNAL ERROR:Can't send given size data. | |
| 1710 | INTERNAL ERROR: Parameter is illegal. | |
| 1711 | Illegal Host Name. | |
| 1712 | Communication ERROR. The connection with the target is closed. | The emulator was disconnected from the target system when communicating with it |
| 1713 | Communication ERROR. Can't send data. | A communication error has occurred when transferring data to the target system. |
| 1714 | Communication ERROR. Can't accept data. | A communication error has occurred when receiving data from the target system. |
| 1715 | Target is already used. | |

Table 16-21 Error Messages (No. 2400 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 2400 | Address value is out of range. | |
| 2401 | Data value is out of range. | |
| 2402 | Start Address value is out of range. | The value specified for the end address is smaller than the start address. |
| 2403 | Value is under ( %d ). | Specify a value that is equal to or smaller than (num). |
| 2404 | Data value is out of range. | |

Table 16-22 Error Messages (No. 5200 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 5200 | Can't execute Come command, without unused break point. | Remove some breakpoints before you execute Come. |

Table 16-23 Error Messages (No.5500 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 5500 | Value is out of range. | |

Table 16-24 Error Messages (No. 5700 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 5700 | The data value is too large. | |
| 5701 | The address area is illegal. | |
| 5702 | Address value is out range for scroll area. | The address specified for a scroll range exceeds the maximum address of FFFFh that the MCU can handle. |

Table 16-25 Error Messages (No. 5800 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 5800 | Sampling period value is out of range. | |
| 5801 | Address value is out of range. | |
| 5802 | Can't change RAM monitor area, when the target program is running. | Stop executing the target program before you change the RAM monitor area. |

Table 16-26 Error Messages (No. 5900 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 5900 | Can't open Script File (%s). | |
| 5901 | Script File is not open. | |
| 5902 | Can't open Log File (%s). | |
| 5903 | Can't open more Log File. | |
| 5904 | Can't open Log File. | |
| 5905 | File (%s) is already log on. | |
| 5906 | Can't open View File (%s) for new/add. | |

Table 16-27 Error Messages (No. 10045 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 10045 | INTERNAL ERROR:ER_ENV_END | |
| 10046 | Can't find directory (%s). | |
| 10047 | Can't create environment definition directory. | |
| 10048 | Can't open environment definition file. | |
| 10049 | Can't find environment definition directory. | |
| 10050 | Can't find MCU File (%s). | |
| 10051 | Can't find Watch File (%s). | |
| 10052 | Can't find Script File (%s). | |

Table 16-28 Error Messages (No. 10055 and over)

| No. | Error Message | Explanation/Corrective Action |
|---|---|---|
| 10055 | Can't open View File (%s) for new/add. | |
| 10056 | Can't write View/Log File because there is no enough disk space. | |
| 10057 | Can't find Script File (%s). | |
| 10058 | Can't execute because it exceeds limit nest level of log file. Maximum level is (%d). | |
| 10059 | File (%s) is already log on. | |
| 10060 | Can't open File (%s). | |
| 10061 | Can't get enough memory. | |
| 10062 | Can't write File (%s) because there is no enough disk space. | |
| 10063 | Script window is already opened. | |
| 10064 | Can't open more dump window. | |
| 10066 | There is no data. | |
| 10068 | There is no address. | |
| 10069 | There is no start address. | |
| 10070 | There is no end address. | |
| 10071 | Write area is illegal. | |
| 10072 | There is no Move address. | |
| 10073 | Unknown address size. | |
| 10074 | The area of display isn't mapped. | |
| 10075 | Can't open window because Map definition denied. | |

# Index

## Index

M3xxxxT-PAC User's Manual
PDB38M CONTROL SOFTWARE

The First Edition Second Impression : January 16th, 1997
The Second Edition First Impression : August 3rd, 2001
Document No. MSD-740PAC-UCE-010803


RENESAS
Renesas Technology Corp.

# M3XXXXT-PAC User's Manual
# PDB38M Control Software

MITSUBISHI ELECTRIC CORPORATION
MITSUBISHI ELECTRIC SEMICONDUCTOR APPLICATION ENGINEERING CORPORATION

RENESAS
Renesas Technology Corp.