MITSUBISHI SEMICONDUCTORS

Debug Control Software for the MELPS 740 Series RTT74 Ver 2.10 Second Edition

USER'S MANUAL

MITSUBISHI
ELECTRIC

USER'S MANUAL

Debug Control Software for the MELPS 740 Series
RTT74 Ver 2.10 Second Edition

MITSUBISHI ELECTRIC

Debug Control Software for the MELPS 740 Series

RTT74 Ver. 2.10

User's Manual

Second Edition

Mitsubishi Electric Corporation
Mitsubishi Electric Semiconductor Software Corporation

This manual applies to RTT74 VER. 2.10 Release 1.

First Edition, November 16, 1989
Second Edition, April 1, 1991

The contents of this user's manual may change due to modification of the product. Information, figures and data described in this manual are correct and accurate. Infringement of third parties' patent rights due to usage of the contents described in this manual are not the responsibility of Mitsubishi Electric Semiconductor Software Corporation.

If you have any questions or suggestions regarding the contents of this manual or the software, contact our Technical Support departments at one of the following addresses by fax or mail.

MAIN OFFICE

Mitsubishi Electric Semiconductor Software Corporation
Development Support Tool Section, Application Engineering Department
1-18 Toyotsu-cho, Suita City
Osaka, 564 Japan

PHONE	(06) 338-7066
FAX	(06) 338-5264

TOKYO MAIN OFFICE

Application Technical Department
Microcomputer Technical Section, Development Tool Software Support
Mitsubishi Electric Semiconductor Software Corporation
Gotanda NT building
1-18-9 Mishi-Gotanda, Shinagawa-ku
Tokyo, 141 Japan

PHONE	(03) 3490-6857
FAX	(03) 3490-7524

Document Number: MSD-RTT74-U

PREFACE

RTT74 is designed to control emulator units (PC4000E option board or PC4600 system) from a personal computer over a serial line.

It loads symbolic files and machine language data files generated by the MELPS 740 Series structured relocatable assembler SRA74 and provides symbolic debugging functions. Source lines can be displayed and specified if the symbol file contains source line number information (this is referred to as source line debugging).

This manual describes the functions and operations of RTT74.

WHAT YOU NEED TO USE RTT74

RTT74 runs under MS-DOS V. 2.11 and subsequent releases. The minimum memory requirement is 256 K bytes (user memory). Additional memory may be required to debug large programs.

The reader is assumed to have sufficient knowledge of the MELPS 740 microcomputer and MS-DOS. Refer to the MS-DOS manual for more information concerning the following topics:

1. Starting MS-DOS
2. Environment variables
3. Directory path, command path
4. Copying files
5. Editor

PRODUCT DESCRIPTION

The RTT74 package contains the following items:

1. Program disk
2. RTT74 User's Manual (this Manual).

If any of these items is missing, please contact your dealer. The contents of the program disk are as follows:

- RTT74. EXE
  A program which controls the MELPS 740 debugger.

- RTT74. HLP
  This file is used when RTT74 displays the help screen.

- RTT74. DAT
  This file is required to start RTT74 when using an option board.

- Mxxxxx.MCU
  This file contains data specific to each MCU. It is required when using the PC4600 system.

PROGRAM DISK BACKUP

Copy the contents of the original disk to your current system disk and store the original program disk in a safe place. This disk will be used when the program is updated. The use of this program is limited to a registered user, or anyone who has been authorized by a registered user. It must be used on a single computer at any one time.

Use the MS-DOS "copy" command to copy the files.

INSTALLING RTT74 ON A HARD DISK

RTT74 can be copied to a hard disk for use on a hard disk system. Copy the contents of the program disk to the command file directory of your hard disk.

Use the MS-DOS "copy" command to copy the files.

CONTENTS OF THIS MANUAL

A MELPS 740 Series debugging system is available for the following systems:

- PC4000E and option board (hereafter referred to as option board system)
- PC4600 system

RTT74 control software supports both of these systems.

This manual first describes the use of RTT74 for an option board system and then for a PC4600 system.

This manual is organized as follows:

- **Part I: Option Board System**

  This part describes the functions of RTT74 when connecting an option board system.

  Chapter 1. Overview This chapter describes the basic functions of RTT74.

  Chapter 2. Operation This chapter describes how to start and end RTT74, use of special function keys, and provides a command overview.

  Chapter 3. Commands This chapter describes the command syntax of each command and provides program examples.

  Chapter 4. Using Symbol Files This chapter describes the use of the symbolic debugging function and source line debugging function with actual execution examples.

- **Part II: PC4600 System**

  This part describes the functions of RTT74 when connected a PC4600 system.

  Chapter 5. Overview

  Chapter 6. Operation

  Chapter 7. Commands This chapter describes the commands which have different specifications from the option board system commands and additional commands.

  Chapter 8. Basic Debugging Command Operation Example This chapter describes the use of basic debugging commands when using a PC4600 system with sample programs.

- **Appendices**

  Appendix A: Error Messages This appendix describes the error messages that are displayed by RTT74 and the necessary actions.

  Appendix B: Scope of Command Address

  Appendix C: Baud Rate Setting This appendix describes how to set the baud rate of the PCs supported by RTT74.

Appendix D: Cable Connection This appendix describes how to connect the PCs supported by RTT74 to PC4000E.

Appendix E: RTT74 Specifications This appendix describes the standard MS-DOS environment under which the specifications where measured and the RTT74 specifications under that environment.

HOW TO USE THIS MANUAL

Read the following sections depending on the debugging system you are using.

- Option board system
  Read Part I and the Appendices.

- PC4600 system
  Read Part II and the Appendices. However, refer to Part I for the description of the following commands that are common with Part I.

  A, D, F, I, L, M, O, QUIT, S, SCOPE, SI, SHOW SOURCE, T, U, X, Z, ?, !, ;

TABLE OF CONTENTS

PART I. Option Board System

PART II. PC4600 System

APPENDIX

LIST OF FIGURES

PART I

PART II

APPENDIX

LIST OF TABLES

PART I

PART II

APPENDIX

LIST OF COMMANDS

The following table lists the RTT74 commands. (OPT) indicates option board system support commands in Part I and (PC4600) indicates PC4600 system support commands in Part II.

Commands	Function	Page
A	Assembles line-by-line	I-16
BP (PC4600)	Specifies and displays breakpoints (batch format) (menu format)	II-21 II-25
CV (PC4600)	Seta and displays coverage analysis	II-27
D (PC4600)	Displays memory contents in hexadecimal and as ASCII data	I-18
F	Writes specified word data in specified memory area	I-19
G (OPT)	Executes a program. Breakpoint can be set.	I-20
(PC4600)	Executes a program without breakpoints.	II-29
GB (PC4600)	Executes a program under break condition set with the BP command.	II-30
GL (OPT)	Controls program execution and break with source file line number.	I-22
I	Loads the contents of a file (HEX, SYM) into RTT74.	I-24
L	Disassembles and displays contents of program memory.	I-26
M	Performs block transfer of memory contents.	I-28
MAP (PC4600)	Sets and references mapping information in debug memory space.	II-31
MCU (PC4600)	Sets target MCU specific information.	II-33
O	Stores program memory contents into a file (HEX file).	I-29
P (OPT)	Displays and changes break counter.	I-30
QC (OPT) (PC4600)	Sets the realtime trace of break conditions.	I-31 II-35
QD (OPT) (PC4600)	Displays realtime trace results (memory dump).	I-32 II-37
QL (OPT) (PC4600)	Displays realtime trace results (disassembly).	I-36 II-40
QP (PC4600)	Specifies and displays realtime trace points. (batch format) (menu format)	II-42 II-46
QUIT	Ends RTT74.	I-39
S	Displays and changes memory contents.	I-40
SCOPE	Specifies scope of local identifier.	I-41
SHOW SOURCE	Displays source list and corresponding address.	I-43
SI	Displays section information.	I-46
STOP (PC4600)	Stops execution of target MCU and returns from runtime command mode to normal command mode.	II-48
T	Executes one instruction at a time and displays the internal status.	I-47
U	Executes only the specified number of instructions. (Displays only internal status before execution).	I-48
X	Displays and changes register contents.	I-49
Z	Resets target MCU.	I-50
?	Displays a list of RTT74 commands and their usage.	I-51
!	Executes an MS-DOS command from RTT74.	I-53
;	Indicates comment line input.	I-54

PART I. OPTION BOARD SYSTEM

CHAPTER 1. OVERVIEW

RTT74 controls an option board system (PC4000E, MELPS 740 Series option board) from a personal computer running under MS-DOS and enables debugging of MELPS 740 series application programs.

RTT stands for the Real-Time Trace function. However, this program can also be used with option boards without realtime trace functions.

1.1 Functions

RTT74 controls an option board system from a PC and enables the following operations:

- Load assembled object file into emulator memory and execute the program.
- Display and modify MCU registers and memories.
- Halt program execution under predefined condition and re-execute after examining the result.
- Trace, MCU output address, MCU executed codes, and external signal status in realtime and display them after program terminates. (Realtime trace function)

1.2 Input files

The following files are used by RTT74:

1. Hexadecimal file (HEX file)

This file contains machine language data generated by LINK74. File extension is .HEX.

2. Symbol file (SYM file)

This file contains symbol information and source line debugging information. A SYM file is generated when the "-S" option is specified when linking. File extension is .SYM.

3. Data file (DAT file)

This file contains data specific to each option board. This file is loaded during startup and is used to initialize the operating mode. The file name is RTT74.DAT.

4. Help screen file (HLP file)

This is a normal text file that contains help screens describing how to use RTT74. The file name is RTT74.HLP. The screens consist of a list of commands and a detailed description of each command.

1.3 System Configuration

1.3.1 Hardware Configuration

The following devices are required in order to use RTT74:

1. Personal computer
2. PC4000E (Debugger Main Unit)
3. PC4000P (Power Supply)
   One power supply for the PC4000E is required.
4. MELPS 740 series option board
5. Serial cable
   This is included when the PC4000E is purchased.

Figure 1.1 Shows the configuration of these devices.

1.3.2 Initial Setting of the Personal Computer

A personal computer and a PC4000E communicate using serial I/O. For fast RTT74 processing, it is necessary to set the baud rate of the personal computer and the PC4000E at 9600 bps.

1. The setup procedures for personal computers differ with each model. Refer to Appendix C, which lists the setup procedures for the personal computers supported by RTT74.
2. The PC4000E baud rate can be selected by removing the front panel to access the jumper switches on the main board. For specific instructions, refer to the PC4000E User's Manual.

1.3.3 Cable Connections

A cable must be connected between the personal computer and the PC4000E to transfer the data through the serial I/O. Cable connections differ with each personal computer model. For cable connection procedures, refer to Appendix D.

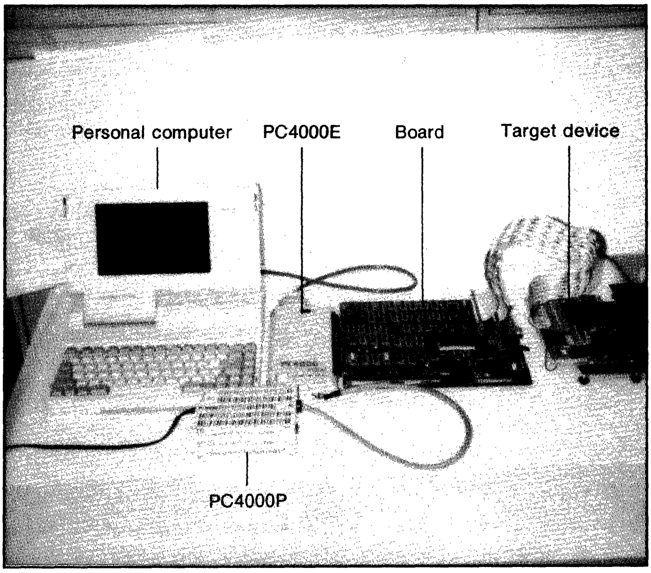


Figure 1.1 System Configuration

CHAPTER 2. OPERATION

2.1 Starting the Program

2.1.1 Normal Startup

RTT74 is started from the MS-DOS command prompt after turning on the PC4000E power.

2.1.2 Startup Options

RTT74 provides options shown in Table 2.1. The options can be specified in uppercase or lowercase.

Table 2.1 Startup Options

Option	Description
-HOST	Specifies the host computer on which RTT74 executes. Either "PC9801" or "IBMPC" can be specified. Example: A>RTT74 -HOST=PC9801
-I	Specifies the directory in which the DAT file and HLP file reside. Example: A>RTT74 -IA:\MSC\TOOL
filename	Specifies the name of the file to be loaded during startup. If the file extension is omitted, Files with extensions.SYM and .HEX are loaded. Example: A>RTT74 -HOST=PC9801 SAMPLE

2.1.3 Normal Startup Screen

The following messages are displayed and the normal command mode is entered when RTT74 starts up normally.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> IBM PC/XT/AT

DEBUGGER SYSTEM   ---> PC4000E
MONITOR           ---> V.1.3E
MCU               ---> M37450S4(MICROPROCESSOR MODE)
     bank0       0XXX  Internal     bank1 2000  Internal
     bank2       4000  Internal     bank3 6000  Internal
     bank4       8000  Internal     bank5 A000  Internal
     bank6       C000  Internal     bank7 E000  Internal
```

] ← Prompt indicates RTT74 is waiting for command input (If RTT function is not available the prompt is "–")

2.1.4 Startup Screen when a File Name is Specified

The following screen is displayed when the load file is specified during startup. If the file extension is omitted, files with extensions .SYM and .HEX are loaded.

```
A>RTT74 TEST<RET>   ← File name (extension may be omitted)
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> IBM PC/XT/AT

DEBUGGER SYSTEM  ---> PC4000E
MONITOR          ---> V.1.3E
MCU              ---> M37450S4(MICROPROCESSOR MODE)
     bank0      0XXX  Internal     bank1 2000  Internal
     bank2      4000  Internal     bank3 6000  Internal
     bank4      8000  Internal     bank5 A000  Internal
     bank6      C000  Internal     bank7 E000  Internal

GLOBAL SYMBOL LOADED
LOCAL  SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
TEST.HEX TEST.SYM LOAD END --> 740 SYMBOLS DEFINED
]
```

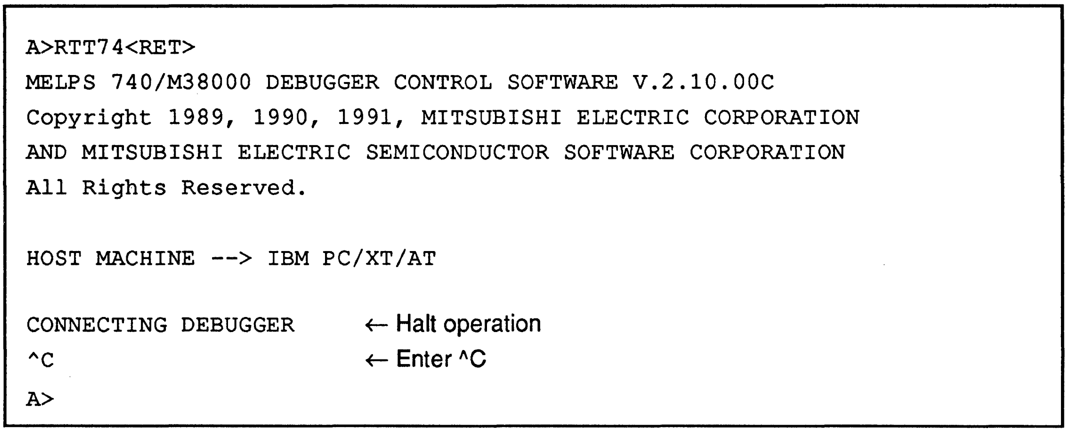
2.1.5 Compatible PC Startup Screen

When using a compatible PC supported by RTT74, the PC model can be specified as follows.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
Can't support this host machine.

   SELECT NEXT COMPATIBLE MACHINE TYPE.

       NEC PC-98 SERIES    ..... 1
       MULTI16-IV          ..... 2
       IBM PC/XT/AT        ..... 3
       EXIT                ..... OTHER

3<RET>           ← Specify the machine with a number.

HOST MACHINE --> IBM PC/XT/AT

DEBUGGER SYSTEM   ---> PC4000E
MONITOR           ---> V.1.3E
MCU               ---> M37450S4(MICROPROCESSOR MODE)
     bank0      0XXX  Internal     bank1 2000  Internal
     bank2      4000  Internal     bank3 6000  Internal
     bank4      8000  Internal     bank5 A000  Internal
     bank6      C000  Internal     bank7 E000  Internal
]
```

2.1.6 Abnormal Startup Screen

The following message is displayed if RTT74 cannot start normally due to communication error between the PC and PC4000E. In this case, terminate RTT74 with ^C (Ctrl+C) and check the PC4000E power, serial cable connections, and PC and PC4000E baud rates.



```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> IBM PC/XT/AT

CONNECTING DEBUGGER         ← Halt operation
^C                          ← Enter ^C
A>
```

2.2 Ending the Program

Enter "QUIT<RET>" at the command line to stop RTT74. The following termination screen is displayed.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> IBM PC/XT/AT

DEBUGGER SYSTEM  ---> PC4000E
MONITOR          ---> V.1.3E
MCU              ---> M37450S4(MICROPROCESSOR MODE)
     bank0      0XXX  Internal     bank1 2000  Internal
     bank2      4000  Internal     bank3 6000  Internal
     bank4      8000  Internal     bank5 A000  Internal
     bank6      C000  Internal     bank7 E000  Internal

]
        :
        :
        :

]QUIT<RET>

A>
```

2.3 Special Keys

The following keys have special functions during RTT74 execution:

1. ^ S (control S)

   Pauses the screen. The program resumes when any key is pressed.

2. ^C (control C)

   Forces termination of RTT74 and returns to MS-DOS prompt.

3. ^P (control P)

   Outputs the screen to a printer. Output to printer is cancelled when ^P is pressed once more.

2.4 Command Overview

RTT74 provides 25 commands shown in Table 2.2. The command character is usually the mnemonic of the operation to be performed.

Refer to Appendix B for the scope of commands depending on the MCU.

Table 2.2 Commands

Commands	Function
A	Assembles line-by-line
D	Displays memory contents in hexadecimal and ASCII data
F	Writes specified word data in specified memory area
G	Executes a program. Breakpoint can be set.
GL	Controls program execution and break with source file line number.
I	Loads the contents of a file (HEX, SYM) into RTT74.
L	Disassembles and displays contents of program memory.
M	Performs block transfer of memory contents.
O	Stores program memory contents into a file (HEX file).
P	Displays and changes break counter.
QC	Sets the realtime trace of break conditions.
QD	Displays realtime trace results (memory dump).
QL	Displays realtime trace results (disassembly).
QUIT	Ends RTT74.
S	Displays and changes memory contents.
SCOPE	Specifies scope of local identifier.
SI	Displays section information.
SHOW SOURCE	Displays source list and corresponding address.
T	Executes program one instruction at a time and displays the internal status.
U	Executes only the specified number of instructions. (Displays only internal status before execution).
X	Displays and changes register contents.
Z	Resets target MCU.
?	Displays a list of RTT74 commands and their usage.
!	Executes an MS-DOS command from RTT74.
;	Indicates comment line input.

2.5 Return Code to MS-DOS

RTT74 always returns 0 to MS-DOS.

2.6 Environment Variables

RTT74 uses MS-DOS environment variables as follows:

- The path for searching DAT file and HLP file can be specified. Use the environment variable "740DAT" to specify the path name. RTT74 searches for DAT and HLP files in the following sequence:

    1. Current directory

    2. Directory specified with the command parameter -I

    3. Path specified with the environment variable "740DAT"

- Environment variable setting example

```
A>SET 740DAT=A:\USR\DAT
        ↑          ↑
Environment variable    Path name
```

CHAPTER 3. COMMANDS

This chapter describes the syntax of each RTT74 command and provides execution examples.

3.1 Syntax

The symbols used in the syntax description are described below.

3.1.1 Input Format

Command name   Parameter,...

- Command name
  - Uppercase or lowercase character string.
  - No distinction is made between uppercase and lowercase characters.
- Parameter
  - A number, string, label, symbol, or expression required by the command.
  - The parameter depends on the command. If more than one parameter is specified, the parameters must be separated by a comma.

3.1.2 Symbols

- Only the first 20 characters are used to identify a symbol. More than 20 characters can be used, but the remaining characters are ignored.
- Uppercase and lowercase characters are distinguished.
- Only the symbols defined in the loaded SYM file can be used.

3.1.3 Numeric Values

- RTT74 treats all input numbers as hexadecimal numbers.
- However, the QD, QL, GL, and SHOW SOURCE commands treat input numbers as decimals.

3.1.4 Register Notation

Table 3.1 Register Notation

Register names	Register notations
Accumulator A	A
Index register X	X
Index register Y	Y
Stack pointers S	S
Program counter PC	P
Processor status register PS	F

3.1.5 Flag Notation

Table 3.2 Flag Notation

Flag names	Flag notations
Negative flag	N
Overflow flag	V
Index X register mode flag	T
Break flag	B
Decimal mode flag	D
Interrupt disabled flag	I
Zero flag	Z
Carry flag	C

3.1.6 Flag Display

The flag status display for the T, U, and X command shows the name of the flag if the flag is set and "-" if not.

(display example)

```
A=FF  X=FF  Y=FF  F=N----I--  S=0F6  PC=E000 INITIAL:
```

When flags N and I are set.

Figure 3.1 Processor Status Register Display Example

3.2 Commands

The RTT74 commands are summarized in the following pages. The notational conventions are described below.

[Example]

Command Name (description) Function summary

Command Syntax

The command syntax is described.

Command syntax 1
Command syntax 2

Command Execution Example

The command execution example and result are shown.

(Example)

```
]SI<RET>      ← Command input example
 #SECTION INFORMATION
 No.  NAME     OBJECT       TYPE  START  LENGTH  SOURCE       LIBRARY
 0000 RAMAREA  SAMPLE0.R74  RAM   0000   0100    SAMPLE0.A74
 0001 PROG1    SAMPLE1.R74  ROM   E000   0074    SAMPLE1.A74
 0003 PROG1    SAMPLE2.R74  ROM   E074   0077    SAMPLE2.A74
 0002 PROG2    SAMPLE1.R74  ROM   F000   0072    SAMPLE1.A74
 ]
```

Description

- The command description, precautions, and usage are described.

A (assemble) Assemble line by line

Command Syntax

A [Address]

Command Execution Example

```
]A E000<RET>                                            (Ex 1)
INITIAL:         E000  SEI<RET>
ADDR:            E001  BAD     MNEMONIC<RET>            (Ex 2)
?
ADDR:            E001  <RET>                            (Ex 3)
]A ACCUMLATOR<RET>
ACCUMLATOR:      E004  INC     A<RET>
                 E005  INC     0A<RET>                  (Ex 4)
                 E007  <RET>
]A BITSYMBOL<RET>
BITSYMBOL:       E00E  SEB     0,0A<RET>
                 E010  SEB     FLAG<RET>                (Ex 5)
                 E012  <RET>
]A IMMEDIATE<RET>
IMMEDIATE:       E016  LDA     #74<RET>                 (Ex 6)
                 E018  LDA     #DATA<RET>
                 E01A  <RET>
]A ZEROPAGE<RET>
ZEROPAGE:        E01F  LDA     74<RET>                  (Ex 7)
                 E021  LDA     DATA<RET>
                 E023  <RET>
]A ABSOLUTE<RET>
ABSOLUTE:        E028  LDA     E074<RET>                (Ex 8)
                 E02B  LDA     ADDR<RET>
                 E02E  <RET>
]A RELATIVE<RET>
RELATIVE:        E031  BRA     E074<RET>                (Ex 9)
                 E033  BRA     ADDR<RET>
                 E035  <RET>
]A SPECIAL<RET>
SPECIAL:         E03A  JSR     FF74<RET>                (Ex 10)
               E03C  JSR     SPCL<RET>
               E03E  <RET>
]A INDIRECT<RET>
INDIRECT:        E043  LDA     (74,X)<RET>              (Ex 11)
                 E045  LDA     (WORK,X)<RET>
                 E047  <RET>
]
```

Description

- The A command assembles line-by-line from the specified address and writes the corresponding machine code into program memory. (Example 1)
- If there is an error in the input mnemonic or operand, a question mark "?" is displayed and the program waits for the same address to be entered. (Example 2)
- To quit the A command, press only the '<RET>' key. (Example 3)
- If the input number starts with A, Prefix it with 0 to distinguish it from the accumulator. (Example 4)
- A bit symbol can be used. (Example 5)
- The addressing mode can be specified as follows:
  - Immediate addressing (Example 6)
    Prefix the immediate data with '#' when using this addressing mode.
  - Zero page addressing (Example 7)
    Specify an address between 0 and FF when using this addressing mode.
  - Absolute addressing (Example 8)
    Specify the absolute target address as operand when using this addressing mode.
  - Relative addressing (Example 9)
    Specify the target address as operand when using this addressing mode.
  - Special page addressing (Example 10)
    Specify an address between FF00 and FFFF when using this addressing mode and prefix the address with '\'.
  - Indirect addressing (Example 11)
    Enclose the address in parentheses when using this addressing mode.
- Refer to Appendix B for the range of each address.

D (dump) Display contents of memory

Command Syntax

D [start address] [,end address]

Command Execution Example

```
]D E089,E098<RET>                                   (Ex 1)
E089 40 F0 03 20 00 E0 CA    @.. ...
E090 F0 04 E6 40 80 F9 C6 40 C8    ...@...@.
]D E080<RET>                                        (Ex 2)
E080 40 80 F9 C6 40 C8 D0 FB A5 40 F0 03 20 00 E0 CA   @...@....@.. ...
E090 F0 04 E6 40 80 F9 C6 40 C8 D0 FB A5 40 F0 03 20   ...@...@....@..
E0A0 00 E0 CA F0 04 E6 40 80 F9 C6 40 C8 D0 FB A5 40   ......@...@....@
E0B0 F0 03 20 00 E0 CA F0 04 E6 40 80 F9 C6 40 C8 D0   .. ......@...@..
E0C0 FB A5 40 F0 03 20 00 E0 CA F0 04 E6 40 80 F9 C6   ..@.. ......@...
E0D0 40 C8 D0 FB A5 40 F0 03 20 00 E0 CA F0 04 E6 40   @....@.. ......@
E0E0 80 F9 C6 40 C8 D0 FB A5 40 F0 03 20 00 E0 CA F0   ...@....@.. ....
E0F0 04 E6 40 80 F9 C6 40 C8 D0 FB A5 40 F0 03 20 00   ..@...@....@.. .
]
```

Description

- The D command displays the contents of specified memory area as hexadecimal and ASCII data. (Example 1)
- The display is cancelled when any key is pressed.
- If the start address is omitted, display starts from the address specified in the DAT file.
- If the end address is omitted, 128 bytes are displayed (8 lines). (Example 2)
- Refer to Appendix B for the range of addresses that can be specified.
- Refer to Appendix B for information on how to display the M50734 data memory.

F (fill)

Fill data in memory block

Command Syntax

F start address, end address, data

Command Execution Example

```
]F 20,5F,74<RET>
set 74 to 005F
]D 20<RET>
0020 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74   tttttttttttttttt
0030 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74   tttttttttttttttt
0040 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74   tttttttttttttttt
0050 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74   tttttttttttttttt
0060 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
0070 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
0080 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
0090 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
]F RAMTOP,RAMBOTTOM,DATA<RET>
set 4D to 003F
]D RAMTOP,RAMBOTTOM<RET>
0000 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D   MMMMMMMMMMMMMMMM
0010 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D   MMMMMMMMMMMMMMMM
0020 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D   MMMMMMMMMMMMMMMM
0030 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D 4D   MMMMMMMMMMMMMMMM
]
```

Description

- The F command writes data in a specified range of addresses.
- The address range, must be specified as low-order address, high-order address.
- The data and destination address are displayed while data is being written.
- The operation is cancelled if any key is pressed while writing. In this case, the last address written is displayed.
- Refer to Appendix B for the range of addresses that can be specified.

G (go) Execute

Command Syntax

G [start address][,break address]

```
G [start address][,break address]
G [start address],T
G [start address],*break address
G [start address],+break address
```

Command Execution Example

```
]G<RET>                                             (Ex 1)
BREAK AT EB08                                       (Ex 2)
]G E000,EB74<RET>                                   (Ex 3)
OFFSET 0 , BREAK ADDRESS EB74 , PASS COUNT 01
]G START,<RET>                                      (Ex 4)
OFFSET 0 , BREAK ADDRESS EB74 , PASS COUNT 01
]G E000,T<RET>                                      (Ex 5)
OFFSET 0 , TRIGGER BREAK E--0
]G START,*ANDPOINT<RET>                             (Ex 6)
OFFSET 0 , AND BREAK E--0 EB01 :ANDPOINT , PASS COUNT 01
]G START,+ORPOINT<RET>                              (Ex 7)
OFFSET 0 , OR BREAK E--0  EB87 :ORPOINT , PASS COUNT 01
]
```

Description

- The G command executes the program from a specified address.
- If the start address is omitted, execution starts from the current MCU address. (Example 1)
- Execution is cancelled and the last address is displayed if any key is pressed while the MCU is executing. Then the prompt is displayed. (Example 2)
- The program operation is as follows when a break address is specified:
  1. When a break address is specified and the break condition is satisfied (the address has been executed for the specified number of times), program execution stops and the program returns to the prompt display. (Example 3)
  2. Once a break address is specified, it can be omitted the next time. (Example 4)
- The program operation is as follows when an external trigger is specified.
  1. When an external trigger is specified and the break condition is specified (external trigger is detected), realtime trace is taken and the program returns to the prompt display. (Example 5)
  2. Prefix the break address with '*' in order to take the AND of address break and external trigger. (Example 6)
  3. Prefix the break address with '+' in order to take the OR of address break and external trigger. (Example 7)

- The program counter changes as follows after a break.

Board without RTT function		Breakpoint
Board with RTT function		
Trace Mode	Before	Address after executing an instruction from the breakpoint.
	About	Address after executing 127 machine cycles from the breakpoint.
	After	Address after executing 255 machine cycles from the breakpoint.

GL (go by line number) — Execution by specifying line number

Command Syntax

GL [start line number[:start file name]][,break line number[:break file name]]
GL [start line number[:start file name]],T
GL [start line number[:start file name]],* break line number[:break file name]
GL [start line number[:start file name]],+ break line number[:break file name]

Command Execution Example

```
]GL 6,14<RET>                                        (Ex 1)
START 6:SAMPLE1.A74  BREAK 14:SAMPLE1.A74
OFFSET 0 , BREAK ADDRESS E007 , PASS COUNT 01
]GL 7:SAMPLE1.A74,15:SAMPLE1.A74<RET>                (Ex 2)
START 7:SAMPLE1.A74  BREAK 15:SAMPLE1.A74
OFFSET 0 , BREAK ADDRESS E008 , PASS COUNT 01
]GL ,16:SAMPLE2.A74<RET>                             (Ex 3)
BREAK 16:SAMPLE2.A74
OFFSET 0 , BREAK ADDRESS E01A , PASS COUNT 01
]GL 6,14<RET>                                        (Ex 4)
START 6:SAMPLE1.A74  BREAK 14:SAMPLE2.A74
OFFSET 0 , BREAK ADDRESS E018 , PASS COUNT 01
]GL 9:SAMPLE1.A74,<RET>                              (Ex 5)
START 9:SAMPLE1.A74  BREAK 14:SAMPLE2.A74
OFFSET 0 , BREAK ADDRESS E018 , PASS COUNT 01
]GL 7:SAMPLE1.A74,T<RET>
START 7:SAMPLE1.A74
OFFSET 0 , TRIGGER BREAK E--0
]GL 7:SAMPLE1.A74 , *15:SAMPLE1.A74<RET>
START 7:SAMPLE1.A74  BREAK 15:SAMPLE1.A74
OFFSET 0 , AND BREAK E--0 E008 , PASS COUNT 01
]GL 7:SAMPLE1.A74 , +26:SAMPLE2.A74<RET>
START 7:SAMPLE1.A74  BREAK 26:SAMPLE2.A74
OFFSET 0 , OR BREAK E--0  E0A2 , PASS COUNT 01
]
```

Description

- The GL command specifies the start address and breakpoint using the source file line number and executes the program. (Example 2)
- If the start line number is not specified, execution starts from the current program counter. (Example 3)
- The line numbers are specified as decimal numbers.

- If the file name is omitted and no GL command has been previously executed, the file corresponding to the current program counter is executed. (Example 1)

- If the file name is omitted and a GL command has been previously executed, the previously specified file is executed. (Example 4)

- If the break line number is omitted, the line number specified with the previous GL command is used. (Example 5)

- For the description of the following items, refer to the respective item for the G command:

  - Key input during MCU execution
  - Break conditions
  - External trigger break conditions

- The SYM file containing the necessary source line information must be loaded in order to use this command.

I (input) Loads SYM, HEX file

Command Syntax

I file name [, file name]

Command Execution Example

```
]I SAMPLE1<RET>                                    (Ex 1)
SAMPLE1.HEX SAMPLE1.SYM LOAD END --> 0 SYMBOLS DEFINED
]I SAMPLE2<RET>                                    (Ex 2)
GLOBAL SYMBOL LOADED
SAMPLE2.HEX SAMPLE2.SYM LOAD END --> 7 SYMBOLS DEFINED
]I SAMPLE3<RET>                                    (Ex 3)
LOCAL  SYMBOL LOADED
SAMPLE3.HEX SAMPLE3.SYM LOAD END --> 4 SYMBOLS DEFINED
]I SAMPLE4<RET>                                    (Ex 4)
SOURCE LINE DEBUG INFORMATION LOADED
SAMPLE4.HEX SAMPLE4.SYM LOAD END --> 0 SYMBOLS DEFINED
]I SAMPLE5<RET>
GLOBAL SYMBOL LOADED
LOCAL  SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
SAMPLE5.HEX SAMPLE5.SYM LOAD END --> 2 SYMBOLS DEFINED
]I SAMPLE.HEX,TEST.SYM<RET>                        (Ex 5)
SAMPLE.HEX TEST.SYM LOAD END --> 0 SYMBOLS DEFINED
]I SAMPLE.BAD<RET>                                 (Ex 6)

?
]
```

Description

- The I command loads the HEX and SYM files into the debugger.
- If the file extension is omitted, the HEX file and SYM file with the same file name is loaded simultaneously.
- Symbolic debugging and source line debugging are enabled when the SYM file is loaded.
- If the loaded SYM file contains global symbol information, the message "GLOBAL SYMBOL LOADED" is displayed. (Example 2)
- If the loaded SYM file contains local symbol information, the message "LOCAL SYMBOL LOADED" is displayed. (Example 3)
- If the loaded SYM file contains source line information, the message "SOURCE LINE DEBUG INFORMATION LOADED" is displayed. (Example 4)

- If the HEX file and SYM file have different names, separate the two names with a comma. In this case, the file extension must be specified for both files. The files can be specified in any order. (Example 5)

- If a file name with extension other than .HEX or .SYM is specified, '?' is displayed. (Example 6)

- Refer to Appendix E for the maximum number of symbols that can be defined.

L (list) Disassemble memory contents line by line

Command Syntax

L [start address] [,end address]

Command Execution Example

```
]L E074,E083<RET>
E074                  BBC    1,A,E087
E076                  BBC    2,A,E087
E078                  LDA    E456 :TABLE,X      (Ex 1)
E07B                  STA    88 :WORK
E07D                  LDA    88 :WORK
E07F                  CMP    #4A
E081                  BCS    E085
E083                  INC    88 :WORK
]L ,E094<RET>                                   (Ex 2)
E085                  BRA    E096 :SUB1
E087                  LDA    E456 :TABLE,Y
E08A                  STA    88 :WORK
E08C                  LDA    88 :WORK
E08E                  CMP    #4A
E090                  BEQ    E096 :SUB1
E092                  BCC    E096 :SUB1
E094                  DEC    88 :WORK
]L SUB1<RET>                                    (Ex 3)
E096 SUB1:            INC    A
E097                  LDA    #74
E099                  LDX    88 :WORK,Y
E09B                  BIT    E456 :TABLE
E09E                  LDA    (88 :WORK,X)
E0A0                  BNE    E054 :SUB2
E0A2                  CLB    0,A
E0A3                  SEB    1,88 :WORK
E0A5                  BBC    0,4A :BITSYM,E054 :SUB2
E0A8                  JSR    FF78 :SPCL
E0AA                  ??=    74                 (Ex 4)
]
```

Description

- The L command lists the memory contents.
- This list is a disassembled list of the memory contents.

- If an address corresponds to address, immediate value, or bit symbol defined in the SYM file, a colon is displayed followed by the symbol. (Example 1)
- If the start address is omitted, display starts from the last address of the previous L command. (Example 2)
- If the end address is omitted, 11 steps are displayed. (Example 3)
- If there is no machine language corresponding to a memory address, "??=" is displayed for that address. (Example 4)
- If any key is pressed while the list is being displayed, execution is cancelled and the input prompt is displayed.
- Refer to Appendix B for the range of addresses that can be listed with the L command.

M (move) Transfer data in memory block units

Command Syntax

M start address 1,end address,start address 2

Command Execution Example

```
]D 0,3F<RET>
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0020 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
0030 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
]M 0,F,28<RET>
set to 0037
]D 0,3F<RET>
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0020 EA EA EA EA EA EA EA EA 00 00 00 00 00 00 00 00   ................
0030 00 00 00 00 00 00 00 00 EA EA EA EA EA EA EA EA   ................
]
```

Description

- The M command transfers the data at the specified range of addresses to a different address in block units.
- The entire block is first read into the host machine. Therefore, overlapping addresses can be specified as the source and destination addresses.
- If the specified area cannot be read or written, execution is cancelled and '?' is displayed. Then the input prompt is displayed.
- During transfer, the data is first read while displaying the source address and then written while displaying the destination address.
- If any key is pressed while data is being read, execution is cancelled and the input prompt is displayed.
- If any key is pressed while data is being written, execution is cancelled and the last address written is displayed. Then the input prompt is displayed.
- Refer to Appendix B for the range of addresses that can be specified.

O (output) Outputs various files

Command Syntax

O file name.HEX[,destination area number]

Command Execution Example

```
]O TEST_E.HEX,E<RET>                          (Ex 1)
]O TEST_F.HEX<RET>
AREA NUMBER ? F<RET>                          (Ex 2)
]
```

Description

- The O command outputs the contents of program memory as Intel hexadecimal format HEX file.

- The block to be output must be specified following the file name separated by a comma. (Example 1)

- If the block to be output is omitted, you are prompted for the block number. Enter the block number to be output at the prompt. Refer to Appendix B for the appropriate number or character. (Example 2)

P (pass count) — Sets pass count

Command Syntax

P[count]

Command Execution Example

```
]P 2<RET>                                               (Ex 1)
]P<RET>
02 FF<RET>                                              (Ex 2)
]P<RET>
FF <RET>                                                (Ex 3)
]
```

Description

- The P command specifies the number of passes to be executed before breaking at the break address. (Example 1)
- An error occurs if the value is more than 1 byte or is 0.
- If "P<RET>" is entered, the current count is displayed and then you are prompted for the new count. (Example 2)

  If only "<RET>" is entered at this point, the count remains unchanged and the input prompt is displayed. (Example 3)

QC (condition) — Set realtime trace mode

Command Syntax

QC

Command Execution Example

```
]QC<RET>                                                    (Ex 1)
OFFSET MODE ? ( 0 = BEFORE , 1 = ABOUT , 2 = AFTER , 9 = END )
0 2<RET>
TRIGGER TYPE ? ( 0 = EDGE , 1 = LEVEL , 9 = END )
0 1<RET>
SIGNAL LOGIC ? ( 0 = LOW , 1 = HIGH , 9 = END )
0 1<RET>
]QC<RET>
OFFSET MODE ? ( 0 = BEFORE , 1 = ABOUT , 2 = AFTER , 9 = END )
2 <RET>                                                     (Ex 2)
TRIGGER TYPE ? ( 0 = EDGE , 1 = LEVEL , 9 = END )
1 9<RET>                                                    (Ex 3)
]
```

Description

- The QC command sets the realtime trace offset mode or trigger break condition. The condition is set by entering a number corresponding to a menu item. (Example 1)
- Offset mode must specify whether to trace BEFORE, ABOUT, or AFTER a breakpoint.
- Trigger type and signal logic conditions are also set by entering a number.
- Enter only "<RET>" if nothing is to be changed. (Example 2)
- Enter "9" to end. (Example 3)

QD (quest-data)

Display realtime trace data

Command Syntax

- Relative specification
  QD [display cycle]

- Absolute specification
  QD [start cycle],[end cycle]

Command Execution Example

```
]G RESET,T1MODULE<RET>
OFFSET 1 , BREAK ADDRESS E039 :T1MODULE , PASS COUNT 01_
]QD<RET>                                          (Ex 1)
    ------------------ MELPS 740 -----------------     --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
      -5                   007F      E0     0    0      1 1 1 1 1 1
      -4                   007E      2C     0    0      1 1 1 1 1 1
      -3                   007D      C8     0    0      1 1 1 1 1 1
      -2                   FFF2      39     0    1      1 1 1 1 1 1
      -1                   FFF3      E0     0    1      1 1 1 1 1 1
ABO    0 T1MODULE:         E039      E6     1    1      1 1 1 1 1 1
       1                   E03A      40     0    1      1 1 1 1 1 1
       2 LCOUNT:           0040      39     0    1      1 1 1 1 1 1
       3 LCOUNT:           0040      39     0    1      1 1 1 1 1 1
       4 LCOUNT:           0040      3A     0    0      1 1 1 1 1 1

]QD -5<RET>                                       (Ex 2)
    ------------------ MELPS 740 -----------------     --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
     -10                   E01C      0F     0    1      1 1 1 1 1 1
      -9                   E01D      F3     0    1      1 1 1 1 1 1
      -8                   E02C      BD     0    1      1 1 1 1 1 1
      -7                   E02C      BD    (1)   1      1 1 1 1 1 1
      -6                   E02C      BD     0    1      1 1 1 1 1 1

]QD 5<RET>                                        (Ex 3)
    ------------------ MELPS 740 -----------------     --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
      -5                   007F      E0     0    0      1 1 1 1 1 1
      -4                   007E      2C     0    0      1 1 1 1 1 1
      -3                   007D      C8     0    0      1 1 1 1 1 1
      -2                   FFF2      39     0    1      1 1 1 1 1 1
      -1                   FFF3      E0     0    1      1 1 1 1 1 1
```

```
]QD -130<RET>
    ------------------ MELPS 740 -----------------    --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
    -128 T1MODULE:         E039      E6    1     1      1 1 1 1 1 1
    -127                   E03A      40    0     1      1 1 1 1 1 1
                            :
                            :
                            :
      -1                   FFF3      E0    0     1      1 1 1 1 1 1
ABO    0 T1MODULE:         E039      E6    1     1      1 1 1 1 1 1
       1                   E03A      40    0     1      1 1 1 1 1 1

]QD -10<RET>                                     (Ex 4)
    ------------------ MELPS 740 -----------------    --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
    -128 T1MODULE:         E039      E6    1     1      1 1 1 1 1 1
    -127                   E03A      40    0     1      1 1 1 1 1 1
    -126 LCOUNT:           0040      27    0     1      1 1 1 1 1 1
    -125 LCOUNT:           0040      27    0     1      1 1 1 1 1 1
    -124 LCOUNT:           0040      28    0     0      1 1 1 1 1 1
    -123                   E03B      A5    1     1      1 1 1 1 1 1
    -122                   E03C      40    0     1      1 1 1 1 1 1
    -121 LCOUNT:           0040      28    0     1      1 1 1 1 1 1
    -120                   E03D      D0    1     1      1 1 1 1 1 1
    -119                   E03E      02    0     1      1 1 1 1 1 1

]QD 236<RET>
    ------------------ MELPS 740 -----------------    --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
    -118                   E03F      E6    0     1      1 1 1 1 1 1
    -117 T1EXIT:           E041      40    0     1      1 1 1 1 1 1
                            :
                            :
                            :
     126                   FFF2      39    0     1      1 1 1 1 1 1
     127                   FFF3      E0    0     1      1 1 1 1 1 1

]QD 10<RET>                                      (Ex 5)
    ------------------ MELPS 740 -----------------    --- EXT ---
    TCNT SYMBOL          ADDRESS    DATA  SYNC  R/W     5 4 3 2 1 0 bit
```

```
]QD 0<RET>                                             (Ex 6)
? PARAMETER ZERO !!
]QD -5,5<RET>                                          (Ex 7)
    ------------------ MELPS 740 -----------------      --- EXT ---
    TCNT SYMBOL          ADDRESS   DATA  SYNC  R/W      5 4 3 2 1 0 bit
      -5                   007F     E0    0     0       1 1 1 1 1 1
      -4                   007E     2C    0     0       1 1 1 1 1 1
      -3                   007D     C8    0     0       1 1 1 1 1 1
      -2                   FFF2     39    0     1       1 1 1 1 1 1
      -1                   FFF3     E0    0     1       1 1 1 1 1 1
ABO    0 T1MODULE:         E039     E6    1     1       1 1 1 1 1 1
       1                   E03A     40    0     1       1 1 1 1 1 1
       2 LCOUNT:           0040     39    0     1       1 1 1 1 1 1
       3 LCOUNT:           0040     39    0     1       1 1 1 1 1 1
       4 LCOUNT:           0040     3A    0     0       1 1 1 1 1 1
       5                   E03B     A5    1     1       1 1 1 1 1 1

]QD 1, -1<RET>                                         (Ex 8)
? START LINE < END LINE
]QD -300,0<RET>                                        (Ex 9)
? INVALID AREA
]QD -129,0<RET>                                        (Ex 10)
? OFFSET MISS MATCH
]QD 0,128<RET>                                         (Ex 11)
? OFFSET MISS MATCH
]
```

Description

- The QD command displays the realtime trace data together with the status of the external signal (6 or 4 bits).

- Realtime trace must be started prior to this command with a G command.

- The following information is displayed:

  1. TCNT indicates the relative number of cycles from the breakpoint.

  2. SYMBOL indicates the symbol corresponding to the address.

  3. ADDRESS indicates the status of the address bus.

  4. DATA indicates the status of the data bus.

  5. SYNC indicates the signal output during instruction operand code fetch. '1' indicates that operand is being fetched. A SYNC value enclosed in parentheses indicates a dummy SYNC[1] and the instruction in this line is not executed.

  6. R/W is a signal indicating the direction of the data bus. '1' indicates READ and '0' indicates WRITE.

[1] A dummy SYNC issued before interrupt processing and not actually executed.

7. EXT indicates an external signal.

- Trace area specification

  - Cycle number is treated as a decimal number.

  1. Relative specification (with 1 parameter)

  - The trace about the breakpoint is displayed upon first execution after a trace. (Example 1)

  - Specify the number of counts to be displayed following the previous QD or QL command display for the second and subsequent execution.

    Specify a number preceded by a minus sign to backtrack from the previous display. (Example 2)

    Specify an unsigned number if the display is to be forward from the previous display. (Example 3)

  - Backtracking further after reaching the top of the trace memory area results in the same area being displayed. (Example 4)

  - If the bottom of the trace memory area is reached, input prompt is displayed even if the specified number of lines is not displayed. If further area is specified, only the title line appears. (Example 5)

  - An error occurs and the message "? PARAMETER ZERO !!" is displayed if 0 is specified as count. Then the input prompt is displayed. (Example 6)

  2. Absolute specification (with 2 parameters)

  - In this case, specify the count with breakpoint as 0. (Example 7)

  - If the start cycle is greater than the end cycle, the message "? START LINE < END LINE" is displayed and then the input prompt is displayed. (Example 8)

  - If the value is not within -255 to 255, the message "? INVALID AREA" is displayed and then the input prompt is displayed. (Example 9)

  - The range of value that can be specified depends on the offset (specified with the QC command) as shown in Table 3.3.

Table 3.3 Range of Absolute Value for the QD and QL Command

Offset	Range
0 (BEFORE)	-255 to 0
1 (ABOUT)	-128 to 127
2 (AFTER)	0 to 255

If this range is exceeded, the message "? OFFSET MISS MATCH" is displayed and then the input prompt is displayed. (Examples 10, 11)

- Note that the previous trace result or random data is displayed if the number of trace lines is less than 256 cycles.

QL (quest-list)

List realtime trace data

Command Syntax

- Relative specification
  QL [display cycle]

- Absolute specification
  QL [start cycle],[end cycle]

Command Execution Example

```
]G RESET,T1EXIT<RET>
OFFSET 1 , BREAK ADDRESS E041 :T1EXIT , PASS COUNT 01_
]QL<RET>                                        (Ex 1)
     -------------- MELPS 740 ------------
    TCNT SYMBOL        ADDRESS MNEMONIC
      -4                  E03D BNE   E041 :T1EXIT
ABO    0 T1EXIT:          E041 RTI

]QL -20<RET>                                    (Ex 2)
     -------------- MELPS 740 ------------
    TCNT SYMBOL        ADDRESS MNEMONIC
     -25                  E01B BBC   4,A,E02C
       ++++  INTERRUPT ---> VECTOR ADDRESS in $FFF2
     -12 T1MODULE:        E039 INC   40 :LCOUNT
      -7                  E03B LDA   40 :LCOUNT

]QL 20<RET>                                     (Ex 3)
     -------------- MELPS 740 ------------
    TCNT SYMBOL        ADDRESS MNEMONIC
      -4                  E03D BNE   E041 :T1EXIT
ABO    0 T1EXIT:          E041 RTI
       6                  E02C LDA   E65D :TABLE,X
      11                  E02F BEQ   E037
      13                  E031 INX

]QL -130<RET>
     -------------- MELPS 740 ------------
    TCNT SYMBOL        ADDRESS MNEMONIC
    -126 RESET:           E000 SEI
    -124                  E001 CLT
                         :

                         :
      -4                  E03D BNE   E041 :T1EXIT
ABO    0 T1EXIT:          E041 RTI
```

```
]QL -20<RET>                                              (Ex 4)
     -------------- MELPS 740 ------------
    TCNT SYMBOL         ADDRESS MNEMONIC
    -126 RESET:            E000 SEI
    -124                   E001 CLT
    -122                   E002 LDX    #7F
    -120                   E004 TXS
    -118                   E005 LDM    #40 :T1DATA,F0
    -114                   E008 LDM    #00,F1
    -110                   E00B LDM    #40 :T1DATA,F2

]QL 240<RET>
     -------------- MELPS 740 ------------
    TCNT SYMBOL         ADDRESS MNEMONIC
    -106                   E00E LDM    #00,F3
    -102                   E011 LDM    #00,ED
                        :
                        :
     105                   E035 STA    42 :WORK
       ++++  INTERRUPT ---> VECTOR ADDRESS in $FFF2
     116 T1MODULE:         E039 INC    40 :LCOUNT

]QL 20<RET>                                               (Ex 5)
     -------------- MELPS 740 ------------
    TCNT SYMBOL         ADDRESS MNEMONIC

]QL 0
? PARAMETER ZERO !!
]QL -10,10                                                (Ex 6)
     -------------- MELPS 740 ------------
    TCNT SYMBOL         ADDRESS MNEMONIC
      -7                   E03B LDA    40 :LCOUNT
      -4                   E03D BNE    E041 :T1EXIT
ABO    0 T1EXIT:           E041 RTI
       6                   E02C LDA    E65D :TABLE,X
]QL 1, -1<RET>
? START LINE < END LINE
]QL -300,0<RET>
? INVALID AREA
]QL -129,0<RET>
? OFFSET MISS MATCH
]QL 0,128<RET>
? OFFSET MISS MATCH
]
```

Description

- The QL command lists the contents of the realtime trace memory.
- Realtime trace must be started with a G command prior to this command.
- The following information is displayed:

  1. TCNT indicates the relative number of cycles from the breakpoint.
  2. SYMBOL indicates the symbol corresponding to the address.
  3. ADDRESS indicates the status of the address bus.
  4. MNEMONIC shows the instruction mnemonic.
  5. If an interrupt occurs, the interrupt vector address is displayed following the message:

     "++++ INTERRUPT ----> VECTOR ADDRESS in $"

- Trace area specification

  - Cycle number is treated as a decimal number.

  1. Relative specification (with 1 parameter)

  - The trace about the breakpoint is displayed upon first execution after a trace. (Example 1)
  - Specify the number of counts to be displayed following the previous QD or QL command display for the second and subsequent execution.

    Specify a number preceded by a minus sign to backtrack from the previous display. (Example 2)

    Specify an unsigned number if the display is to be forward from the previous display. (Example 3)

  - Backtracking further after reaching the top of the trace memory area results in the same area being displayed. (Example 4)
  - If the bottom of the trace memory area is reached, input prompt is displayed even if the specified number of lines is not displayed. If further area is specified, only the title line appears. (Example 5)
  - The error message is identical to the QD command.

  2. Absolute specification (with 2 parameters)

  - In this case, specify the count with breakpoint as 0. (Example 6)
  - The error message is identical to the QD command.

- Note that the previous trace result or random data is displayed if the number of trace lines is less than 256 cycles.

Q (QUIT) End RTT74

Command Syntax

QUIT

Command Execution Example

```
]QUIT<RET>
A>
```

Description

- Terminates execution of RTT74.

S (set) Access and set memory in bytes

Command Syntax

S address

Command Execution Example

```
]D 70,8F<RET>
0070 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
0080 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
]S 74<RET>
                  0074 EA DATA<RET>
                  0075 EA 54<RET>
                  0076 EA 54<RET>
                  0077 EA 37<RET>
                  0078 EA 34<RET>
                  0079 EA <RET>                (Ex 1)
                  007A EA .<RET>               (Ex 2)
]D 70,8F<RET>
0070 EA EA EA EA 52 54 54 37 34 EA EA EA EA EA EA EA   ....RTT74.......
0080 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA   ................
]
```

Description

- The S command displays and changes the contents of memory at the specified address.
- When this command is entered, the program displays the memory contents and waits for input.
  - Enter the new data to change the memory contents.
  - If only "<RET>" is entered, the memory is unchanged and the contents of the next address is displayed. (Example 1)
  - Enter ".<RET>" to terminate the command. (Example 2)
- Refer to Appendix B for the range of addresses that can be specified with the S command.
- Refer to Appendix B for information on how to access M50734 data memory.

SCOPE Specify scope of local symbols

Command Syntax

SCOPE [relocatable file name]
SCOPE [section number]
SCOPE

Command Execution Example

```
  ]SI<RET>
  #SECTION INFORMATION
  No.   NAME        OBJECT       TYPE  START  LENGTH  SOURCE
LIBRARY
  0000  RAMAREA     SAMPLE0.R74  RAM   0000   0100    SAMPLE0.A74
  0001  PROG1       SAMPLE1.R74  ROM   E000   0074    SAMPLE1.A74
  0004  PROG1       SAMPLE2.R74  ROM   E074   008A    SAMPLE2.A74
  0003  PROG2       SAMPLE2.R74  ROM   F000   0072    SAMPLE2.A74
  0002  VECTOR      SAMPLE1.R74  ROM   FFE0   0020    SAMPLE1.A74
  ]L E000, E002<RET>
  E000 START:           SEI
  E001                  CLT
  E002                  CLD
  ]L E074, E077<RET>
  E074 .F0:             LDA   C786 :TABLE,X
  E077                  BEQ   E08F :.F1
  ]L F000, F002<RET>
  F000 START:           CPX   #00 :WORK
  F002                  BNE   F008 :.S2
  ]XP<RET>
  PC=E000

  ]L START<RET>
  E000 START:           SEI
  E001                  CLT
                  :
                  :
  ]SCOPE SAMPLE2.R74<RET>                       (Ex 1)
  ]L START<RET>
  F000 START:           CPX   #00 :WORK
  F002                  BNE   F008 :.S2
```

```
                  :
                  :
]SCOPE 4<RET>                                          (Ex 2)
]L ??START<RET>
E074 ??START:          LDA   C786 :TABLE,X
E077                   BEQ   E08F :.F1
                  :
                  :
]SCOPE<RET>                                            (Ex 3)
]L START<RET>
E000 START:            SEI
E001                   CLT
                 :
                 :
]
```

Description

- The SCOPE command specifies the scope of local symbols.
- The scope of local symbols can be specified in three different ways.

1. Relocatable file (Example 1)

- The scope of local symbols is limited to the specified source file.

2. Section number (Example 2)

- The scope of local symbols is limited to the files containing the specified section.
- This method enables the use of section local labels in the specified section.

3. "SCOPE<RET>" (Example 3)

- When the command is entered, the scope of local symbols is limited to the file corresponding to the current program counter. While the command is executing, the scope of local symbols is limited to the file corresponding to the file corresponding to the program counter of the currently processing address.
- This is the default when RTT74 is started.

SHOW SOURCE (show-source-file) Display source list

Command Syntax

SHOW SOURCE [source file name] [,line number]

Command Execution Example

```
]SHOW SOURCE SAMPLE.A74,1
FILE SAMPLE.A74 , LINE 1
     1              .INCLUDE        SYMDEF.H
     2              .INCLUDE        SFR.H
     3              .INCLUDE        EXTMAIN.H
     4       ;
     5              .INCLUDE        M74.MAC
     6       ;
     7       ;
     8              .SECTION        MAIN
     9              .ORG            0E000H
    10              .FUNC           MAIN
    11       ;
    12 E000 :RESET:
    13 E000         I = 1
    14 E001         T = 0
    15 E002         D = 0
    16 E003         S = $7F
    17       ;
    18 E006         A = 0
    19 E008         X = 0
    20 E00A         FOR  X <= 0BFH
    21 E010                 [0, X] = A

]SHOW SOURCE                                        (Ex 1)
FILE SAMPLE.A74 , LINE 22
    22 E012                 X = ++X
    23 E013         NEXT
    24       ;
    25 E015         [Port0] = 0
    26 E018         [Port1] = 0
    27 E01B         [Port2] = 0
    28 E01E         [Port3] = 0
```

```
     29 E021          [Port4] = 0
     30 E024          [Port5] = 0
     31 E027          [Port6] = 0
     32          ;
     33 E02A          [Port0Dir] = OUTDIR
     34 E02D          [Port1Dir] = OUTDIR
     35 E030          [Port2Dir] = INDIR
     36 E033          [Port3Dir] = 00011001B
     37 E036          [Port5Dir] = 01000101B
     38 E039          [Port6Dir] = 11111001B
     39          ;
     40 E03C          [IODev] = OFF
     41 E03E          [DevEnvir] = 0
     42          ;

]SHOW SOURCE SAMPLE.A74                              (Ex 2)
FILE SAMPLE.A74 , LINE 1
      1               .INCLUDE         SYMDEF.H
      2               .INCLUDE         SFR.H
      3               .INCLUDE         EXTMAIN.H
      4          ;
      5               .INCLUDE         M74.MAC
      6          ;
      7          ;
      8               .SECTION         MAIN
      9               .ORG             0E000H
     10               .FUNC            MAIN
     11          ;
     12 E000 :RESET:
     13 E000          I = 1
     14 E001          T = 0
     15 E002          D = 0
     16 E003          S = $7F
     17          ;
     18 E006          A = 0
     19 E008          X = 0
     20 E00A          FOR  X <= 0BFH
     21 E010                   [0, X] = A

]SHOW SOURCE ,7400                                   (Ex 3)
FILE SAMPLE.A74 , LINE 7400
   7400 E041 IOout:
```

```
  7401 E041         A = [DevEnvir]
  7402 E043         IF  BIT_A1 == ON
  7403 E045                 [IODev] = ON
  7404 E049         ELSE
  7405 E049                 [IODev] = OFF
  7406 E04B         ENDIF
  7407        ;
  7408        ;
  7409 E04B         FOR  --X != 0
  7410        ;
  7411 E04E                 DO
  7412 E04E                 WHILE [TxRDY] == OFF
  7413        ;
  7414 E051                 Y = [TABLE, X]
  7415 E054                 IF  BIT_A7
  7416 E056                         [Port1] = [Upper_Pattern, Y]
  7417 E05D                 ELSE
  7418 E05D                         [Port1] = [Lower_Pattern, Y]
  7419 E062                 ENDIF
  7420        ;

]
```

Description

- The SHOW SOURCE command displays the contents of the source file together with the absolute address.

  However, the SYM file containing the source line information must be loaded in order for the absolute address to be displayed.

- 21 lines are displayed at once.

- The line numbers are in decimal.

- The file name and line number may be omitted.

  - If both the source file name and line number are omitted (Example 1), the lines of the previously displayed source file are displayed.

  - If only the line number is omitted (Example 2), display starts from line 1.

  - If only the file name is omitted (Example 3), the previously specified source file is displayed.

SI (section-information) Display section information

Command Syntax

SI

Command Execution Example

```
]SI<RET>
#SECTION INFORMATION
No.   NAME    OBJECT       TYPE  START  LENGTH  SOURCE         LIBRARY
0000  Z_RAM   SAMPLE0.R74  RAM   0000   0100    SAMPLE0.A74
0001  MAIN    SAMPLE1.R74  ROM   E000   0740    SAMPLE1.A74
0006  MAIN    SAMPLE2.R74  ROM   E740   0038    SAMPLE2.A74
0002  SUB     SAMPLE1.R74  ROM   E778   0077    SAMPLE1.A74
0005  SUB     SAMPLE2.R74  ROM   E7EF   0072    SAMPLE2.A74
0007  SUB     SAMPLE2.R74  ROM   E861   0045    SAMPLE2.A74
0003  DATA    SAMPLE1.R74  ROM   E8A6   0076    SAMPLE1.A74
0008  P       SRA74.R74    ROM   E91C   0085    SRA74.A74  MYLIB.LIB
0004  VECTOR  SAMPLE1.R74  ROM   FFF0   0010    SAMPLE1.A74
]
```

Description

- The SI command displays the section information loaded from the SYM file.
- The following section information is displayed from left to right.

No.	Section number
NAME	Section name
OBJECT	Relocatable file name
TYPE	ROM/RAM type
START	Allocation start address
LENGTH	Allocation area size
SOURCE	Source file name
LIBRARY	Library file name

T (trace) Step execution

Command Syntax

T [step count]

Command Execution Example

```
]T 3<RET>                                                   (Ex 1)
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E000 RESET:
      SEI
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E001
      CLT
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E002
      CLD
*E003
]T<RET>                                                     (Ex 2)
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E003
      LDX   #7F<SPACE>
A=00  X=7F  Y=00  F=------I--  S=0BF  PC=E005
      TXS  <SPACE>
A=00  X=7F  Y=00  F=------I--  S=07F  PC=E006
      LDA   #00<SPACE>
A=00  X=7F  Y=00  F=------IZ-  S=07F  PC=E008
      LDX   #00<RET>
*E00A
]
```

Description

- The T command executes the specified number of steps from the current program counter. (Example 1)
- Each display line shows the status just before execution and the instruction to be executed.
- When "T<RET>" is entered, one step is executed and then input is prompted. Then a step is executed each time the space key is pressed. The command terminates if any other key is pressed. (Example 2)
- The next program counter is displayed preceded by an asterisk at the end of the T command.
- An error occurs if 0 is specified for step count.

U (untrace)

Step execution

Command Syntax

U [step count]

Command Execution Example

```
]U<RET>
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E000 RESET:
      SEI
*E001
]U 4<RET>
A=00  X=00  Y=00  F=N----I--  S=0BF  PC=E001
      CLT
*E006
]
```

Description

- The U command displays the status before single step execution and then executes the specified number of steps from the current program counter. The contents of the current program counter is displayed preceded by an asterisk at the end of the command.
    - Each display line shows the contents of each register before execution and the instruction to be executed.
    - Execution is cancelled if any key is pressed during execution.
- An error occurs if 0 is specified for step count.

X (examine) Examine and change register

Command Syntax

Examine register
X[register name(A,X,Y,F,S,P)]

Change register
X{register name}value

Command Execution Example

```
]X<RET>                                                (Ex 1)
A=72  X=74  Y=77  F=N----I--  S=0BF  PC=E000 RESET:
      SEI

]XA<RET>                                               (Ex 2)
A=72 <RET>

]XX<RET>                                               (Ex 3)
X=74 1<RET>

]XY FF<RET>                                            (Ex 4)

]X<RET>
A=72  X=01  Y=FF  F=N----I--  S=0BF  PC=E000 RESET:
      SEI

]
```

Description

- The X command displays and changes the contents of MCU registers.
- A register is accessed as follows:
  - "X<RET>" (Example 1)

    All registers are displayed.
  - Specifying register name after "X" (Example 2)

    The contents of the specified register is displayed and the input prompt appears.
- The contents of a register is changed as follows:
  - Enter the new value at the input prompt after the current contents. (Example 3)
  - To change the value directly from the input prompt, specify the new value following the register name when entering the X command. (Example 4)

Z (reset) — Reset MCU hardware

Command Syntax

Z

Command Execution Example

```
]Z<RET>
]
```

Description

- The Z command resets the target MCU.

? (help) Display command list

Command Syntax

? [command name]

Command Execution Example

```
  ]?<RET>                                            (Ex. 1)
  [Command List]                                          VER 2.10.00
Command        | Function Description
   A           | Assemble line by line.
   BP          | Set and display breakpoints. (PC4600)
   ^C          | Force termination of RTT74.
   CV          | Perform coverage analysis control. (PC4600)
   D           | Display memory contents.
   F           | Write specified data in specified memory area.
   G           | Execute program with breakpoint. (PC4000E)
   G           | Execute program without breakpoint. (PC4600)
   GB          | Execute program with break condition specified with BP
               | command. (PC4600)
   GL          | Execute program with address specified line number.
               | (PC4000E)
   I           | Include file (HEX or SYM file) into RTT74.
   L           | Display disassembled result of program memory.
   M           | Block transfer contents of memory.
   MAP         | Set and display debug memory space map information.
   MCU         | Set unique data of target MCU.
   QC          | Set realtime trace break condition.
   QD          | Display (dump format) realtime trace results.
   QL          | Display (list format) realtime trace results.
  ==next page: <SPACE>key, next line :<RET>key, end : other key==<SPACE>
   QP          | Set and display trace points.
   QUIT        | Stop RTT74 execution.
   SCOPE       | Specify scope of local symbols.
   SI          | Display section information of program being debugged.
   S           | Display and change contents of memory.
   SHOW SOURCE | Display SRA source list and corresponding address.
   STOP        | Stop execution of target MCU.(PC4600)
   T           | Execute program step by step and display contents of
               | registers.
   U           | Execute the program for the specified number of
               | repetitions after displaying the contents of registers.
   X           | Display and change contents of registers.
   Z           | Hardware reset the target MCU.
   ?           | Display RTT74 command list and usage.
  !             | Execute MS-DOS command from RTT74.
  ;             | Indicates a comment line.
]?Z<RET>                                          (Ex 2)
Z(reset)                                          MCU hardware reset

Command format
  Z

Command execution example
>Z<RET>
>

Content
(1) Hardware resets MCU.
]
```

Description

- The ? command displays how to use each command.
- When "?<RET>" is entered, the first page is displayed followed by the input prompt. The next page is displayed when any key is pressed. (Example 1)
  - Pressing the space key scrolls one screen.
  - Entering "<RET>" scrolls one line.
  - The command is cancelled if any other key is pressed.
- If a command is specified following "?", the description of the specified command is displayed. (Example 2)

! (shell-escape) Escape to shell

Command Syntax

! [MS-DOS command name]

Command Execution Example

```
]!<RET>                                            (Ex 1)
Now MS-DOS system!
If you return to RTT74, enter EXIT<RET>.
Command  Version 3.10
A>SRA74 FILE -L<RET>
       :
       :
A>EXIT<RET>                                        (Ex 2)
]!MORE FILE.PRN<RET>                               (Ex 3)
       :
       :
]
```

Description

- The ! command executes an MS-DOS command (including external commands).
- The command can be entered in one of the following ways.

  1. !<RET> (Example 1)

     An MS-DOS prompt is displayed and MS-DOS commands can be entered. Enter "EXIT<RET>" to return to RTT74 command prompt. (Example 2)

  2. !xxx<RET> (xxx: MS-DOS command) (Example 3)

     The specified MS-DOS command is executed. The RTT74 input prompt is displayed after execution.

Caution

The PATH must be set correctly in order to use the ! command. In addition, the name of the command file must be set to the environment variable "COMSPEC" in order to specify the load directory when the non-resident section of COMMAND.COM is destroyed.

; (comment-line)   Comment line

Command Syntax

; [character string]

Command Execution Example

```
]; this line is a comment.
]
```

Description

- Any character string can be entered as comment. This line does not affect the execution of RTT74.

CHAPTER 4. USING SYMBOLIC FILES

4.1 Symbolic Debugging Function

RTT74 enables the use of labels and symbols defined in the source program for address specification by loading the SYM file. The labels and symbols are also displayed during disassembly and tracing to simplify debugging. This is referred to as symbolic debugging.

4.1.1 Symbol and Label Definition

RTT74 classifies symbols and labels into the following seven types in order to facilitate debugging in module units.

1. Symbols, labels, and their scope

   (a) Global label
   (b) Global symbol
   (c) Global bit symbol
   (d) Local label
   (e) Local symbol
   (f) Local bit symbol
   (g) Section local label

- Global labels, global symbols, and global bit symbols are valid in the entire work area.
- Local labels, local symbols, and local bit symbols are valid in the work area corresponding to the file in which they are defined.
- Section local labels are valid in the section in which the scope is defined with the SCOPE command.

2. Symbol and label search sequence

RTT74 searches for symbols and labels in the following sequence.

(a) Address search sequence

1. Section local label
2. Local label
3. Local symbol
4. Global label
5. Global symbol

(b) Immediate value search sequence

1. Local symbol
2. Local label
3. Global symbol
4. Global label

(c) Bit symbol search sequence

1. Global bit symbol
2. Local bit symbol

4.1.2 Operation Examples

The following examples show how symbols and labels are managed.

- Sample program

- Source file 1 (WORK1.A74)

```
        .SECTION        Z
        .ORG            0
:ZADDRESS:
   .BLKB           1
;
:GLOBALBIT        .EQU  0,0
;
        .END
```

- Source file 2 (ID1.A74)

```
        .ZEXT   ZADDRESS
LOCAL1DATA      .EQU        0
LOCAL1BIT       .EQU        0,0
;
        .SECTION        PROG1
        .ORG            0E000H
:RESET:
        SEI
        CLT
        CLD
        S = 07FH
:GLOBALSTART:
LOCAL1START:
??SECTION1START:
        AND     #LOCAL1DATA
        AND     ZADDRESS
        CLB     LOCAL1BIT
        JMP     0F000H
:FIN1:
        NOP
        .END
```

- Source file 3 (ID2.A74)

```
        .EXT    GLOBALSTART,RESET
        .ZEXT   ZADDRESS
LOCAL2DATA      .EQU    0
LOCAL2BIT       .EQU    0,0
;
        .SECTION        PROG2
        .ORG            0F000H
LOCAL2START:
??SECTION2START:
        NOP
        ORA     #LOCAL2DATA
        ORA     ZADDRESS
        SEB     LOCAL2BIT
        JMP     GLOBALSTART
:FIN2:
        NOP
;
        .SECTION        VECTOR
        .ORG            0FFFEH
        .WORD           RESET
        .END
```

- Assemble, link example

```
A>SRA74 WORK<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1  ( WORK.A74 )
now processing pass 2  ( WORK.A74 )

ERROR   COUNT            00000
WARNING COUNT            00000
STRUCTURED STATEMENT     00000 LINES
TOTAL   LINE ( SOURCE )  00008 LINES
TOTAL   LINE ( OBJECT )  00008 LINES
COMMENT LINE ( SOURCE )  00002 LINES
COMMENT LINE ( OBJECT )  00002 LINES
OBJECT  SIZE ( Z      )  00001 (0001) BYTES

A>SRA74 ID1 -S<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1  ( ID1.A74 )
now processing pass 2  ( ID1.A74 )

ERROR   COUNT            00000
WARNING COUNT            00000
STRUCTURED STATEMENT     00001 LINES
TOTAL   LINE ( SOURCE )  00021 LINES
TOTAL   LINE ( OBJECT )  00021 LINES
COMMENT LINE ( SOURCE )  00001 LINES
COMMENT LINE ( OBJECT )  00001 LINES
OBJECT  SIZE ( PROG1  )  00016 (0010) BYTES



A>SRA74 ID2.A74 -S<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
```

```
now processing pass 1   ( ID2.A74 )
now processing pass 2   ( ID2.A74 )

ERROR   COUNT               00000
WARNING COUNT               00000
STRUCTURED STATEMENT        00000 LINES
TOTAL   LINE  ( SOURCE )    00021 LINES
TOTAL   LINE  ( OBJECT )    00021 LINES
COMMENT LINE  ( SOURCE )    00002 LINES
COMMENT LINE  ( OBJECT )    00002 LINES
OBJECT  SIZE  ( PROG2  )    00011 (000B) BYTES
OBJECT  SIZE  ( VECTOR )    00002 (0002) BYTES

A>LINK74 WORK ID1 ID2,,,-S -FID<RET>
MELPS 740 LINKER V.1.01.01C
Copyright 1989,1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1
processing "WORK.R74"
processing "ID1.R74"
processing "ID2.R74"
now processing pass 2
processing "WORK.R74"
processing "ID1.R74"
processing "ID2.R74"
TOTAL ROM SIZE 29 (1DH) BYTES
TOTAL RAM SIZE 1 (1H) BYTES
```

- Section information

The following is the section information displayed with the SI command after loading the SYM file and HEX file.

```
]I ID<RET>
GLOBAL SYMBOL LOADED
LOCAL  SYMBOL LOADED
ID.HEX ID.SYM LOAD END --> 14 SYMBOLS DEFINED
]SI<RET>
#SECTION INFORMATION
No.   NAME         OBJECT       TYPE  START  LENGTH  SOURCE
0000  Z            WORK.R74     RAM   0000   0001    WORK.A74
0001  PROG1        ID1.R74      ROM   E000   0010    ID1.A74
0002  PROG2        ID2.R74      ROM   F000   000B    ID2.A74
0003  VECTOR       ID2.R74      ROM   FFFE   0002    ID2.A74
]
```

- Major labels and symbols

Table 4.1 shows the major labels and symbols used in examples in the next section.

Table 4.1 Major Labels and Symbols

Value	Name	Type	File
E006	GLOBALSTART	Global label	
	LOCAL1START	Local label	ID1.A74
	??SECTION1START	Section local label	PROG1
F000	LOCAL2START	Local label	ID2.A74
	??SECTION2START	Section local label	PROG2
0	ZADDRESS	Global label	
	GLOBALDATA	Global symbol	
	LOCAL1DATA	Local symbol	ID1.A74
	LOCAL2DATA	Local symbol	ID2.A74

- Symbolic debugging example. An example of symbolic debugging using the L command is shown below.

  - Check for valid labels and symbols with command

    Use global labels, local labels, and section local labels to verify ones that are valid.

    1. "] L GLOBALSTART,FIN1<RET>"
       Global label specification valid in entire range.

    2. "]L LOCAL1START,FIN1<RET>"
       Specification using local label in file ID1.A74.

    3. "]L LOCAL2START,FIN2<RET>"
       Specification using local label in file ID2.A74.

4. "]L ??SECTION1START,FIN1<RET>"
   Specification using section local label in section "PROG1" (section no. 1).

5. "]L ??SECTION2START,FIN2<RET>"
   Specification using section local label in section "PROG2" (section no. 2).

- Priority of labels and symbols displayed during disassembly.
  The priorities of immediate values, addresses, and bit symbols are checked. The operand is "0" and the following items are displayed as shown in Table 4.1.

  - ZADDRESS (global label)
  - GLOBALDATA (global symbol)
  - LOCAL1DATA (local symbol)
  - LOCAL2DATA (local symbol)

  Check is made in the following sequence.

  (1) AND #00 (immediate value)
  (2) AND 00 (address)
  (3) CLB 0,00 (bit symbol)

  The valid labels and symbols are specified with the SCOPE command. Examples are provided for startup specification and three instances of the SCOPE command.

1. Startup

The default at startup is equivalent to specifying "SCOPE<RET>". The program counter at startup is E000H (reset) and local labels, local symbols, and local bit symbols in the file ID1.A74 are valid for commands.

```
]L GLOBALSTART,FIN1<RET>
E006 LOCAL1START:      AND    #00 :LOCAL1DATA
E008                   AND    00 :ZADDRESS
E00A                   CLB    0,00 :LOCAL1BIT
E00C                   JMP    F000
E00F FIN1:             NOP
]L LOCAL1START,FIN1<RET>
E006 LOCAL1START:      AND    #00 :LOCAL1DATA
E008                   AND    00 :ZADDRESS
E00A                   CLB    0,00 :LOCAL1BIT
E00C                   JMP    F000
E00F FIN1:             NOP
]L LOCAL2START,FIN2<RET>

?
]L ??SECTION1START,FIN1<RET>

?
]L ??SECTION2START,FIN2<RET>

?
```

2. Relocatable file name (.R74) specification

In this case, ID2.R74 is specified. The local labels, local symbols, and local bit symbols defined in the file ID2.A74 are valid.

```
]SCOPE ID2.R74<RET>
]L GLOBALSTART,FIN1<RET>
E006 GLOBALSTART:      AND    #00 :LOCAL2DATA
E008                   AND    00 :ZADDRESS
E00A                   CLB    0,00 :LOCAL2BIT
E00C                   JMP    F000 :LOCAL2START
E00F FIN1:             NOP
]L LOCAL1START,FIN1<RET>

?
]L LOCAL2START,FIN2<RET>
F000 LOCAL2START:      NOP
F001                   ORA    #00 :LOCAL2DATA
F003                   ORA    00 :ZADDRESS
F005                   SEB    0,00 :LOCAL2BIT
F007                   JMP    E006 :GLOBALSTART
F00A FIN2:             NOP
]L ??SECTION1START,FIN1<RET>

?
]L ??SECTION2START,FIN2<RET>

?
```

3. Section no. specification

In this case, "1(PROG1 section)" is specified. Specifying the section number enables section local labels. In this case, the section local labels defined in section "PROG1" are valid.

Furthermore, local labels defined in file ID1.A74, in which section "PROG1" belongs, are also valid.

```
]SCOPE 1<RET>
]L GLOBALSTART,FIN1<RET>
E006 ??SECTION1START: AND   #00 :LOCAL1DATA
E008                  AND   00 :ZADDRESS
E00A                  CLB   0,00 :LOCAL1BIT
E00C                  JMP   F000
E00F FIN1:            NOP
]L LOCAL1START,FIN1<RET>
E006 ??SECTION1START: AND   #00 :LOCAL1DATA
E008                  AND   00 :ZADDRESS
E00A                  CLB   0,00 :LOCAL1BIT
E00C                  JMP   F000
E00F FIN1:            NOP
]L LOCAL2START,FIN2<RET>

?
]L ??SECTION1START,FIN1<RET>
E006 ??SECTION1START: AND   #00 :LOCAL1DATA
E008                  AND   00 :ZADDRESS
E00A                  CLB   0,00 :LOCAL1BIT
E00C                  JMP   F000
E00F FIN1:            NOP
]L ??SECTION2START,FIN2<RET>

?
```

4. Specification without parameter

The result is the same as startup if no parameter is specified with the SCOPE command.

```
]SCOPE<RET>
]L GLOBALSTART,FIN1<RET>
E006 LOCAL1START:      AND    #00 :LOCAL1DATA
E008                   AND    00 :ZADDRESS
E00A                   CLB    0,00 :LOCAL1BIT
E00C                   JMP    F000
E00F FIN1:             NOP
]L LOCAL1START,FIN1<RET>
E006 LOCAL1START:      AND    #00 :LOCAL1DATA
E008                   AND    00 :ZADDRESS
E00A                   CLB    0,00 :LOCAL1BIT
E00C                   JMP    F000
E00F FIN1:             NOP
]L LOCAL2START,FIN2<RET>
?
]L ??SECTION1START,FIN1<RET>
?
]L ??SECTION2START,FIN2<RET>
?
]
```

4.2 Source Line Debugging Function

The source line debugging function enables the use of line numbers in the source file to specify addresses and execute the GL command. It also displays the source line corresponding to the executing address for the A, D, L, and X commands.

The source file must reside in the current directory in order to use the source line debugging function.

- Operation example

A simple program example is used to describe the source line debugging function.

1. Sample program (SAMPLE.A74)

The range of instructions for source line debugging must be specified with the pseudo instructions ".FUNC - .ENDFUNC".

```
:WORK   .EQU            0
        .SECTION        P
        .ORG            0E000H
RESET:
    SEI
    CLT
    CLD
    S = 07FH
    .FUNC           SAMPLE
    [WORK] = ++[WORK]
    X = 0FFH
    FOR  --X
        Y = 0FFH
        DO
            IF  Y & 0FH
                [WORK] = ++[WORK]
            ENDIF
        WHILE  --Y
    NEXT
    .ENDFUNC        SAMPLE

        .ORG            0FFFEH
        .WORD           RESET
        .END
```

2. Assemble link example

```
A>SRA74 SAMPLE<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1  ( SAMPLE.A74 )

now processing pass 2  ( SAMPLE.A74 )

ERROR   COUNT             00000
WARNING COUNT             00000
STRUCTURED STATEMENT      00011 LINES
TOTAL   LINE ( SOURCE ) 00024 LINES
TOTAL   LINE ( OBJECT ) 00024 LINES
COMMENT LINE ( SOURCE ) 00001 LINES
COMMENT LINE ( OBJECT ) 00001 LINES
OBJECT  SIZE ( P      ) 00029 (001D) BYTES

A>LINK74 SAMPLE,,,-S<RET>
MELPS 740 LINKER V.1.01.01C
Copyright 1989,1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1
processing "SAMPLE.R74"
now processing pass 2
processing "SAMPLE.R74"
TOTAL ROM SIZE 8192 (2000H) BYTES
TOTAL RAM SIZE 0 (0H) BYTES
```

3. Operation example

(a) SHOW SOURCE command

The SHOW SOURCE command displays absolute addresses during source line debugging.

```
]I SAMPLE<RET>
GLOBAL SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
SAMPLE.HEX SAMPLE.SYM LOAD END --> 1 SYMBOLS DEFINED
]SHOW SOURCE SAMPLE.A74,2<RET>
FILE SAMPLE.A74 , LINE 2
     2              .SECTION        P
     3              .ORG            0E000H
     4      RESET:
     5              SEI
     6              CLT
     7              CLD
     8              S = 07FH
     9              .FUNC           SAMPLE
    10 E006         [WORK] = ++[WORK]
    11 E008         X = 0FFH
    12 E00A         FOR  --X
    13 E00D             Y = 0FFH
    14 E00F             DO
    15 E00F                 IF  Y & 0FH
    16 E014                     [WORK] = ++[WORK]
    17 E016                 ENDIF
    18 E016             WHILE  --Y
    19 E019         NEXT
    20              .ENDFUNC        SAMPLE
    21      ;
    22              .ORG            0FFFEH
]
```

(b) Address specification

If the source file name is omitted in the specification ".line no.[source file name]", the source file corresponding to the current program counter is assumed.

```
]D .12[SAMPLE.A74],.19[SAMPLE.A74]<RET>
E00A CA F0 0E A0 FF 98   ......
E010 29 0F F0 02 E6 00 88 D0 F6 80   )........
]
```

(c) A, L, T, X command

The source line corresponding to each address is displayed together with the source file name and line number.

```
]L .12,.18<RET>
SAMPLE.A74        12            FOR  --X
E00A                  DEX
E00B                  BEQ   E01B
SAMPLE.A74        13                Y = 0FFH
E00D                  LDY   #FF
SAMPLE.A74        14                DO
SAMPLE.A74        15                    IF  Y & 0FH
E00F                  TYA
E010                  AND   #0F
E012                  BEQ   E016
SAMPLE.A74        16                        [WORK] = ++[WORK]
E014                  INC   00 :WORK
SAMPLE.A74        17                    ENDIF
SAMPLE.A74        18                WHILE  --Y
E016                  DEY
]Z<RET>
]X<RET>
A=04  X=7F  Y=FE  F=-----I-C  S=07F  PC=E000
      SEI
]XP<RET>
PC=E000 .12
]X<RET>
SAMPLE.A74        12            FOR  --X
A=04  X=7F  Y=FE  F=-----I-C  S=07F  PC=E00A
      DEX
]T 3<RET>
SAMPLE.A74        12            FOR  --X
A=04  X=7F  Y=FE  F=-----I-C  S=07F  PC=E00A
      DEX
A=04  X=7E  Y=FE  F=---B-I-C  S=07F  PC=E00B
      BEQ   E01B
SAMPLE.A74        13                Y = 0FFH
A=04  X=7E  Y=FE  F=---B-I-C  S=07F  PC=E00D
      LDY   #FF
*E00F
]A .12<RET>
```

```
SAMPLE.A74         12           FOR   --X
                  E00A  NOP<RET>
                  E00B  NOP<RET>
                  E00C  NOP<RET>
SAMPLE.A74         13                 Y = 0FFH
                  E00D  NOP<RET>
                  E00E  <RET>
]
```

PART II. PC4600 SYSTEM

CHAPTER 1. OVERVIEW

This chapter describes the use of RTT74 with PC4600.

The description of the following commands which are common with Part I are omitted. Refer to Part I for the details of these commands.

A, D, F, I, L, M, O, QUIT, S, SCOPE, SI, SHOW SOURCE, T, U, X, Z, ?, !, ;

RTT74 controls the PC4600 system from a personal computer running under MS-DOS and enables debugging of MELPS 740 series application programs.

1.1 Function

The following functions have been added or enhanced compared to the use of RTT74 with option board systems.

- Additional functions

  - Runtime debugging function
  - Coverage analysis function

- Enhanced functions

  - Realtime trace
  - Break condition

Due to the above functional enhancements, the following commands have been enhanced or modified compared to the use of RTT74 with option board systems.

	Command
Additional commands	BP, CV, GB, MAP, MCU, QP, STOP
Modify commands	G, QC, QD, QL
Deleted commands	GL, P

1.1.1 Runtime Debugging Function

- Runtime command mode is entered when an MCU execution command (G, GB) is executed from the normal command mode.
- The prompt "Go>" is displayed in runtime command mode.
- Runtime command mode enables the use of commands while the target MCU is executing (excluding some instructions). (Table 1.1)
- Use the STOP command to stop the execution of the target MCU and exit the runtime command mode to return to the normal command mode.

Table 1.1 Commands that Can be Used in Runtime Command Mode

Operation	Command	Restrictions
Memory access	A, D, F, L, M, S	None
Execution control	X	Reference only
	STOP	None
Realtime trace	QD, QL	If the trace point is not executed, BEFORE mode display from the point where the command is executed.
Software analysis	CV	None
Symbol operation	SCOPE, SI, SHOW SOURCE	None
Utility	?, !, ;	None

1.1.2 Coverage Analysis Function

- The coverage analysis function records the address executed or accessed by the MCU and displays them after execution is stopped. This is useful in detecting unexecuted code and evaluating program reliability.
- The coverage is measured from the start of program execution to the end.

1.1.3 Realtime Trace Function

The following functions have been enhanced compared with the use with the option board system.

- Up to 8192 cycles can be traced.
- Breakpoints and trace points can be separated to provide more detailed trace information.
- Trace points can be set for up to 6 addresses and one external trigger.
- The following trace conditions are available:

  1. Trace Before Trace-Point
     Break asynchronous BEFORE displaying 8192 cycles prior to the realtime trace point.

  2. Trace About Trace-Point
     Break asynchronous ABOUT displaying 8192 cycles about the realtime trace point.

  3. Trace After Trace-Point
     Break asynchronous AFTER displaying 8192 cycles after the realtime trace point.

4. Trace Before Break-Point
   Break synchronous BEFORE displaying 8192 cycles prior to the breakpoint.

1.1.4 Break Condition

- Breakpoint

  A break occurs at a breakpoint when the pass count, data, and access conditions for that breakpoint are all satisfied. Two address points can be specified. Twenty-eight sequential combination conditions with the trigger (rising/falling edge) input from an external trace cable, and two combinations of specified addresses (6) and pass count for each and trigger input from external trace cable can be set for a total of 30 different break conditions.

- Protect break

  A break also occurs when a protected area is accessed or when an attempt is made to write to a read-only area.

1.1.5 Flags

The PC4600 system sets the following flags to "1" after a command is executed or when a program is stopped.

Z command:	I flag (interrupt disable flag)
After program execution:	B flag (break flag)
Single step execution:	B flag and I flag

1.2 Input Files

RTT74 uses the following files:

- Hexadecimal file (HEX file)

  This file contains machine language data generated by LINK74. File extension is .HEX.

- Symbol file (SYM file)

  This file contains symbol information and source line debugging information. A SYM file is generated when the "-S" option is specified with LINK74. File extension is .SYM.

- Help screen file (HLP file)

  This is a normal text file that contains help screens describing how to use RTT74. The file name is RTT74.HLP. The screens consist of a list of commands and a detailed description of each command.

- MCU file
  This file contains MCU specific information. The file name is Mxxxxx.MCU.

The following points are different, compared to the use of RTT74 with the option board system.

- Data file is unnecessary
  When using RTT74 with option board system, a data file (RTT74.DAT) containing information specific to each option board is required during startup. However, this file is not required when using RTT74 with PC4600 system.

- MCU file may be required during MCU command execution
  The PC4600 system supports various MCUs by replacing the emulator MCU. With RTT74, emulator MCU specific information is specified with the MCU command. If the MCU name is specified as an argument, the MCU file containing the information for the specified chip is required.

1.3 System Configuration

1.3.1 Hardware Configuration

The following devices are required to use RTT74:

1. Personal computer

2. PC4000E debugger

   (a) PC4000E

   (b) Serial port

3. PC4600 debugger

   (a) PC4600

   (b) M38000T-POD

4. Emulator MCU

Figure 1.1 shows the configuration of these devices.

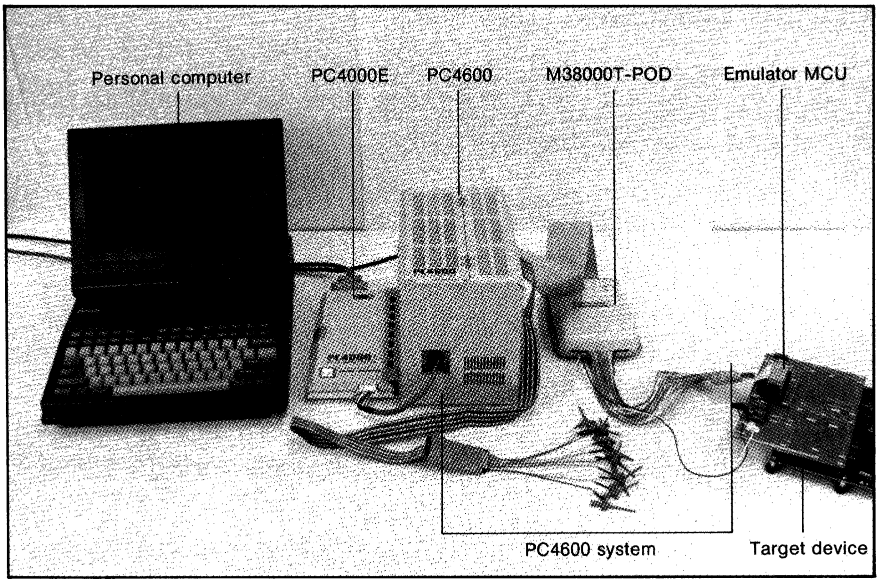


Figure 1.1 System Configuration

1.3.2 Initial Setting of the Personal Computer

A personal computer and a PC4000E communicate using serial I/O. For fast RTT74 processing, it is necessary to set the baud rate of the personal computer and the PC4000E at 9600 bps.

1. The setup procedures for personal computers differ with each model. Refer to Appendix C, which lists the setup procedures for the personal computers that can be used with RTT74.

2. The PC4000E baud rate can be selected by removing the front panel to access the jumper switches on the main board. For specific instructions, refer to the PC4000E User's Manual.

1.3.3 Cable connections

A cable must be connected between the personal computer and the PC4000E to transfer the data through the serial I/O. Cable connections differ with each personal computer model. For cable connection procedures, refer to Appendix D.

CHAPTER 2. OPERATION

2.1 Starting the Program

2.1.1 Normal Startup

RTT74 is started from the MS-DOS command prompt after turning on the PC4000E power.

2.1.2 Startup Options

RTT74 provides options shown in Table 2.1. The options can be specified in uppercase or lowercase.

Table 2.1 Startup Options

Option	Description
-HOST	Specifies the host computer on which RTT74 executes. Either "PC9801" or "IBMPC" can be specified. Example: A>RTT74 -HOST=PC9801
-I	Specifies the directory in which the DAT file and HLP file reside. Example: A>RTT74 -IA:\MSC\TOOL
-MCU	Specifies the target MCU. The MCU file is loaded and the MCU information is set in PC4600. Example: Z>RTT74 -MCU=M37450
filename	Specifies the name of the file to be loaded during startup. If the file extension is omitted, Files with extensions.SYM and .HEX are loaded. Example: A>RTT74 -HOST=PC9801 SAMPLE

2.1.3 Normal Startup Screen

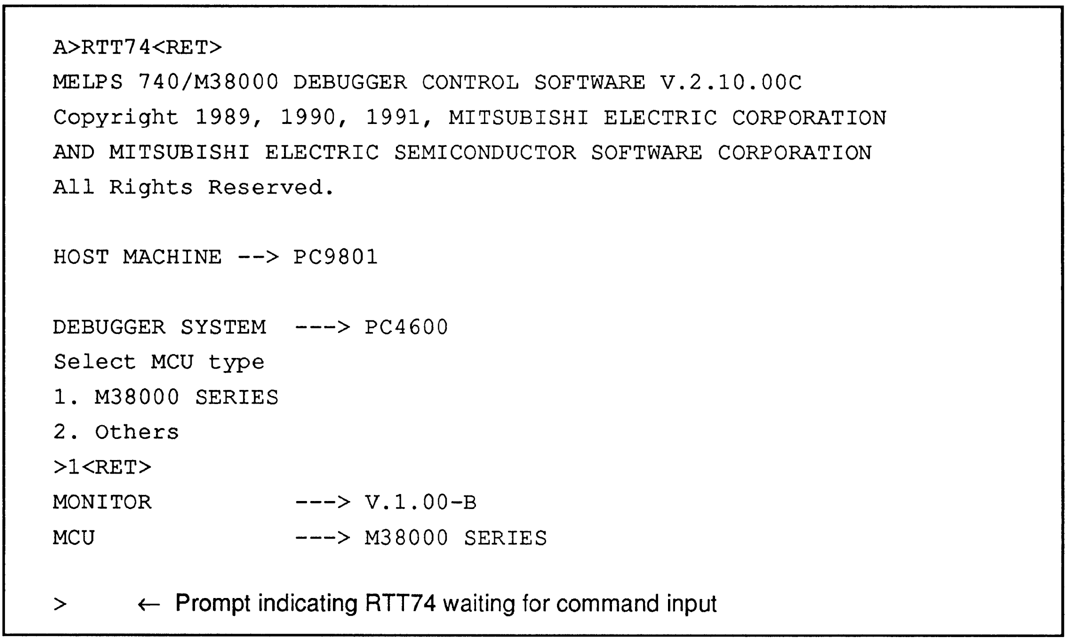
The following messages are displayed when RTT74 starts up normally.

```
A>RTT74<RET>
  MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
  Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
  AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
  All Rights Reserved.
  HOST MACHINE --> PC9801
  DEBUGGER SYSTEM  ---> PC4600
  Select MCU type
  1. M38000 SERIES
  2. Others
  >
```

- Enter 1 if the MCU is M38000 and 2 if otherwise.
- Normal command mode is entered when MCU is specified and RTT74 starts normally.

The startup example for each case is shown in Figure 2.1 and Figure 2.2.

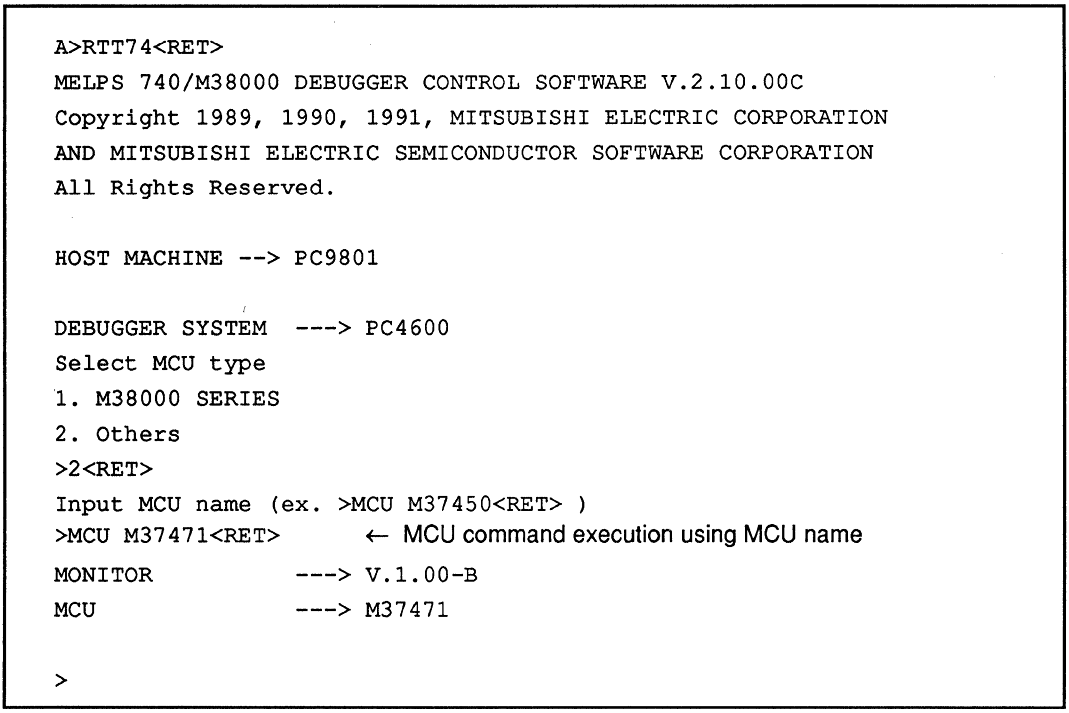
1. When M38000 Series is selected



```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>1<RET>
MONITOR          ---> V.1.00-B
MCU              ---> M38000 SERIES

>     ← Prompt indicating RTT74 waiting for command input
```

Figure 2.1 When M38000 Series is Selected

2. When others is selected

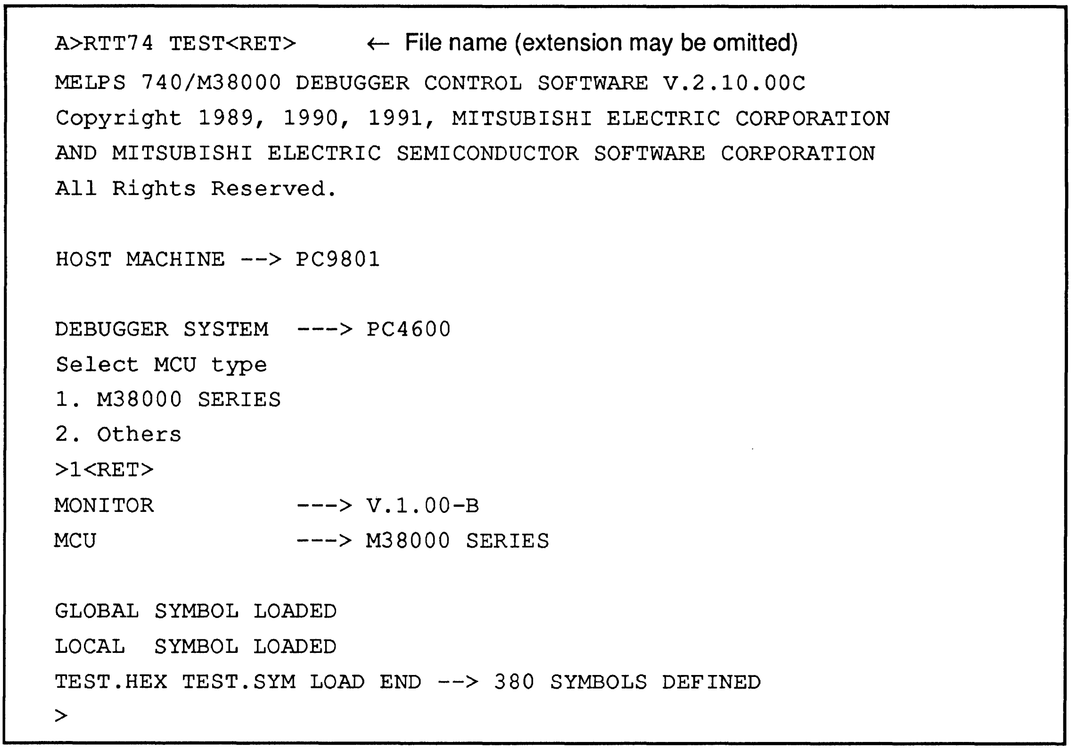


```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>2<RET>
Input MCU name (ex. >MCU M37450<RET> )
>MCU M37471<RET>        ← MCU command execution using MCU name
MONITOR          ---> V.1.00-B
MCU              ---> M37471

>
```

Figure 2.2 When Others is Selected

2.1.4 Startup Screen when a File Name is Specified

The following screen is displayed when the load file is specified during startup. If the file extension is omitted, files with extensions .SYM and .HEX are loaded.



```
A>RTT74 TEST<RET>        ← File name (extension may be omitted)
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>1<RET>
MONITOR          ---> V.1.00-B
MCU              ---> M38000 SERIES

GLOBAL SYMBOL LOADED
LOCAL  SYMBOL LOADED
TEST.HEX TEST.SYM LOAD END --> 380 SYMBOLS DEFINED
>
```

2.1.5 Compatible PC Startup Screen

When using a compatible PC supported by RTT74, the PC model can be specified as follows.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
Can't support this host machine.

   SELECT NEXT COMPATIBLE MACHINE TYPE.

       NEC PC-98 SERIES    ..... 1
       MULTI16-IV          ..... 2
       IBM PC/XT/AT        ..... 3
       EXIT                ..... OTHER

3<RET>          ← Specify the machine with a number.

HOST MACHINE --> IBM PC/XT/AT
DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>1<RET>
MONITOR          ---> V.1.00-B
MCU              ---> M38000 SERIES

>
```

2.1.6 Abnormal Startup Screen

The following message is displayed if RTT74 cannot start normally due to communication error between the PC and PC4600E. In this case, terminate RTT74 with ^C (Ctrl+C) and check the PC4000E power, serial cable connections, and PC and PC4600E baud rates.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

CONNECTING DEBUGGER        ← Halt operation
^C                         ← Enter ^C
A>
```

2.2 Ending the Program

Enter "QUIT<RET>" at the command line to stop RTT74. The following termination screen is displayed.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>1<RET>
MONITOR          ---> V.1.00-B
MCU              ---> M38000 SERIES

>
      :
      :
      :
>QUIT<RET>
A>
```

2.3 MCU Files

In order to use RTT74 with the PC4600 system, MCU specific information must be set as shown in the startup example.

The MCU command is used for this purpose. The MCU command uses the MCU name as argument and sets the data loaded from the MCU file.

The MCU file contains the following data corresponding to the MCU command argument.

- Stack page selection bit
- MCU mode register address
  Address of SFR containing the stack page selection bit
- Stack end
- Reset vector

The MCU file can be created as follows using an editor program.

- File name
  File name must be "MCU name.MCU".

  - Example: if the MCU is M37471
    M37471.MCU

  However, M38000.MCU corresponds to all MCUs in the M38000 series.

- Items
  Specify each of the following items in hexadecimal one item to a line (hexadecimal number may be in upper or lowercase)

  1. Stack page selection bit number
  2. MCU mode register address
  3. Stack end
  4. Reset vector

- (Example) M37471.MCU

`2`	Stack page selection number
`FB`	MCU mode register address
`7F`	Stack end
`FFFE`	Reset vector

Note: Specify only necessary items.

2.4 Command Descriptions

RTT74 provides 30 commands shown in Table 2.2. The command character is usually the mnemonic of the operation to be performed.

Table 2.2 Commands

Commands	Function
A	Assembles line-by-line
BP	Sets and displays breakpoints
CV	Measures coverage
D	Displays memory contents in hexadecimal and ASCII data
F	Writes specified data in specified memory area
G	Executes a program without breakpoints.
GB	Executes program under the break condition specified with the BP command.
I	Loads the contents of a file (HEX, SYM) into RTT74.
L	Disassembles and displays contents of program memory.
M	Performs block transfer of memory contents.
MAP	Sets and access debug memory space mapping information.
MCU	Sets target MCU specific information.
O	Stores program memory contents into a file (HEX file).
QC	Sets the realtime trace of break conditions.
QD	Displays realtime trace results (memory dump).
QL	Displays realtime trace results (disassembly).
QP	Sets and displays realtime trace points.
QUIT	Ends RTT74.
S	Displays and changes memory contents.
SCOPE	Specifies scope of local identifier.
SI	Displays section information.
SHOW SOURCE	Displays source list and corresponding address.
STOP	Stops execution of the target MCU and returns to normal command mode.
T	Executes program one instruction at a time and displays the internal status.
U	Executes only the specified number of instructions. (Displays only internal status before execution).
X	Displays and changes register contents.
Z	Resets target MCU.
?	Displays a list of RTT74 commands and their usage.
!	Executes an MS-DOS command from RTT74.
;	Indicates comment line input.

2.5 Return Code to MS-DOS

RTT74 always returns 0 to MS-DOS.

2.6 Environment Variables

RTT74 uses MS-DOS environment variables as follows:

- The path for searching DAT file and HLP file can be specified. Use the environment variable "740DAT" to specify the path name. RTT74 searches for DAT and HLP files in the following sequence:

  1. Current directory

  2. Directory specified with command parameter "-I"

  3. Path specified with environment variable "740DAT"

- Environment variable "740DAT"setting example

```
A>SET 740DAT=A:\USR\DAT
        ↑           ↑
    Environment variable  Path name
```

CHAPTER 3. COMMANDS

Compared to the option board system, the following commands are different for PC4600 system.

Table 3.1 Different Commands

	Command
Added	BP, CV, GB, MAP, MCU, QP, STOP
Modified	G, QC, QD, QL
Deleted	GL, P

This chapter describes the syntax and examples of added and modified commands.

3.1 Syntax

The symbols used in the syntax description are described below.

3.1.1 Input Format

Command name   Argument,...

- Command name
  - Uppercase or lowercase character string.
  - No distinction is made between uppercase and lowercase characters.
- Argument
  - A number, string, label, symbol, or expression required by the command.
  - The argument depends on the command. If more than one argument is specified, the arguments must be separated by a comma.

3.1.2 Symbols

- Only the first 20 characters are used to identify a symbol. More than 20 characters can be used, but the remaining characters are ignored.
- Uppercase and lowercase characters are distinguished.
- Only the symbols defined in the loaded SYM file can be used.

3.1.3 Numeric Values

- RTT74 treats all input numbers as hexadecimal numbers.
- However, the items are treated as decimals:

  - Line number when source line is used to specify an address

  - QD, QL command arguments

  - Line numbers for SHOW SOURCE command

  - Pass count for BP and QP commands

  - Break combination numbers in BP and QP commands ("BP C", "QP C")

3.1.4 Register Notation

Table 3.2 Register Notation

Register names	Register notations
Accumulator A	A
Index register X	X
Index register Y	Y
Stack pointers S	S
Program counter PC	P
Processor status register PS	F

3.1.5 Flag Notation

Table 3.3 Flag Notation

Flag names	Flag notations
Negative flag	N
Overflow flag	V
Index X register mode flag	T
Break flag	B
Decimal mode flag	D
Interrupt disabled flag	I
Zero flag	Z
Carry flag	C

3.1.6 Flag Display

The flag status display for the T, U, and X command shows the name of the flag if the flag is set and "-" if not.

(display example)

```
A=FF  X=FF  Y=FF  F=N----I--  S=0F6  PC=E000 INITIAL:
```

When flags N and I are set.

Figure 3.1 Processor Status Register Display Example

3.2 Additional and Modified Commands

[Example]

Command Name (description) **Function summary**

Command Syntax

The command syntax is described.

Command syntax 1
Command syntax 2

Command Execution Example

The command execution example and result are shown.

(Example)

```
>SI<RET>         ← Command input example
#SECTION INFORMATION
No.   NAME     OBJECT        TYPE  START  LENGTH  SOURCE        LIBRARY
0000  RAMAREA  SAMPLE0.R74   RAM   0000   0100    SAMPLE0.A74
0001  PROG1    SAMPLE1.R74   ROM   E000   0074    SAMPLE1.A74
0003  PROG1    SAMPLE2.R74   ROM   E074   0077    SAMPLE2.A74
0002  PROG2    SAMPLE1.R74   ROM   F000   0072    SAMPLE1.A74
>
```

Description

- The command description, precautions, and usage are described.

BP (breakpoint) Access, set breakpoint (batch format)

Command Syntax

- Set address breakpoint 1
  BP A1[=[pass count],[address][:mask],[data][:mask],[access condition]]

- Set address breakpoint 2
  BP A2[=[pass count],[address][:mask],[data][:mask],[access condition]]

- Set address breakpoint 3
  BP A3[=[pass count],[address][:mask]]

- Set address breakpoint 4
  BP A4[=[pass count],[address][:mask]]

- Set address breakpoint 5
  BP A4[=[pass count],[address][:mask]]

- Set address breakpoint 6
  BP A6[=[pass count],[address][:mask]]

- Set trigger breakpoint
  BP T[=[pass count],[H or L]]
  H is rising edge and L is falling edge

- Set break sequence
  BP C[=[pass count(1-30)]]

- Set protect break
  BP PROTECT[=WRITE (or W), ACCESS (or A), DIS]]

Command Execution Example

```
>BP A1<RET>                                        (Ex 1)
A1 = 00001,  0000H:0000H, 00H:00H, FETCH
A1 Pass=<RET>
A1 Address=WORKS:0F<RET>
A1 Data=0:0FF<RET>
A1 Access=W<RET>
A1 = 00001, WORKS 0050H:000FH, 00H:FFH, WRITE
>BP A1=<RET>                                       (Ex 2)
A1 = 00001, WORKS 0050H:000FH, 00H:FFH, WRITE
>BP T=1,H<RET>
T = 00001, HIGH
>BP C<RET>                                         (Ex 3)
 1  A1*A1P                            15  ((A1+A2)*T)*CP
 2  A2*A2P                            16  ((A1+T)*A2)*CP
 3  T*TP                              17  (A1*A1P)->(A2*A2P)
 4  (A1*A1P)+(A2*A2P)                 18  (A1*A1P)->(T*TP)
 5  (A1*A1P)+(T*TP)                   19  (T*TP)->(A1*A1P)
 6  (A1*A1P)+(A2*A2P)+(T*TP)          20  (A1->A2)*CP
 7  (A1*A1P)*(A2*A2P)                 21  (A1->T)*CP
 8  (A1*A1P)*(T*TP)                   22  (T->A1)*CP
 9  (A1*A1P)*(A2*A2P)*(T*TP)          23  (A1*A1P)->(A2*A2P)->(T*TP)
10  (A1*A2)*CP                        24  (A1*A1P)->(T*TP)->(A2*A2P)
11  (A1*T)*CP                         25  (T*TP)->(A1*A1P)->(A2*A2P)
12  (A1*A2*T)*CP                      26  (A1->A2->T)*CP
13  ((A1*A2)+T)*CP                    27  (A1->T->A2)*CP
14  ((A1*T)+A2)*CP                    28  (T->A1->A2)*CP
29  (A1*A1P)+(A2*A2P)+(A3*A3P)+(A4*A4P)+(A5*A5P)+(A6*A6P)
30  (A1*A1P)+(A2*A2P)+(A3*A3P)+(A4*A4P)+(A5*A5P)+(A6*A6P)+(T*TP)
Condition  1. A1*A1P
C=21<RET>
C = 21. (A1->T)*CP
>BP PROTECT=DIS<RET>
PROTECT = DIS
>
```

Description

- The BP command command sets and displays breakpoints.
- Entering the breakpoint only as "BP A1<RET>" enables break conditions to be entered interactively. (Example 1)
- Entering the breakpoint and equal sign as "BP A1=<RET>" displays the currently set breakpoints. (Example 2)
- The "BP C" command can be used to specify the breakpoint passing sequence for breakpoints (A1, A2, T). (Example 3)
- The pass count for the entire combination can be set with the CP (Common Pass) command.
- Combination condition can be selected from the 30 conditions shown in Table 3.4.
- A period indicates the end of interactive specification.
- Mask data is enabled when 0 and masked when 1. Therefore, data match is not checked if FFH is specified as mask data.
- The following conditions can be specified as protect break detection conditions.
  - ACCESS (or A)
    A break occurs if an inaccessible area (set to DIS with the MAP command) is accessed, or if attempt is made to write to write-protected area (set to ROM with the MAP command).
  - WRITE (or W)
    A break occurs when an attempt is made to write to write-protected area (set to ROM with the MAP command).
  - DIS
    Disables protect break.
- The following three access conditions can be specified.
  - READ (or R)
    Reads from specified address (including FETCH condition)
  - WRITE (or W)
    Writes to specified address.
  - RW
    Reads or writes to specified address.
  - FETCH (or F)
    Fetches instruction from the specified address.

- Table 3.4 shows the break sequence numbers.

Table 3.4 Break Sequence Numbers

No.	Combination	No.	Combination
1	A1*A1P	15	((A1+A2)*T)*CP
2	A2*A2P	16	((A1+T)*A2)*CP
3	T*TP	17	(A1*A1P)->(A2*A2P)
4	(A1*A1P)+(A2*A2P)	18	(A1*A1P)->(T*TP)
5	(A1*A1P)+(T*TP)	19	(T*TP)->(A1*A1P)
6	(A1*A1P)*(A2*A2P)+(T*TP)	20	(A1->A2)*CP
7	(A1*A1P)*(A2*A2P)	21	(A1->T)*CP
8	(A1*A1P)*(T*TP)	22	(T->A1)*CP
9	(A1*A1P)*(A2*A2P)*(T*TP)	23	(A1*A1P)->(A2*A2P)->(T*TP)
10	(A1*A2)*CP	24	(A1*A1P)->(T*TP)->(A2*A2P)
11	(A1*T)*CP	25	(T*TP)->(A1*A1P)->(A2*A2P)
12	(A1*A2*T)*CP	26	(A1->A2->T)*CP
13	((A1*A2)+T)*CP	27	(A1->T->A2)*CP
14	((A1*T)+A2)*CP	28	(T->A1->A2)*CP
29	(A1*A1P)+(A2*A2P)+....+(A5*A5P)+(A6*A6P)		
30	(A1*A1P)+....+(A6*A6P)+(T*TP)		

- The symbols used in the condition are shown in Table 3.5.

Table 3.5 Symbols in Condition

Symbol	Description
A1	Address of breakpoint 1
A1P	Pass count of breakpoint 1
A2	Address of breakpoint 2
A2P	Pass count of breakpoint 2
A3-A6	Address of breakpoints 3-6
T	Trigger breakpoint level
TP	Trigger breakpoint pass count
CP	Common pass count
->	Sequential condition
+	OR condition
*	AND condition

- BP command error message

Error Message	Description
Invalid input value. (xxx)	Input value for "xxx" is invalid.
too many parameters	Too many parameters are specified.

BP (break point) — Access or set break point (menu format)

Command Syntax

BP

Command Execution Example

```
>BP<RET>
    :
    :    Go to menu mode
    :
********  Break Point Table **************************************
         Pass  Label      Address  (Mask)  Access H/L  Line [ Source  ]
  A1     00002 WORKS      0050H    (000FH) FETCH  --
  data   --                 FFH    (  00H) --     --
  A2     00001            0000H    (0000H) FETCH  --
  data   --                 00H    (  FFH) --     --
  A3     00001 SUB1       E720H    (0000H) --     --
  A4     00001 SUB3       E770H    (0000H) --     --
  A5     00001            0000H    (0000H) --     --
  A6     00001            0000H    (0000H) --     --
  T      00001            --       --      --     HIGH
  CP     00005
*********************************************************************
Condition   1. A1*A1P
PROTECT (DIS/WRITE(W)/ACCESS(A)) : ACCESS
Break Point><RET>          ← Return to RTT74 command prompt by pressing Return
>BP C=21<RET>              ← Specify break sequence condition
C = 21. (A1->T)*CP


>BP<RET>
    :
    :    Re-display menu
    :
********  Break Point Table *************************************
         Pass  Label      Address  (Mask)  Access H/L  Line [ Source  ]
  A1     00002 WORKS      0050H    (000FH) FETCH  --
  data   --                 FFH    (  00H) --     --
  A2     00001            0000H    (0000H) FETCH  --
  data   --                 00H    (  FFH) --     --
```

```
   A3     00001 SUB1       E720H    (0000H) --     --
   A4     00001 SUB3       E770H    (0000H) --     --
   A5     00001            0000H    (0000H) --     --
   A6     00001            0000H    (0000H) --     --
   T      00001            --       --      --     HIGH
   CP     00005
*********************************************************************
 Condition  21. (A1->T)*CP
 PROTECT (DIS/WRITE(W)/ACCESS(A)) : DIS
 Break Point><RET>       ← Terminate the command
 >
```

Description

- The BP command enables breakpoints to be set in menu format. The menu displays the current breakpoints and allows them to be changed on screen. Use the cursor movement keys (↑ or ^E, ↓ or ^X, ← , → or ^D) to move the cursor to the value to be changed and enter the value.
- To terminate the BP command, enter "<RET>" at the "Break Point>" prompt.
- The menu format is available only on personal computers that use ANSI escape sequences.

CV (Coverage Analysis)

Set and display coverage analysis

Command Syntax

- Display coverage analysis result
  CV LOCAL [start address][,end address]
  CV GLOBAL [start address][,end address]
  CV TOTAL [start address][,end address]

- Initialize coverage memory
  CV CLEAR

Command Execution Example

```
  >CV CLEAR<RET>                                  (Ex 1)
  >GB <RET>
  Go>
  Break at E124H
  Trace Point Passed  Trace Mode = Trace Before Trace-Point
  Time = 0h 0m 0s  125m 65u sec
  >CV LOCAL e000,e111<RET>                        (Ex 2)
  ADDRESS.>
0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
   E000 *****************      *************      ****************   *
   E040 ********                                                     *
   E080        *****************     ************   *****************
   E0C0      ********  *
   E100              *****
  >CV GLOBAL<RET>                                 (Ex 3)
  ADDRESS.>0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
   E000 FFFF81FFF07FFF81FF0000000000000101FFFF07FF81FFFF07F9000000000000
   E100 0007C
  >CV TOTAL<RET>                                  (Ex 4)
  Coverage Percent (E000H-E111H) = 42.33 %
  >
```

Description

- The CV command is used to set and display coverage analysis.

- The coverage analysis result is displayed in three different formats: CV LOCAL, CV GLOBAL, and CV TOTAL.

- To start a new coverage analysis, initialize the coverage memory with the CV CLEAR command and then execute the target MCU. (Example 1)

- CV LOCAL displays the coverage analysis result in 1 byte units with "*" indicating accessed address and blank indicating unaccessed address. (Example 2)

- CV GLOBAL displays the coverage analysis result in 4 byte units with accessed result expressed as value between 0 and F. (Example 3)

  For example, if the access status of E110H to E113H is expressed as C, addresses E110H and E111H are accessed and E112H and E113H are unaccessed.

- CV TOTAL displays the coverage result as a percentage. (Example 4)
- If the address is omitted, the previously set value is used.
- Processing can be cancelled by pressing ^Z (Ctrl+Z) during execution.
- CV Command Error Messages

Error Message	Description
invalid 1st parameter	The first parameter is invalid.
invalid 2nd parameter	The second parameter is invalid.
too many parameters	There are too many parameters.
Can't get valid data from PC4600 to PC 4600	Correct data cannot be obtained from the PC4600 system.
checksum error	Correct data cannot be obtained from the PC4600 system.

G (go) — Execute without break

Command Syntax

G [start address]

Command Execution Example

```
>G start<RET>
Go>STOP<RET>
Terminate at E005H  exit
Time = 0h 0m 2s  751m 516u sec
>
```

Description

- The G command executes the program from the specified address without breakpoints.
- If the start address is omitted, execution is resumed from the address at which the MCU is currently halted.
- Other commands can be executed while the target MCU is running.

  1. The prompt "Go>" is displayed while the target MCU is running.
  2. When the STOP command is executed at the "Go>" prompt, the target MCU is stopped and control returns to normal command input mode (">").

- G Command Error Messages

Error Message	Description
Can't set Break (RTT) point data to PC 4600.	An attempt to set breakpoint or trace point has failed.
error occurred! MCU was reset automatically.	An error has occurred. The MCU is reset.

GB (go with break_table) Execute with break table condition

Command Syntax

GB [start address]

Command Execution Example

```
>GB start<RET>
Go><RET>
Break at E003H  stopadd
Trace Point Passed  Trace Mode = Trace About Trace-Point
Time = 0h 0m 0s  3m 650u sec
>
```

Description

- The GB command executes the program from the specified address until the break condition specified with the BP command is satisfied.

- If the start address is omitted, execution is resumed from the address at which the MCU is currently halted.

- Other commands can be executed while the target MCU is running.

  1. The prompt "Go>" is displayed while the target MCU is running.

  2. When the STOP command is executed at the "Go>" prompt, the target MCU is stopped and control returns to normal command input mode (">").

- Refer to section "4.3 Break Operation" in the "PC4600 System User's Manual" for the program counter after break.

- GB Command Error Messages

Error Message	Description
Can't set Break (RTT) point data to PC4600	An attempt to set breakpoint or trace point has failed.
error occurred! MCU was reset automatically.	An error has occurred. The MCU is reset.

MAP

Set and access mapping information

Command Syntax

MAP [start address,end address,{RAM|ROM|DIS},{EXT|INT}]
MAP [start address,L byte count,{RAM|ROM|DIS},{EXT|INT}]

Command Execution Example

```
>MAP 0,1FFF,RAM,EXT<RET>
>MAP 2000,FFFF,ROM,INT<RET>
>MAP<RET>
Memory Area    RAM/ROM EXT/INT
0000  -  1FFF  RAM     EXT
2000  -  FFFF  ROM     INT
>MAP 0,L10,DIS,EXT<RET>
>MAP<RET>
Memory Area    RAM/ROM EXT/INT
0000  -  000F  DIS     EXT
0010  -  1FFF  RAM     EXT
2000  -  FFFF  ROM     INT
>MAP F000,FFEF,RAM,INT<RET>
>MAP<RET>
Memory Area    RAM/ROM EXT/INT
0000  -  000F  DIS     EXT
0010  -  1FFF  RAM     EXT
2000  -  EFFF  ROM     INT
F000  -  FFEF  RAM     INT
FFF0  -  FFFF  ROM     INT
>
```

Description

- The MAP command specifies the attribute of the target MCU memory space. Each item is described below.

- RAM/ROM
  Displays the attribute of the memory space as read only (ROM), read/write (RAM), or access disabled (DIS). This attribute is used to detect PROTECT break.

- EXT/INT
  Specifies whether the memory space is PC4600 internal memory area (INT) or external memory area (EXT). If external memory area is specified, the system memory space is accessed.

- MAP Command Error Messages

Error Message	Description
invalid 1st parameter	First parameter is invalid.
invalid 2nd parameter	Second parameter is invalid.
invalid 3rd parameter	Third parameter is invalid.
invalid 4th parameter	Fourth parameter is invalid.
too few parameters	Insufficient number of parameters.
too many parameters	There are too many parameters.

MCU — Set MCU specific information

Command Syntax

MCU MCU name
MCU stack page selection bit no.,MCU mode register address,stack end, reset vector address

Command Execution Example

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>2<RET>                                            (Ex 1)
Input MCU name (ex. >MCU M37451<RET> )
>MCU M37471<RET>                                   (Ex 2)
MONITOR          ---> V.1.00
CPU              ---> M37471

  :
  :

>MCU 2,FE,7F,FFFE<RET>
>MCU M37471<RET>
>
```

Description

- The MCU command sets target MCU specific information.
- RTT74 asks for the MCU type when started. (See example)
  - If "1" (M38000 SERIES) is specified
    RTT74 sets MCU information for the M38000 SERIES.
  - If "2" (other MCU) is specified (Example 1)
    1. The prompt ">MCU" appears to prompt for the MCU command parameter.
    2. If an MCU file name is entered, the MCU information is input from the MCU file with the specified name. (Example 2)
    3. Each value can be entered separately.
    4. If the MCU file does not exist or if input parameter is invalid, RTT74 ends and returns to MS-DOS.
- MCU Command Error Messages

Error Message	Description
invalid 1st parameter	First parameter is invalid.
invalid 2nd parameter	Second parameter is invalid.
invalid 3rd parameter	Third parameter is invalid.
invalid 4th parameter	Fourth parameter is invalid.
too few parameters	Insufficient number of parameters.
too many parameters	There are too many parameters.
invalid MCU name	The MCU name is invalid.
Can't open "xxx"	The MCU file xxx cannot be opened.
Can't read "xxx"	Data cannot be read correctly from MCU file xxx.

QC (quest condition) Set, display realtime trace mode

Command Syntax

QC [store condition (1-4)]

Command Execution Example

```
>QC<RET>
1. Trace Before Trace-Point
2. Trace About Trace-Point
3. Trace After Trace-Point
4. Trace Before Break-Point

Select No.><RET>
1. Trace Before Trace-Point Selected
>QC 2<RET>                                   (Ex 1)
2. Trace About Trace-Point Selected
>QC<RET>
1. Trace Before Trace-Point
2. Trace About Trace-Point
3. Trace After Trace-Point
4. Trace Before Break-Point

Select No.>3<RET>                            (Ex 2)
3. Trace After Trace-Point Selected
>
```

Description

- The QC command sets realtime trace mode.
- Trace mode can be set in two ways as follows:
  1. With command parameter (Example 1)
  2. At QC command prompt (Example 2)
- The following trace areas can be selected:
  1. Trace Before Trace-Point
     Stores information prior to the trace point (set with QP command).
  2. Trace About Trace-Point
     Stores information around the trace point.
  3. Trace After Trace-Point
     Stores information after the trace point.
  4. Trace Before Break-Point
     Stores information prior to the break point (set with the BP command).
- Trace information for 8192 cycles is stored.
- QC Command Error Messages

Error Message	Description
too many parameters	There are too many parameters.
invalid parameter	The parameter is invalid.

QD (RD)(quest-data) Display realtime trace data

Command Syntax

- Absolute specification
  QD start cycle,end cycle

- Relative specification
  QD [display cycle count]

Command Execution Example

```
>QD -10<RET>                                          (Ex 1)
Trace Point Passed
Written by Trace Before Trace-Point Mode
 -------------------- MELPS 740/M38000 --------------------   - EXT --
 TCNT SYMBOL           Address  Data  Sync Read Write B-T Q-T 12345678
-0010 MAIN:            E014     C9    0    0    1     1   1   11111111
-0009                  00FF     E0    0    1    0     1   1   11111111
-0008                  00FE     14    0    1    0     1   1   11111111
-0007                  00FD     80    0    1    0     1   1   11111111
-0006                  FFF2     2D    0    0    1     1   1   11111111
-0005                  FFF3     E0    0    0    1     1   1   11111111
-0004 TXMODULE:        E02D     48    1    0    1     1   1   11111111
-0003                  E02E     8A    0    0    1     1   1   11111111
-0002                  00FC     00    0    1    0     1   1   11111111
-0001                  E02E     8A    1    0    1     1   1   11111111
 0000                  E02F     48    0    0    1     1   1   11111111

>QD -8191,-8181<RET>                                  (Ex 2)
Trace Point Passed
Written by Trace Before Trace-Point Mode
 -------------------- MELPS 740/M38000 --------------------   - EXT --
 TCNT SYMBOL           Address  Data  Sync Read Write B-T Q-T 12345678
-8191                  E041     40    0    0    1     1   1   11111111
-8190 WORK1:           0041     40    0    1    1     1   1   11111111
-8189                  00FC     00    0    0    1     1   1   11111111
-8188                  E041     40    1    0    1     1   1   11111111
-8187                  E042     EA    0    0    1     1   1   11111111
-8186                  00FC     EA    0    1    1     1   1   11111111
-8185                  00FD     80    0    0    1     1   1   11111111
-8184                  00FE     1D    0    0    1     1   1   11111111
-8183                  00FF     E0    0    0    1     1   1   11111111
-8182                  E01D     86    1    0    1     1   1   11111111
-8181                  E01E     42    0    0    1     1   1   11111111

>
```

Description

- The QD command displays the contents of the realtime trace memory as hexadecimal together with the external signal status.
- Realtime trace must be started with the G or GB command prior to executing this command.
- The name and function of the external signal displayed on screen are as follows:

  1. Sync
     Signal issued during instruction code fetch. '1' if instruction is being fetched.
  2. Read
     Signal indicating the data bus direction. '0' if read.
  3. Write
     Signal indicating the data bus direction. '0' if write.
  4. B-T
     Level of the break external trigger. '1' if High and '0' if Low.
  5. Q-T
     Level of the trace external trigger. '1' if High and '0' if Low.
  6. EXD
     Status of the 8-bit external signal.

- If 1 parameter is specified, it is treated as the display cycle count since the last QD or QL command (0 at startup or when G or GB command is executed). (Example 1)
- If two parameters are specified, they are assumed to be absolute cycle specifications. (Example 2)
- Execution can be stopped with ^Z.

- QD Command Error Messages

Error Message	Description
Can't get real-time-trace data from PC4600	Data could not be obtained properly from PC4600.
checksum error	Data could not be obtained properly from PC4600.
invalid 1st parameter	The first parameter is invalid.
invalid 2nd parameter	The second parameter is invalid.
too many parameters	There are too many parameters.

QL (RL)(quest-list) List realtime trace data

Command Syntax

- Absolute specification
  QL start cycle,end cycle

- Relative specification
  QL [display cycle count]

Command Execution Example

```
>QL -20<RET>                                            (Ex 1)
Trace Point Passed
Written by Trace Before Trace-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL               ADDRESS MNEMONIC
-0017                      E01F CMP    #0A
-0015                      E021 BCC    E028
   ++++  INTERRUPT ---> VECTOR ADDRESS in $FFF2
-0004 TXMODULE:            E02D PHA
-0001                      E02E TXA

>QL -8191,-8171<RET>                                    (Ex 2)
Trace Point Passed
Written by Trace Before Trace-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL               ADDRESS MNEMONIC
-8191                      E03E PLA
-8187                      E03F TAX
-8185                      E040 PLA
-8181                      E041 RTI
-8175                      E01D STX    42 :WORK2
-8171                      E01F ??=C9

>
```

Description

- The QL lists the contents of the realtime trace memory.
- Realtime trace must be started with the G or GB command prior to executing this command.
- If 1 parameter is specified, it is treated as the display cycle count since the last QD or QL command (0 at startup or when G or GB command is executed). (Example 1)
- If two parameters are specified, they are assumed to be absolute cycle specifications. (Example 2)
- Execution can be stopped with ^Z.
- QL Command Error Messages

Error Message	Description
Can't get real-time-trace data from PC4600	Data could not be obtained properly from PC4600.
checksum error	Data could not be obtained properly from PC4600.
invalid 1st parameter	The first parameter is invalid.
invalid 2nd parameter	The second parameter is invalid.
too many parameters	There are too many parameters.

QP (quest-point)

Access, set realtime trace point (batch format)

Command Syntax

- Set trace point address 1
  QP A1[=[pass count],[address][:mask],[data][:mask],[access condition]]

- Set trace point address 2
  QP A2[=[pass count],[address][:mask],[data][:mask],[access condition]]

- Set trace point address 3
  QP A3[=[pass count],[address][:mask]]

- Set trace point address 4
  QP A4[=[pass count],[address][:mask]]

- Set trace point address 5
  QP A5[=[pass count],[address][:mask]]

- Set trace point address 6
  QP A6[=[pass count],[address][:mask]]

- Set trigger trace point
  QP T[=[pass count],[H or L]]

- Set trace sequence
  QP C[=[condition no. (1-30)]]

- Set common pass count
  QP CP[=[pass count]]

Command Execution Example

```
>QP A1<RET>                                         (Ex 1)
A1 = 00001,  0000H:0000H, 00H:00H, FETCH
A1 Pass=2<RET>
A1 Address=SUB1<RET>
A1 Data=.<RET>                                      (Ex 2)
A1 = 00002, SUB1 E720H:0000H, 00H:00H, FETCH
>QP A2<RET>
A2 = 00001,  0000H:0000H, 00H:00H, FETCH
A2 Pass=3<RET>
A2 Address=SUB2<RET>
A2 Data=.<RET>
A2 = 00003, SUB2 E740H:0000H, 00H:00H, FETCH
>QP A1=<RET>                                        (Ex 3)
A1 = 00002, SUB1 E720H:0000H, 00H:00H, FETCH
>QP A2=<RET>
A2 = 00003, SUB2 E740H:0000H, 00H:00H, FETCH
>QP C<RET>                                          (Ex 4)
 1  A1*A1P                           15  ((A1+A2)*T)*CP
 2  A2*A2P                           16  ((A1+T)*A2)*CP
 3  T*TP                             17  (A1*A1P)->(A2*A2P)
 4  (A1*A1P)+(A2*A2P)                18  (A1*A1P)->(T*TP)
 5  (A1*A1P)+(T*TP)                  19  (T*TP)->(A1*A1P)
 6  (A1*A1P)+(A2*A2P)+(T*TP)         20  (A1->A2)*CP
 7  (A1*A1P)*(A2*A2P)                21  (A1->T)*CP
 8  (A1*A1P)*(T*TP)                  22  (T->A1)*CP
 9  (A1*A1P)*(A2*A2P)*(T*TP)         23  (A1*A1P)->(A2*A2P)->(T*TP)
10  (A1*A2)*CP                       24  (A1*A1P)->(T*TP)->(A2*A2P)
11  (A1*T)*CP                        25  (T*TP)->(A1*A1P)->(A2*A2P)
12  (A1*A2*T)*CP                     26  (A1->A2->T)*CP
13  ((A1*A2)+T)*CP                   27  (A1->T->A2)*CP
14  ((A1*T)+A2)*CP                   28  (T->A1->A2)*CP
29  (A1*A1P)+(A2*A2P)+(A3*A3P)+(A4*A4P)+(A5*A5P)+(A6*A6P)
30  (A1*A1P)+(A2*A2P)+(A3*A3P)+(A4*A4P)+(A5*A5P)+(A6*A6P)+(T*TP)
Condition  1. A1*A1P
C=20<RET>
C = 20. (A1->A2)*CP
>
```

Description

- The QP command sets and displays realtime trace points.
- Entering the realtime trace point only as "QP A1<RET>" enables trace conditions to be set interactively. (Example 1)
- A period indicates the end of interactive specification. (Example 2)
- Entering the realtime trace point and equal sign as "QP A1=<RET>" displays the currently set trace points. (Example 3)
- The "QP C" command can be used to specify the breakpoint passing sequence for breakpoints (A1, A2, T). (Example 4)
- The pass count for the entire combination can be set with the CP (Common Pass) command.
- Combination condition can be selected from the 30 conditions shown in Table 3.6.
- Mask data is enabled when 0 and masked when 1. Therefore, data match is not checked if FFH is specified as mask data.
- Table 3.6 shows the realtime trace sequence numbers.

Table 3.6 Realtime Trace Sequence Numbers

No.	Combination	No.	Combination
1	A1*A1P	15	((A1+A2)*T)*CP
2	A2*A2P	16	((A1+T)*A2)*CP
3	T*TP	17	(A1*A1P)->(A2*A2P)
4	(A1*A1P)+(A2*A2P)	18	(A1*A1P)->(T*TP)
5	(A1*A1P)+(T*TP)	19	(T*TP)->(A1*A1P)
6	(A1*A1P)*(A2*A2P)+(T*TP)	20	(A1->A2)*CP
7	(A1*A1P)*(A2*A2P)	21	(A1->T)*CP
8	(A1*A1P)*(T*TP)	22	(T->A1)*CP
9	(A1*A1P)*(A2*A2P)*(T*TP)	23	(A1*A1P)->(A2*A2P)->(T*TP)
10	(A1*A2)*CP	24	(A1*A1P)->(T*TP)->(A2*A2P)
11	(A1*T)*CP	25	(T*TP)->(A1*A1P)->(A2*A2P)
12	(A1*A2*T)*CP	26	(A1->A2->T)*CP
13	((A1*A2)+T)*CP	27	(A1->T->A2)*CP
14	((A1*T)+A2)*CP	28	(T->A1->A2)*CP
29	(A1*A1P)+(A2*A2P)+.....+(A5*A5P)+(A6*A6P)		
30	(A1*A1P)+.....+(A6*A6P)+(T*TP)		

- The symbols used in the condition are shown in Table 3.7.

Table 3.7 Symbols in Condition

Symbol	Description
A1	Address of trace point 1
A1P	Pass count of trace point 1
A2	Address of trace point 2
A2P	Pass count of trace point 2
A3-A6	Address of trace points 3-6
A3P-A6P	Pass count of trace point 3-6
T	Trigger trace point level
TP	Trigger trace point pass count
CP	Common pass count
->	Sequential condition
+	OR condition
*	AND condition

- QP Command Error Messages

Error Message	Description
Invalid input value. (xxx)	Input value for "xxx" is invalid.
too many parameters	Too many parameters are specified.

Command Syntax

QP

Command Execution Example

```
>QP<RET>
    :
    :    Go to menu mode
    :
********  Real Time Trace Point Table  ****************************
         Pass  Label     Address  (Mask)  Access H/L  Line [ Source     ]
  A1     00002 WORKS     0050H    (000FH) FETCH  --
  data   --                FFH    (  00H) --     --
  A2     00001           0000H    (0000H) FETCH  --
  data   --                00H    (  FFH) --     --
  A3     00001 SUB1      E720H    (0000H) --     --
  A4     00001 SUB3      E770H    (0000H) --     --
  A5     00001           0000H    (0000H) --     --
  A6     00001           0000H    (0000H) --     --
  T      00001           --       --      --     HIGH
  CP     00005
**********************************************************************
Condition  21. (A1->T)*CP
RTT Point><RET>                ← Press Return to return to the RTT74 command prompt.
>QP C=20<RET>                  ← Set the trace sequence condition.
C = 20. (A1->A2)*CP
    :
    :    Re-display menu
    :
********  Real Time Trace Point Table  ****************************
         Pass  Label     Address  (Mask)  Access H/L  Line [ Source     ]
  A1     00002 WORKS     0050H    (000FH) FETCH  --
  data   --                FFH    (  00H) --     --
  A2     00001           0000H    (0000H) FETCH  --
  data   --                00H    (  FFH) --     --
```

```
    A3     00001 SUB1      E720H   (0000H) --      --
    A4     00001 SUB3      E770H   (0000H) --      --
    A5     00001           0000H   (0000H) --      --
    A6     00001           0000H   (0000H) --      --
    T      00001           --      --      --      HIGH
    CP     00005
**********************************************************************
Condition  20. (A1->A2)*CP
RTT Point><RET>
>
```

Description

- The QP command enables trace points to be set in menu format. The menu displays the current trace points and allows them to be changed on screen. Use the cursor movement keys (↑ or ^E, ↓ or ^X, ←, → or ^D) to move the cursor to the value to be changed and enter the value.
- To terminate the QP command, enter "<RET>" at the "RTT Point>" prompt.
- The menu format is available only on personal computers that use ANSI escape sequences.

STOP (stop) Stop program execution

Command Syntax

STOP

Command Execution Example

```
>GB<RET>
Go>STOP<RET>

Terminate at E740H  SUB1
Trace Point Passed  Trace Mode = Trace About Trace-Point
Time = 0h 0m 16s  34m 860u sec
>STOP<RET>
MCU has already stopped.
>
```

Description

- The STOP command forces termination of the target MCU.
- STOP Command Error Message

Error Message	Description
MCU has already stopped.	The MCU is already stopped.

CHAPTER 4. USING BASIC DEBUGGING COMMANDS

4.1 Starting RTT74

This chapter assumes that PC4600 system setup has been completed and describes how to debug PC4600 system applications.

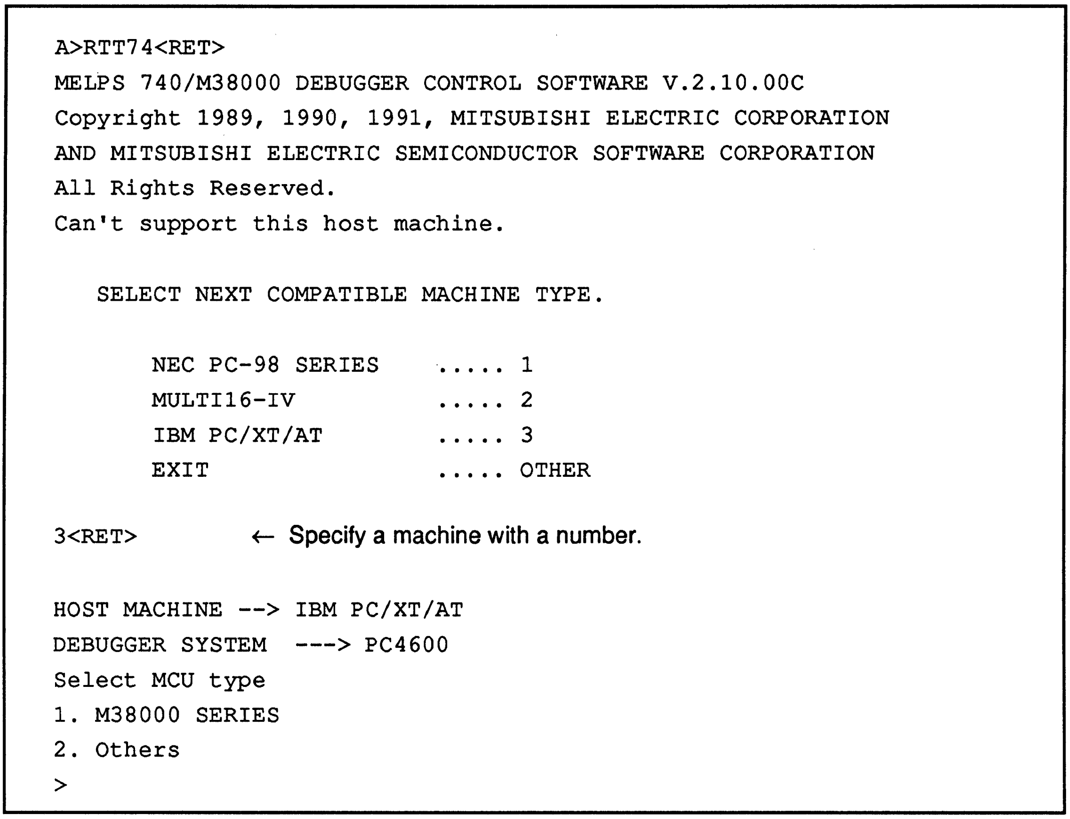
The following operations are described:

- Starting RTT74
- Using basic debugging commands

4.1.1 RTT74 Startup

RTT74 is started by entering "RTT74" at the MS-DOS command prompt and then pressing the Return key (or Enter key). The messages shown below appear when RTT74 starts. Messages shown in Figure 4.1 appear if a PC-9801 or IBM-PC is used. In this case, specify the host machine with a number.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
HOST MACHINE --> PC9801
DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>
```



```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
Can't support this host machine.

   SELECT NEXT COMPATIBLE MACHINE TYPE.

       NEC PC-98 SERIES    ..... 1
       MULTI16-IV          ..... 2
       IBM PC/XT/AT        ..... 3
       EXIT                ..... OTHER

3<RET>          ← Specify a machine with a number.

HOST MACHINE --> IBM PC/XT/AT
DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>
```

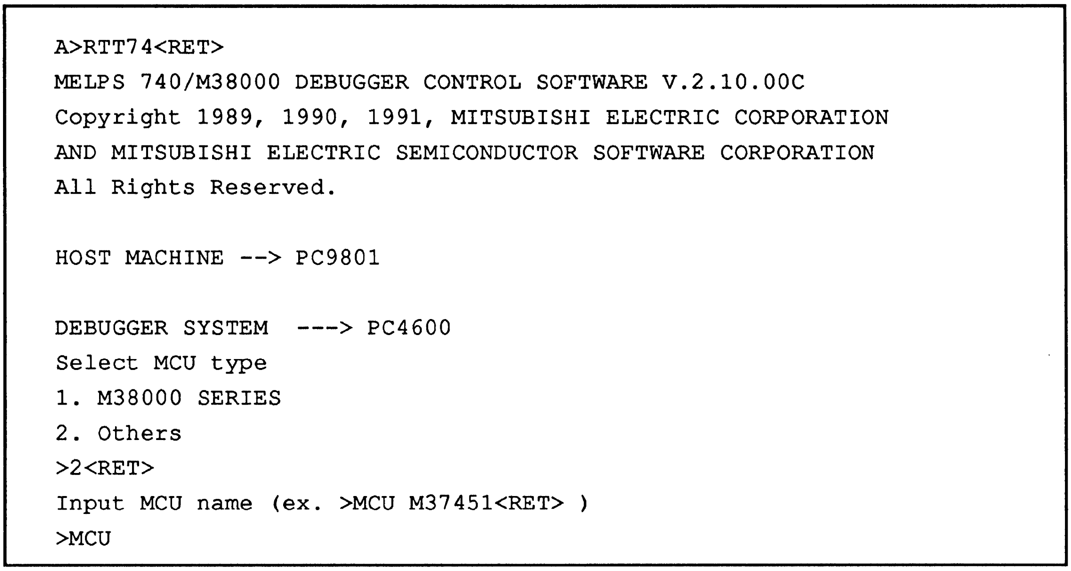
Figure 4.1 Startup Screen (Normal startup from compatible PC)

When the target MCU selection menu appears, specify the target MCU with a number.

1. M38000 Series

2. MELPS 740 Series (other than M38000 Series)

In this example, '2' is entered to specify MELPS 740 Series (M37471) as target. The messages shown Figure 4.2 appear after the number is entered.



```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>2<RET>
Input MCU name (ex. >MCU M37451<RET> )
>MCU
```

Figure 4.2 Specification After Startup (MELPS 740 Series)

After specifying the target MCU, a prompt ">MCU" appears asking for MCU command parameters (MCU specific information). (See Figure 4.2.)

Enter the MCU command parameter. MCU command parameter can be specified in two ways.

- By directly entering a number
  Enter the stack page selection bit no., MCU mode register address, stack end, and reset vector address in this order.

  Example (M37471)
  >MCU 2,FB,7F,FFFE<RET>

- With MCU name
  Enter the MCU name.

  Example (M37471)
  >MCU M37471<RET>

  In this case, the file "M37471.MCU" must exist in a directory accessible by RTT74.

The following figure shows an example of specifying the MCU name.

```
Select MCU type
1. M38000 SERIES
2. Others
>2<RET>
Input MCU name (ex. >MCU M37451<RET> )
>MCU M37471<RET>        ← MCU command execution using MCU name.
MONITOR            ---> V.1.00-B
MCU                ---> M37471

>
```

Now RTT74 is ready to accept debug commands.

The following figure shows the operation flow up to this point.

```
A>RTT74<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> PC9801

DEBUGGER SYSTEM  ---> PC4600
Select MCU type
1. M38000 SERIES
2. Others
>2<RET>
Input MCU name (ex. >MCU M37451<RET> )
>MCU M37471<RET>
MONITOR            ---> V.1.00-B
MCU                ---> M37471

>
```

4.1.2 Sample Programs

Figures 4.3 to 4.4 show the sample programs (M37471.EXT, ZRAM.A74, DISPDBG.A74, DISPSUB.A74) used in the subsequent descriptions. Figure 4.7 shows an assemble and link example.

```
        .ZEXT    RAM_TOP
        .ZEXT    PORT0, P0DIR, PORT1, P1DIR
```

Figure 4.3 M37471.EXT

```
    .SECTION    Z
    .ORG        0
:RAM_TOP:
:DISP_PTR:  .BLKB   10H
:DISP_DATA: .BLKB   10H
;
    .ORG        0C0H
:PORT0:     .BLKB   1
:P0DIR:     .BLKB   1
:PORT1:     .BLKB   1
:P1DIR:     .BLKB   1
;
:LSB_DISP_PTR   .EQU    0,DISP_PTR
;
    .END
```

Figure 4.4 ZRAM.A74

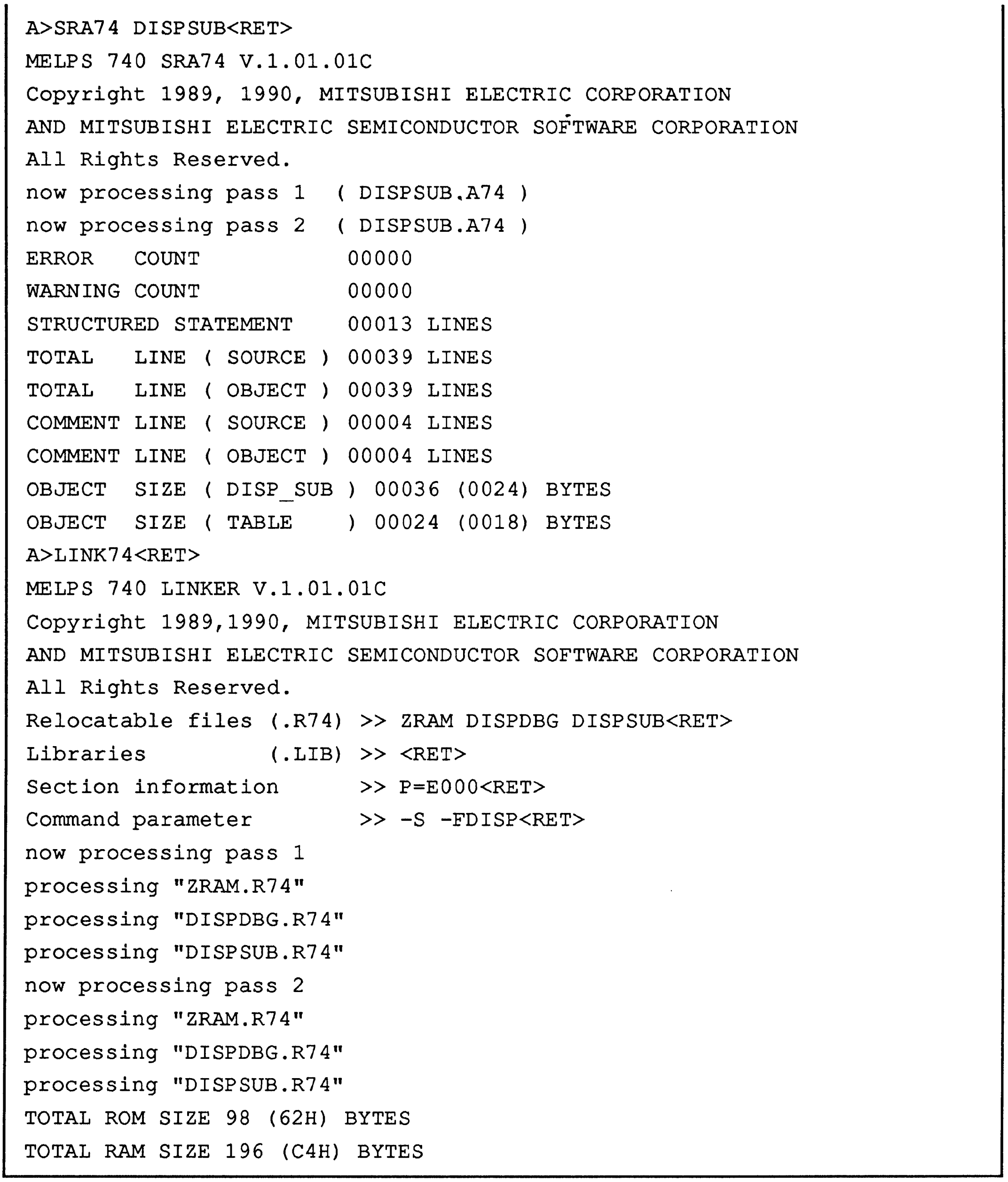
```
    .EXT          DISP_LED
    .INCLUDE      M37471.EXT
;
    .SECTION      P
    .FUNC         DEBUG_LED
:INIT:
    I = 1
    T = 0
    D = 0
    S = 07FH
    X = 0
    A = 0
    DO
        [RAM_TOP, X] = A
    WHILE  ++X <= 07FH
    [PORT0] = 0
    [P0DIR] = 0FFH
    [PORT1] = 0FFH
    [P1DIR] = 0FFH
:DEBUG_LED:
    FOR  EVER
        JSR       DISP_LED
    NEXT
    .ENDFUNC      DEBUG_LED
;
    .SECTION      VECTOR
    .ORG          0FFFEH
    .WORD         INIT
    .END
```

Figure 4.5 DISPDBG.A74

```
    .ZEXT       DISP_PTR, DISP_DATA
    .ZBEXT      LSB_DISP_PTR
    .INCLUDE    M37471.EXT
;
    .SECTION    DISP_SUB
    .ORG        0F000H
    .FUNC       DISP_SUB
:DISP_LED:
    A = [DISP_PTR]
    A = ++A
    Y = A & 07H
    [DISP_PTR] = Y
    X = A >> 1
    A = [DISP_DATA, X]
    IF  [LSB_DISP_PTR] == 1
        A = A >> 4
    ENDIF
    X = A & 07H
    [PORT1] = 0FFH
    [PORT0] = [SEG_DATA, X]
    [PORT1] = [DIGIT_DATA, Y]
    RTS
    .ENDFUNC    DISP_SUB
;
    .SECTION    TABLE
    .ORG        0C000H
:DIGIT_DATA:
    .BYTE   0FEH, 0FDH, 0FBH, 0F7H
    .BYTE   0EFH, 0DFH, 0BFH, 07FH
;
:SEG_DATA:
    .BYTE   0F3H, 060H, 0B6H, 0F4H
    .BYTE   066H, 0D6H, 0D7H, 072H
    .BYTE   0F7H, 076H, 07FH, 0C6H
    .BYTE   093H, 0E5H, 097H, 017H
;
    .END
```

Figure 4.6 DISPSUB.A74

```
A>SRA74 ZRAM<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
now processing pass 1  ( ZRAM.A74 )
now processing pass 2  ( ZRAM.A74 )
ERROR   COUNT           00000
WARNING COUNT           00000
STRUCTURED STATEMENT    00015 LINES
TOTAL   LINE ( SOURCE ) 00031 LINES
TOTAL   LINE ( OBJECT ) 00031 LINES
COMMENT LINE ( SOURCE ) 00002 LINES
COMMENT LINE ( OBJECT ) 00002 LINES
OBJECT  SIZE ( P      ) 00036 (0024) BYTES
OBJECT  SIZE ( VECTOR ) 00002 (0002) BYTES
A>SRA74 DISPDBG<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
now processing pass 1  ( DISPDBG.A74 )
now processing pass 2  ( DISPDBG.A74 )
ERROR   COUNT           00000
WARNING COUNT           00000
STRUCTURED STATEMENT    00015 LINES
TOTAL   LINE ( SOURCE ) 00031 LINES
TOTAL   LINE ( OBJECT ) 00031 LINES
COMMENT LINE ( SOURCE ) 00002 LINES
COMMENT LINE ( OBJECT ) 00002 LINES
OBJECT  SIZE ( P      ) 00036 (0024) BYTES
OBJECT  SIZE ( VECTOR ) 00002 (0002) BYTES
```



```
A>SRA74 DISPSUB<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
now processing pass 1  ( DISPSUB.A74 )
now processing pass 2  ( DISPSUB.A74 )
ERROR   COUNT           00000
WARNING COUNT           00000
STRUCTURED STATEMENT    00013 LINES
TOTAL   LINE ( SOURCE ) 00039 LINES
TOTAL   LINE ( OBJECT ) 00039 LINES
COMMENT LINE ( SOURCE ) 00004 LINES
COMMENT LINE ( OBJECT ) 00004 LINES
OBJECT  SIZE ( DISP_SUB ) 00036 (0024) BYTES
OBJECT  SIZE ( TABLE    ) 00024 (0018) BYTES
A>LINK74<RET>
MELPS 740 LINKER V.1.01.01C
Copyright 1989,1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.
Relocatable files (.R74) >> ZRAM DISPDBG DISPSUB<RET>
Libraries         (.LIB) >> <RET>
Section information      >> P=E000<RET>
Command parameter        >> -S -FDISP<RET>
now processing pass 1
processing "ZRAM.R74"
processing "DISPDBG.R74"
processing "DISPSUB.R74"
now processing pass 2
processing "ZRAM.R74"
processing "DISPDBG.R74"
processing "DISPSUB.R74"
TOTAL ROM SIZE 98 (62H) BYTES
TOTAL RAM SIZE 196 (C4H) BYTES
```

Figure 4.7 Assemble and Link Example

4.2 Using Debugging Commands

The target systems listed below are used in the following examples:

- M37471RSS (Emulator Dedicated MCU)

- M37471T-ADS (Pin Processing Board)

4.2.1 MCU Memory Space Type Attributes

The type attribute of MCU memory space is set with the MAP command. The default setting after turning on the PC4600 power is as shown below. Enter "MAP<RET" to view the mapping information.

```
>MAP<RET>
Memory Area     RAM/ROM EXT/INT
0000  -  1FFF   RAM     EXT
2000  -  FFFF   ROM     INT
>
```

Then disable access of areas 0100H-DFFFH as follows:

```
>MAP 0100,DFFF,DIS,INT<RET>
>MAP<RET>
Memory Area     RAM/ROM EXT/INT
0000  -  00FF   RAM     EXT
0100  -  DFFF   DIS     INT
E000  -  FFFF   ROM     INT
>
```

These attributes are used in protect breaks in the break condition.

4.2.2 Loading the Sample Program

Use the “I” command to load the necessary information (symbol information and source line debug information) from the symbol file.

```
>I DISP<RET>
GLOBAL SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
DISP.HEX DISP.SYM LOAD END --> 13 SYMBOLS DEFINED
>
```

Information is loaded in the following sequence.

1. Global symbol information
   Loaded in the sequence of description in the symbol file. Symbols loaded up to the point where the memory runs out are valid.

2. Local symbol information
   Loaded in the sequence of description in the symbol file. Symbols loaded up to the point where the memory runs out are valid.

3. Source line debug information
   Source line debug information become invalid if memory runs out.

4.2.3 Sample Program Execution

The sample program loaded in the previous section is executed as follows:

1. Reset the target MCU with the Z command.
2. Enter the G command to execute the sample program.
3. Enter the STOP command to stop the sample program.

RTT74 provides two command input modes depending on the status of the target MCU. When the target MCU is stopped, it is referred to as the normal command mode. In this mode, enter the command following the prompt '>'. When the target MCU is running, it is referred to as the runtime command mode. In this mode, enter the command following the prompt 'Go>'. There are some restrictions on the type of commands that can be used in the runtime command mode. Table 4.1 shows a list of commands that can be used in runtime command mode.

Table 4.1 Commands that can be used in Runtime Command Mode

Operation	Command	Restrictions
Memory access	A, D, F, L, M, S	None
Execution control	X	Reference only
	STOP	None
Realtime trace	QD, QL	BEFORE mode display from the point where the command is executed.
Software analysis	CV	None
Symbol operation	SCOPE, SI, SHOW SOURCE	None
Utility	?, !, ;	None

4.2.4 Breakpoints

The PC4600 system has the following breakpoints:

- A1, A2
  Breaks when the pass count to the specified address from program execution and data and access conditions are satisfied.

- A3, A4, A5, A6
  Breaks when the pass count to the specified address from program execution is satisfied.

- T
  Breaks when the trigger (rising/falling edge) input from an external trace cable and pass count are satisfied.

- PROTECT
  Breaks when access protected area is accessed or when an attempt is made to write to a read only area.

The PC4600 system allows breakpoints A1-A6, and T to be set as 30 different combinations shown in Table 4.2.

Table 4.2 Break Sequence Numbers

No.	Combination	No.	Combination
1	A1*A1P	15	((A1+A2)*T)*CP
2	A2*A2P	16	((A1+T)*A2)*CP
3	T*TP	17	(A1*A1P)->(A2*A2P)
4	(A1*A1P)+(A2*A2P)	18	(A1*A1P)->(T*TP)
5	(A1*A1P)+(T*TP)	19	(T*TP)->(A1*A1P)
6	(A1*A1P)*(A2*A2P)+(T*TP)	20	(A1->A2)*CP
7	(A1*A1P)*(A2*A2P)	21	(A1->T)*CP
8	(A1*A1P)*(T*TP)	22	(T->A1)*CP
9	(A1*A1P)*(A2*A2P)*(T*TP)	23	(A1*A1P)->(A2*A2P)->(T*TP)
10	(A1*A2)*CP	24	(A1*A1P)->(T*TP)->(A2*A2P)
11	(A1*T)*CP	25	(T*TP)->(A1*A1P)->(A2*A2P)
12	(A1*A2*T)*CP	26	(A1->A2->T)*CP
13	((A1*A2)+T)*CP	27	(A1->T->A2)*CP
14	((A1*T)+A2)*CP	28	(T->A1->A2)*CP
29	(A1*A1P)+(A2*A2P)+....+(A5*A5P)+(A6*A6P)		
30	(A1*A1P)+....+(A6*A6P)+(T*TP)		

4.2.5 Breakpoint Setting Examples

The following example shows how to set a breakpoint that stops the program after writing the data FxH (x is any number) 333 times in RAM area PORT1 (address: C2H).

The BP command is used to set the breakpoint. The menu format of the BP command is used in this example. With the menu format, data is set by moving the cursor to the desired location and pressing the Return or Enter key.

1. The menu shown in Figure 4.8 appears when the command "BP<RET>" is entered at the prompt ">".

2. Move the cursor to Pass for A1 with the cursor movement key and enter "333". (Pass count is specified in decimals.)

3. Move the cursor to Address and enter "PORT1:0".

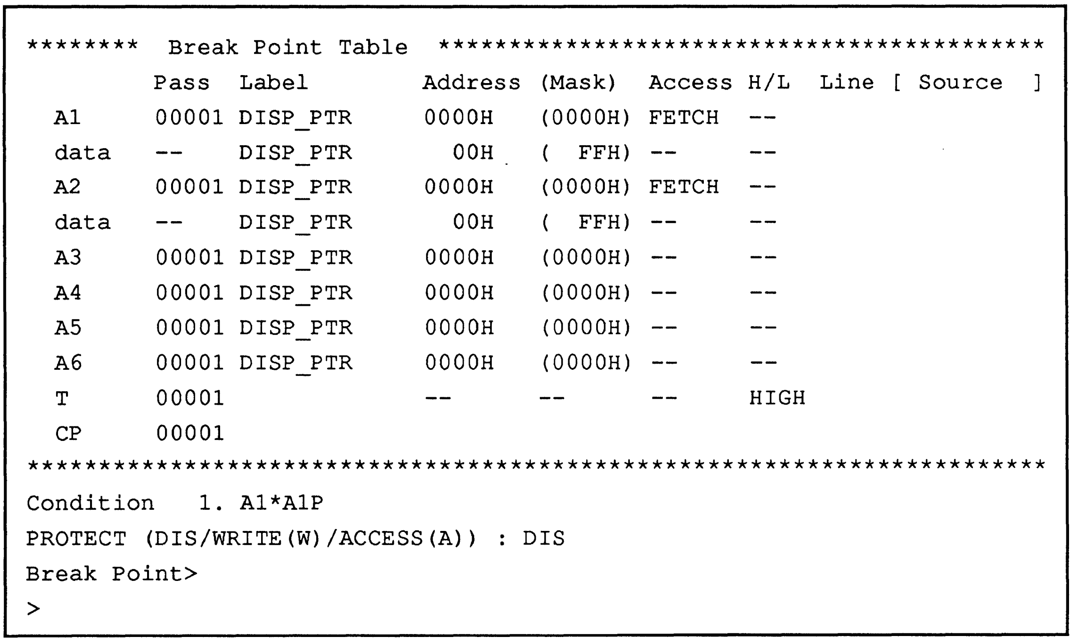
4. Move the cursor to Data and enter "FF:0F". Mask is disabled when 0 and enabled when 1. In this case, the lower 4 bits of the data are not compared because 0FH is specified. In other words, the target data is F0H-FFH.

5. Move the cursor to Access and enter "W" or "WRITE".

6. Move the cursor to Condition and enter "1".

7. Move the cursor to PROTECT and enter "DIS".

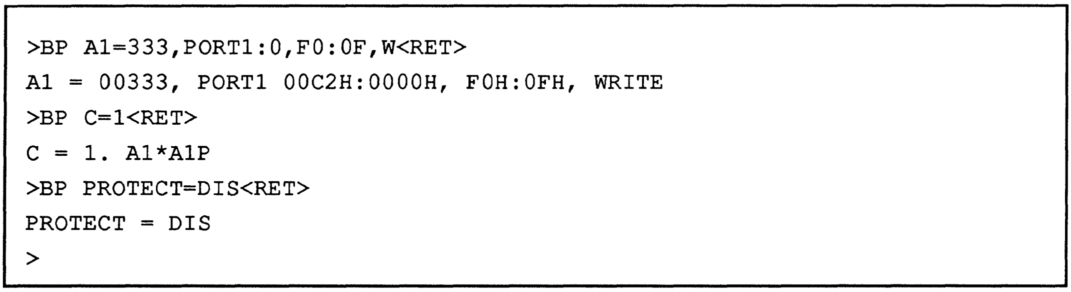
8. Enter "<RET>" to return to the ">" prompt.



```
********  Break Point Table  ******************************************
         Pass  Label        Address  (Mask)  Access H/L  Line [ Source  ]
  A1     00001 DISP_PTR     0000H    (0000H) FETCH  --
  data   --    DISP_PTR       00H    (  FFH) --     --
  A2     00001 DISP_PTR     0000H    (0000H) FETCH  --
  data   --    DISP_PTR       00H    (  FFH) --     --
  A3     00001 DISP_PTR     0000H    (0000H) --     --
  A4     00001 DISP_PTR     0000H    (0000H) --     --
  A5     00001 DISP_PTR     0000H    (0000H) --     --
  A6     00001 DISP_PTR     0000H    (0000H) --     --
  T      00001              --       --      --     HIGH
  CP     00001
***********************************************************************
Condition    1. A1*A1P
PROTECT (DIS/WRITE(W)/ACCESS(A)) : DIS
Break Point>
>
```

Figure 4.8 BP Command Menu Mode

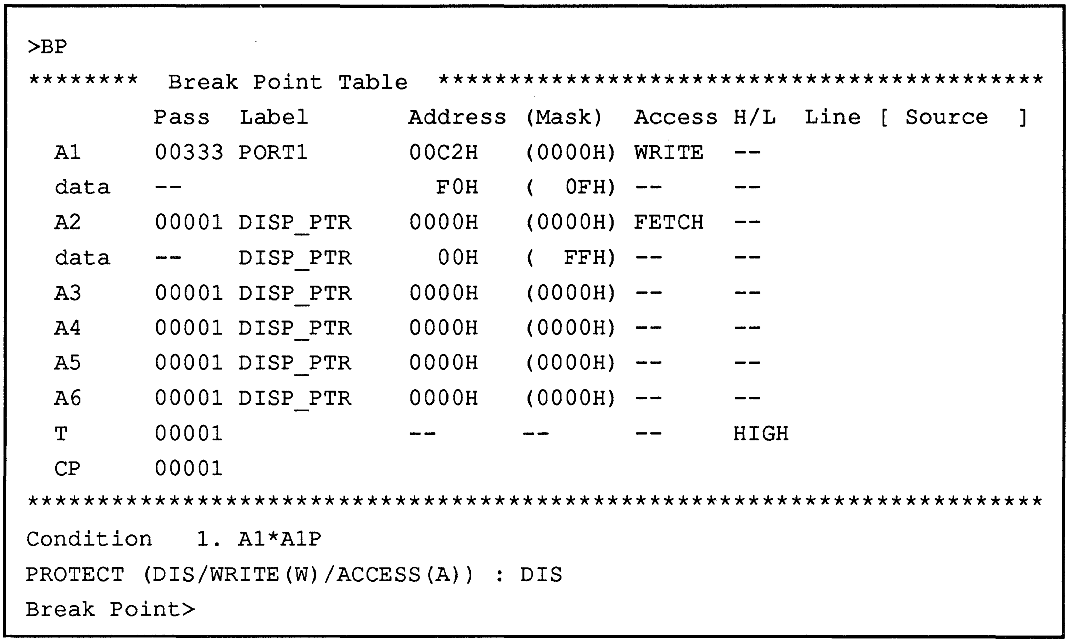
The following figure shows a batch mode input example.



```
>BP A1=333,PORT1:0,F0:0F,W<RET>
A1 = 00333, PORT1 00C2H:0000H, F0H:0FH, WRITE
>BP C=1<RET>
C = 1. A1*A1P
>BP PROTECT=DIS<RET>
PROTECT = DIS
>
```

This completes the setting of breakpoints.

The following figure shows the settings displayed with the menu mode BP command.



```
>BP
********  Break Point Table  ******************************************
         Pass  Label       Address (Mask)  Access H/L  Line [ Source  ]
  A1     00333 PORT1       00C2H   (0000H) WRITE  --
  data   --                  F0H   (  0FH) --     --
  A2     00001 DISP_PTR    0000H   (0000H) FETCH  --
  data   --    DISP_PTR      00H   (  FFH) --     --
  A3     00001 DISP_PTR    0000H   (0000H) --     --
  A4     00001 DISP_PTR    0000H   (0000H) --     --
  A5     00001 DISP_PTR    0000H   (0000H) --     --
  A6     00001 DISP_PTR    0000H   (0000H) --     --
  T      00001             --      --      --     HIGH
  CP     00001
**********************************************************************
Condition   1. A1*A1P
PROTECT (DIS/WRITE(W)/ACCESS(A)) : DIS
Break Point>
```

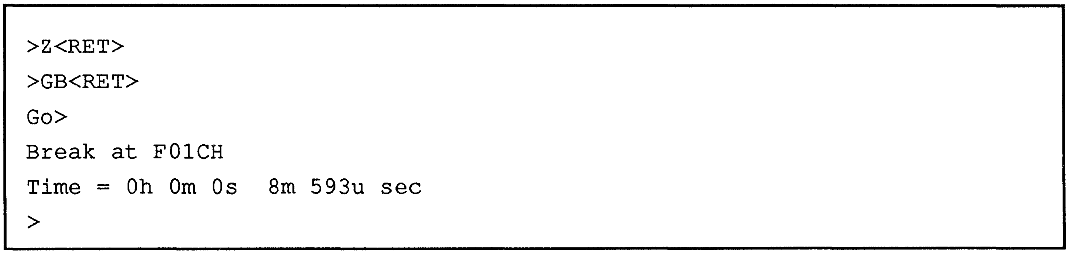
4.2.6 Executing Program with Breakpoints

The following example shows how to execute the program and stop it at the breakpoints set in the previous section. Figure 4.9 shows the corresponding screen.

1. Reset the target MCU with the Z command.

2. Enter the GB command to execute the sample program. Runtime command mode is entered when the program starts execution and the prompt changes to "Go>".

3. The program stops when the break condition is satisfied. The break address, trace status, and execution time are displayed.



```
>Z<RET>
>GB<RET>
Go>
Break at F01CH
Time = 0h 0m 0s  8m 593u sec
>
```

Figure 4.9 Program Execution with Breakpoint

4.2.7 PROTECT Break

A PROTECT break causes a break when an area disabled (DIS) for access with the MAP command is accessed (read or write) or when an attempt is made to write to a read only area (ROM).

The MAP command is used to change the attribute of the memory area and allow the disabled area to be accessed. The following procedure describes how a PROTECT break is generated. Figure 4.10 shows the corresponding execution example.

1. Disable access to memory area C000H to CFFFH with the MAP command.

2. Use the BP command set a breakpoint at a point that will not be executed.

3. Use the BP command to set PROTECT to "ACCESS".

4. Reset the MCU with the Z command.

5. Execute the program with the GB command.

6. A break occurs because an attempt is made to read from area C000H-CFFFH.

```
>MAP 100,DFFF,DIS,INT<RET>
>MAP<RET>
Memory Area    RAM/ROM EXT/INT
0000  -  00FF  RAM     INT
0100  -  DFFF  DIS     INT
E000  -  FFFF  ROM     INT
>BP PROTECT=ACCESS<RET>
PROTECT = ACCESS
>Z<RET>
>GB<RET>
Go>
Break at F01CH
Time = 0h 0m 0s  0m 951u sec
>
```

Figure 4.10 Protect Break Execution Example

4.2.8 Realtime Trace

The PC4600 system provides a realtime trace function which stores execution results in trace memory (8192 steps) without affecting program execution time when a trace condition is satisfied.

The PC4600 system allows specification of a trace area synchronized with a breakpoint and four different trace areas shown in Figure 4.11 synchronized with trace points.

The following four trace areas can be selected:

1. Trace Before Trace-Point
   Stores information prior to the trace point (set with the QP command).

2. Trace About Trace-Point
   Stores information around the trace point.

3. Trace After Trace-Point
   Stores information after the trace point.

4. Trace Before Break-Point
   Stores information prior to the breakpoint (set with the BP command).

4.2.9 Trace Points

The PC4600 system allows trace points equivalent to breakpoints A1-A6, and T to be set.

The PC4600 system allows trace points A1-A6, and T to be set as 30 different combinations shown in Table 4.3.

Table 4.3 Trace Sequence Combination

No.	Combination	No.	Combination
1	A1*A1P	15	((A1+A2)*T)*CP
2	A2*A2P	16	((A1+T)*A2)*CP
3	T*TP	17	(A1*A1P)->(A2*A2P)
4	(A1*A1P)+(A2*A2P)	18	(A1*A1P)->(T*TP)
5	(A1*A1P)+(T*TP)	19	(T*TP)->(A1*A1P)
6	(A1*A1P)*(A2*A2P)+(T*TP)	20	(A1->A2)*CP
7	(A1*A1P)*(A2*A2P)	21	(A1->T)*CP
8	(A1*A1P)*(T*TP)	22	(T->A1)*CP
9	(A1*A1P)*(A2*A2P)*(T*TP)	23	(A1*A1P)->(A2*A2P)->(T*TP)
10	(A1*A2)*CP	24	(A1*A1P)->(T*TP)->(A2*A2P)
11	(A1*T)*CP	25	(T*TP)->(A1*A1P)->(A2*A2P)
12	(A1*A2*T)*CP	26	(A1->A2->T)*CP
13	((A1*A2)+T)*CP	27	(A1->T->A2)*CP
14	((A1*T)+A2)*CP	28	(T->A1->A2)*CP
29	(A1*A1P)+(A2*A2P)+....+(A5*A5P)+(A6*A6P)		
30	(A1*A1P)+....+(A6*A6P)+(T*TP)		

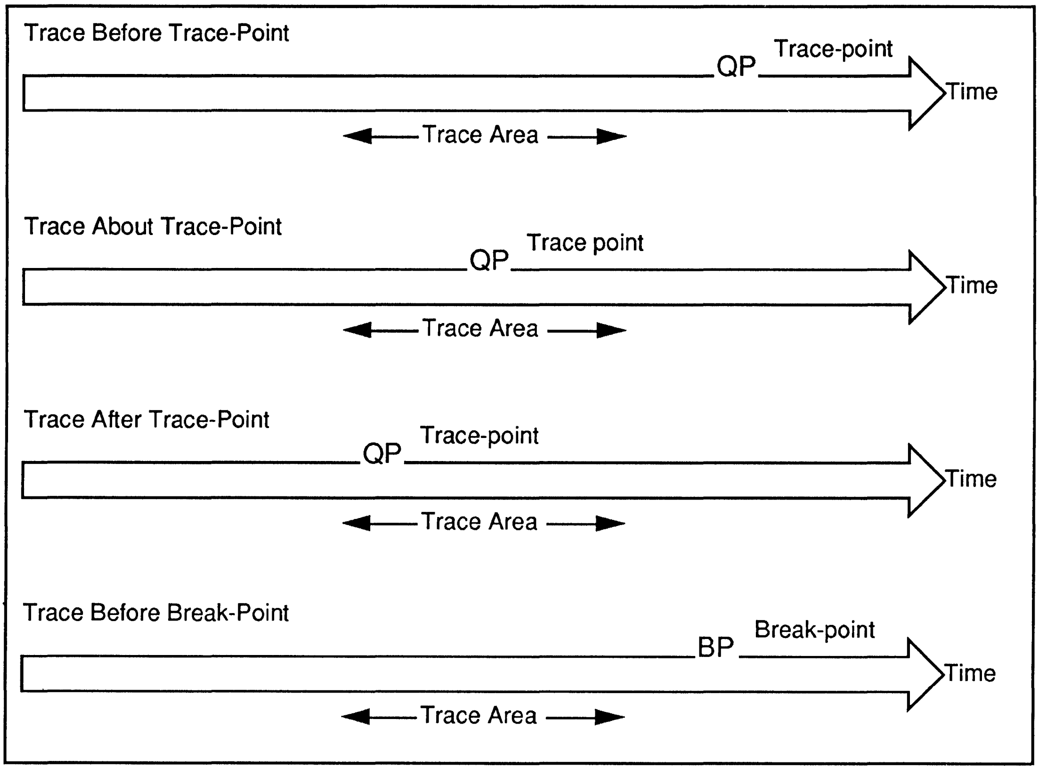


Figure 4.11 Trace Area

4.2.10 Trace Point Setting Examples

This section describes how to set a trace point when data FEH is read from DIGIT_DATA (address: C000H) in ROM area 10 times and trace program execution for 2000H (8192) steps about the trace point. The trace point is set as follows:

The trace area is set with the QC command.

The QP (RP) command is used to set the trace point. The menu format of the QP command is used in the following example. The menu format allows data to be set by moving the cursor to the desired location, entering the data, and pressing the Return or Enter key.

1. The menu shown in Figure 4.12 appears when the "QC<RET>" command is entered at the prompt ">".

2. Enter "2" (Trace About Trace-Point) at this prompt.

3. The menu shown in Figure 4.13 appears when the "QP<RET>" command is entered at the prompt ">".

4. Use the cursor keys to move the cursor to Pass for A1 and then enter "10". (Pass count is specified in decimals.)

5. Move the cursor to Address and enter "DIGIT_DATA:0".

6. Move the cursor to Data and enter "FE:0".

7. Move the cursor to Access and enter "W" or "WRITE".

8. Move the cursor to Condition and enter "1".

9. Enter "<RET>" to return to the ">" prompt.

Now the trace point is set.

Figure 4.14 shows the sample program being executed with the G command to obtain trace data.

When the program is stopped with the STOP command, the reached trace points are displayed together with the trace mode.

Then the trace results obtained in the previous section are displayed. The trace result can be displayed with either of the following two commands:

1. QD (RD) command
   Displays the contents of trace memory in hexadecimal as the status of external signals and external trace cable.

2. QL (RL) command
   Lists the contents of the trace memory as a disassembled list

```
>QC<RET>

1. Trace Before Trace-Point
2. Trace About Trace-Point
3. Trace After Trace-Point
4. Trace Before Break-Point

Select No.>2<RET>
2. Trace About Trace-Point Selected

>
```

Figure 4.12 QC Command Execution Example

```
>QP
********  Real Time Trace Point Table  *********************************
         Pass  Label        Address (Mask)  Access H/L  Line [ Source   ]
  A1     00001 DISP_PTR     0000H   (0000H) FETCH  --
  data   --    DISP_PTR       00H   (  FFH) --     --
  A2     00001 DISP_PTR     0000H   (0000H) FETCH  --
  data   --    DISP_PTR       00H   (  FFH) --     --
  A3     00001 DISP_PTR     0000H   (0000H) --     --
  A4     00001 DISP_PTR     0000H   (0000H) --     --
  A5     00001 DISP_PTR     0000H   (0000H) --     --
  A6     00001 DISP_PTR     0000H   (0000H) --     --
  T      00001              --      --      --     HIGH
  CP     00001
************************************************************************
Condition   1. A1*A1P
RTT Point>
>
```

Figure 4.13 QP Command Execution Example

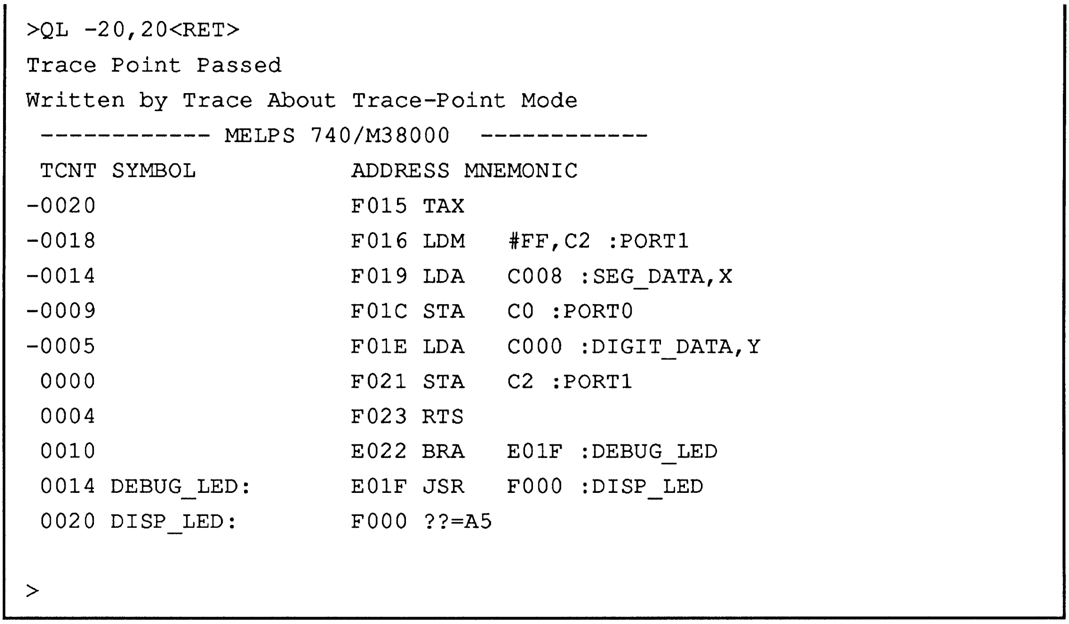
The following figure shows a batch mode input example.

```
>QP A1=10,DIGIT_DATA:0,FE:0,R<RET>
A1 = 00010, DIGIT_DATA C000H:0000H, FEH:00H, READ
>QP C=1<RET>
C = 1. A1*A1P
>
```

The following figure shows the a list of settings displayed with the QP command.

```
>QP<RET>
********  Real Time Trace Point Table  *********************************
         Pass  Label         Address (Mask)  Access H/L  Line [ Source   ]
  A1     00010 DIGIT_DATA    C000H   (0000H) READ   --
  data   --                    FEH   (  00H) --     --
  A2     00001 DISP_PTR      0000H   (0000H) FETCH  --
  data   --    DISP_PTR        00H   (  FFH) --     --
  A3     00001 DISP_PTR      0000H   (0000H) --     --
  A4     00001 DISP_PTR      0000H   (0000H) --     --
  A5     00001 DISP_PTR      0000H   (0000H) --     --
  A6     00001 DISP_PTR      0000H   (0000H) --     --
  T      00001               --      --      --     HIGH
  CP     00001
************************************************************************
Condition   1. A1*A1P
RTT Point>
```

```
>Z<RET>
>G<RET>
Go>STOP<RET>
Terminate at F015H
Trace Point Passed  Trace Mode = Trace About Trace-Point
Time = 0h 0m 0s  5m 647u sec
>QD -10,10<RET>
Trace Point Passed
Written by Trace About Trace-Point Mode
 -------------------- MELPS 740/M38000 --------------------  - EXT --
 TCNT SYMBOL          Address  Data  Sync Read Write B-T Q-T 12345678
-0010 SEG_DATA:       C008     F3    0    0    1     1   1   11111111
-0009                 F01C     85    1    0    1     1   1   11111111
-0008                 F01D     C0    0    0    1     1   1   11111111
-0007 PORT0:          00C0     F3    0    1    1     1   1   11111111
-0006 PORT0:          00C0     F3    0    1    0     1   1   11111111
-0005                 F01E     B9    1    0    1     1   1   11111111
-0004                 F01F     00    0    0    1     1   1   11111111
-0003                 F020     C0    0    0    1     1   1   11111111
-0002 DIGIT_DATA:     C000     FE    0    1    1     1   1   11111111
-0001 DIGIT_DATA:     C000     FE    0    0    1     1   1   11111111
 0000                 F021     85    1    0    1     1   1   11111111
 0001                 F022     C2    0    0    1     1   1   11111111
 0002 PORT1:          00C2     C2    0    1    1     1   1   11111111
 0003 PORT1:          00C2     FE    0    1    0     1   1   11111111
 0004                 F023     60    1    0    1     1   1   11111111
 0005                 F024     EA    0    0    1     1   1   11111111
 0006                 007D     EA    0    1    1     1   1   11111111
 0007                 007E     21    0    0    1     1   1   11111111
 0008                 007F     E0    0    0    1     1   1   11111111
 0009                 E021     F0    0    1    1     1   1   11111111
 0010                 E022     80    1    0    1     1   1   11111111
```



```
>QL -20,20<RET>
Trace Point Passed
Written by Trace About Trace-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL          ADDRESS MNEMONIC
-0020                 F015 TAX
-0018                 F016 LDM   #FF,C2 :PORT1
-0014                 F019 LDA   C008 :SEG_DATA,X
-0009                 F01C STA   C0 :PORT0
-0005                 F01E LDA   C000 :DIGIT_DATA,Y
 0000                 F021 STA   C2 :PORT1
 0004                 F023 RTS
 0010                 E022 BRA   E01F :DEBUG_LED
 0014 DEBUG_LED:      E01F JSR   F000 :DISP_LED
 0020 DISP_LED:       F000 ??=A5

>
```

Figure 4.14 Realtime Trace Data Display Example

4.2.11 Coverage Analysis Function

The coverage analysis function checks whether the memories between two specified points (addresses) are accessed or not. The access status of each address is expressed as 1 bit data in a 64K bit coverage memory within the PC4600 system.

The following is an example of performing coverage analysis of a sample program. The target addresses are 00H to FFH in RAM. The program is executed from DISP_LED (address: E01FH) because the sample program zero clears the area between 00H and 7FH. The coverage analysis procedure is as follows:

1. Set breakpoints and trace points to addresses that will not be executed.
2. Execute the program once to initialize the program.
3. Enter "CV CLEAR<RET>" at the prompt ">" (normal command mode) to initialize the coverage memory.
4. Execute the sample program from DEBUG_LED with the G command.
5. The coverage analysis data is displayed as "CV TOTAL", "CV GLOBAL", and "CV LOCAL".

The entire execution flow is shown in the following figure.

```
>I DISP<RET>
GLOBAL SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
DISP.HEX DISP.SYM LOAD END --> 13 SYMBOLS DEFINED
>BP A2=,0:0,0:0,F<RET>
A2 = 00001, DISP_PTR 0000H:0000H, 00H:00H, FETCH
>BP C=2<RET>
C = 2. A2*A2P
>BP PROTECT=DIS<RET>
PROTECT = DIS
>QP A2=,0:0,0:0,F<RET>
A2 = 00001, DISP_PTR 0000H:0000H, 00H:00H, FETCH
>QP C=2<RET>
C = 2. A2*A2P
>Z<RET>
>G<RET>

Go>STOP<RET>
Terminate at F021H
Time = 0h 0m 8s  305m 207u sec
>CV CLEAR<RET>
>Z<RET>
>G DEBUG_LED<RET>

Go>STOP<RET>
Terminate at F01EH
Time = 0h 0m 12s  854m 913u sec
>CV TOTAL 0,FF<RET>
Coverage Percent (0000H-00FFH) = 3.51 %
>CV GLOBAL<RET>
ADDRESS>0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
 0000   8000F00000000000000000000000000300000000000000000A000000000000000
>CV LOCAL<RET>
ADDRESS>0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
 0000     *               ****
 0040                                                                 **
 0080
 00C0     * *
>
```

(a) CV TOTAL

CV TOTAL shows the percentage of the addresses in the specified area that have been accessed. In this example, 3.51% of the addresses between 00H and 0FFH have been accessed.

(b) CV GLOBAL

CV GLOBAL shows the coverage analysis result in 4 byte units with a hexadecimal value between 0 and F. In this example, the access status of addresses 00H to 03H is 8. Therefore, address 00H is accessed and addresses 01H, 02H, and 03H are not accessed.

Address	00H	01H	02H	03H
8H=1000B	1	0	0	0
Access	Yes	No	No	No

(c) CV LOCAL

CV LOCAL shows the coverage analysis result in 1 byte units indicating accessed address with an asterisk and unaccessed address with a blank. In this example, addresses 80H to BFH are not accessed.

4.3 Runtime Debugging

The PC4600 system enables registers, memories, and realtime trace memory to be accessed during program execution without stopping the MCU.

The following is an example of runtime debugging.

Note that you may not be able to produce the same result because the execution result depends on the timing of the command.

- Set the trace mode to "4. Trace Before Break-Point"
- The runtime debugging mode prompt is "Go>".

```
>I DISP<RET>
GLOBAL SYMBOL LOADED
SOURCE LINE DEBUG INFORMATION LOADED
DISP.HEX DISP.SYM LOAD END --> 13 SYMBOLS DEFINED
>QP A1=2,E000:0,00:FF,F<RET>
A1 = 00002, INIT E000H:0000H, 00H:FFH, FETCH  00006[DISPDBG.A74 ]
>QP C=1<RET>
C = 1. A1*A1P
>QC 4<RET>
4. Trace Before Break-Point Selected

>Z<RET>
>G<RET>

Go>
```

- The realtime trace data can be referenced while the MCU is executing (with QD or QL command).

  The following information is displayed.

  - Trace point passage
  - Write mode (Before, About, After)

The following example shows that the trace point has not been reached and the data up to the point of QL command execution has been written in Before mode.

```
>Z<RET>
>G<RET>

Go>QL -20,0<RET>
Trace Point Unpassed
Written by Before Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL           ADDRESS MNEMONIC
-0019                  F006 STY   00 :DISP_PTR
-0015                  F008 LSR   A
-0013                  F009 TAX
-0011                  F00A LDA   10 :DISP_DATA,X
-0007                  F00C BBC   0,00 :LSB_DISP_PTR,F013
 0000                  F013 ??=29

Go>
```

- The contents of registers can be referenced (with X command).
- The contents of memories can be changed (with A, D, L, or S command).

```
Go>X<RET>
A=00  X=00  Y=07  F=----B-IZ-  S=07D  PC=F005
      TAY
Go>S PORT1<RET>
PORT1:           00C2 FF .<RET>
Go>D 0,F<RET>
0000 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
Go>
```

- The result will be different if the same command is entered again because the MCU is executing.

```
Go>X<RET>
A=04  X=00  Y=04  F=---B-I--  S=07D  PC=F006
      STY   00 :DISP_PTR

Go>S PORT1<RET>
PORT1:           00C2 BF .<RET>
Go>D 0,F<RET>
0000 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
Go>STOP<RET>
Terminate at E033H
Trace Point Passed  Trace Mode = Trace Before Break-Point
Time = 0h 0m 8s  628m 948u sec
>QL -20,0<RET>
Trace Point Passed
Written by Trace Before Break-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL           ADDRESS MNEMONIC
-0020                  F013 AND   #07
-0018                  F015 TAX
-0016                  F016 LDM   #FF,C2 :PORT1
-0012                  F019 LDA   C008 :SEG_DATA,X
-0007                  F01C STA   C0 :PORT0
-0003                  F01E LDA   C000 :DIGIT_DATA,Y

>
```

- The following example shows a case where a trace point is reached during program execution and trace is ended at that point.

- The trace mode is set to "2. Trace About Trace-Point".

```
>QP A1=200,PORT1:0,F0:0F,W<RET>
A1 = 00200, PORT1 00C2H:0000H, F0H:0FH, WRITE
>QC 2<RET>
2. Trace About Trace-Point Selected

>Z<RET>
>G<RET>

Go>QL -8,8<RET>
Trace Point Passed
Written by Trace About Trace-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL          ADDRESS MNEMONIC
-0008                 F013 AND   #07
-0006                 F015 TAX
-0004                 F016 LDM   #FF,C2 :PORT1
 0000                 F019 LDA   C008 :SEG_DATA,X
 0005                 F01C STA   C0 :PORT0

Go>STOP<RET>
Terminate at F012H
Trace Point Passed  Trace Mode = Trace About Trace-Point
Time = 0h 0m 0s  7m 446u sec
>QL -8,8<RET>
Trace Point Passed
Written by Trace About Trace-Point Mode
 ------------ MELPS 740/M38000  ------------
 TCNT SYMBOL          ADDRESS MNEMONIC
-0008                 F013 AND   #07
-0006                 F015 TAX
-0004                 F016 LDM   #FF,C2 :PORT1
 0000                 F019 LDA   C008 :SEG_DATA,X
 0005                 F01C STA   C0 :PORT0

>
```

APPENDIX A. ERROR MESSAGES

The error messages that appear when using the option board system commands and error messages pertaining to all commands are described below.

The error messages pertaining to PC4600 system commands are described under each command.

An error message appears when command execution is unsuccessful. Table A.1 lists the error messages.

Table A.1 Error Messages

Error Message	Description
?	An invalid character or command is entered. ⇒ Check the command format and enter the correct command.
ADDRESS ERROR	The contents of the program counter is abnormal. ⇒ Check that the value of the reset vector is correct and reset the target MCU.
BANK DATA CANNOT RECEIVE	Data set in bank cannot be received correctly when the reset (Z) command is executed. ⇒ Check that the serial cable is connected properly and then execute the reset command.
BANK SELECT ERROR	The bank selection switch on the option board is not set properly. ⇒ Properly set the bank selection switch.
Can't close [xxx]	File xxx cannot be closed. ⇒ Check for diskette errors.
Can't get source file (including program counter)	File containing the program counter cannot be found. ⇒ Check the current program counter.
Can't open source file [xxx]	Cannot find source file xxx or it cannot be opened. ⇒ Check that the file is in the proper location.
Can't open symbol file [xxx.sym]	Cannot find SYM file xxx.sym or it cannot be opened. ⇒ Check that the file is in the proper location.
Can't read source file [xxx]	Failed to obtain information from source file xxx. ⇒ Check for diskette errors.
? Can't not refer to -	An area inaccessible with the program counter has been accessed in M50734 mode. ⇒ Change the value of the program counter with the XP command to an accessible value.
Can't close data file.	The data file cannot be closed. ⇒ Check the diskette.

Error Message	Description
Can't find data file.	The data file does not exist. ⇒ Copy the DAT file to an accessible directory or use the -I option.
Can't find data in data file.	Data cannot be found in the data file. ⇒ Contact MSC if this message appears.
Can't read valid data in data file.	There is an error in data file information. ⇒ Contact MSC if this message appears.
CHECKSUM ERROR	There is an error in the data when option board trace memory is transmitted to the host machine. ⇒ Check the serial cable connection and re-execute the command.
MCU NUMBER ERROR	Option board other not supporting MELPS 740 or RTT74 is used. ⇒ Check the DAT file version to determine whether the MCU information is stored in the DAT file.
? FILE NAME ERROR	A new file cannot be opened. ⇒ Check whether free space is available on the diskette.
? FILE WRITE ERROR	A file cannot be written. ⇒ Check for diskette errors.
? error occurred! MPU was reset automatically	An unrecoverable error has occurred in PC4000E. ⇒ The target MCU is automatically reset to recover from the error.
RTT74.HLP NOT OPEN ERROR	The HLP file is not in the proper location. ⇒ Copy the HLP file to a the proper location with the file name RTT74.HLP.
RTT74.HLP NOT CLOSE ERROR	The help message file is not closed. ⇒ Check for diskette errors.
? INVALID AREA	Absolute specification is specifying an area outside the valid area. ⇒ Specify a valid area.
invalid line number [xxx]	Information for line number xxx is not defined in the source line debug information.
invalid parameter(s)	There is an error in the specified argument.
invalid source file[xxx]	Source file xxx is not in the source line debug information.
name.HEX NOT CLOSE ERROR	The HEX file cannot be closed after loading. ⇒ Check for diskette errors.
? name.HEX DATA ERROR	There is an error in the Intel hexadecimal file format or the Intel hexadecimal file address is invalid. ⇒ Specify a proper Intel hexadecimal file. Change the origin to point to the correct address.

Error Message	Description
? name.HEX NOT OPEN ERROR	HEX file cannot be found or the file cannot be opened. ⇒ Check that the file exists on the diskette.
No data	There is no data. ⇒ Check that the MCU information is in the DAT file and the DAT file version is correct.
no define	Symbolic information is not defined.
no default value[xxx]	There is no default for xxx. ⇒ Explicitly specify the argument.
no source line debug information	There is no source line debug information in the loaded SYM file.
not enough memory	There is not enough memory to start the shell. ⇒ Expand the memory to 50000 bytes or more.
? OFFSET MISS MATCH	The range of addresses that can be specified in absolute specification mode is limited by the value offset mode value specified during realtime trace. This message appears when this limit is exceeded. ⇒ Specify a correct value corresponding to the offset mode.
Out of heap space.	There is insufficient memory. ⇒ Expand the memory.
PARAMETER?	Number other than 0-9 is entered. ⇒ Enter a decimal number between 0 and 9 as parameter.
PARAMETER ZERO!!	0 is specified as the count for relative specification mode. ⇒ Specify a non-zero value as count.
? PORT SET ERROR	The #D or #S command was executed in M50734 mode with the DME signal output port P05 set to input mode. ⇒ Set the port mode to output.
? programmable limit! MCU was reset automatically	Memory modification is repeated in a loop in M50734 mode and inaccessible area has been reached. ⇒ The target MCU is automatically reset to recover from the error.
? START LINE<END LINE	The start address exceeds the end address in absolute specification mode range specification. ⇒ Keep the start value smaller than the end value.
symbol file format error	There is an error in the SYM file format. ⇒ Use SYM file generated with LINK74.
too large section number[xxx]	xxx exceeds the largest defined section number.
too large line number[xxx]	xxx exceeds the largest defined line number.

APPENDIX B COMMAND SCOPE LIST

B.1 Command Scope

The scope of the following commands depend on the MCU (option board) being used.

1. Difference in scope

Table B.1 shows the scope of each command according to MCU.

Table B.1 Command Scope by MCU

Mode/Command	A, L, X	D, F, M, S
M50740[1] Single chip mode	$F000_{16}$-$FFFF_{16}$	0000_{16}-$00FF_{16}$ $F000_{16}$-$FFFF_{16}$
M50745 Single chip mode	$E800_{16}$-$FFFF_{16}$	0000_{16}-$00FF_{16}$ $E800_{16}$-$FFFF_{16}$
M50747[1] Single chip mode	8000_{16}-$FFFF_{16}$	0000_{16}-$013F_{16}$ 8000_{16}-$FFFF_{16}$
M50747 Microprocessor mode	$BANK_0$-$BANK_3$ (2000_{16}-$FFFF_{16}$)	0000_{16}-$FFFF_{16}$
M50754 Single chip mode	$E000_{16}$-$FFFF_{16}$	0000_{16}-$027F_{16}$ $E000_{16}$-$FFFF_{16}$
M509551[1] Single chip mode	8000_{16}-$FFFF_{16}$	0000_{16}-$027F_{16}$ 8000_{16}-$FFFF_{16}$
M37450[1] Single chip mode	8000_{16}-$FFFF_{16}$	0000_{16}-$0FEF_{16}$ 8000_{16}-$FFFF_{16}$
M37450[2] Microprocessor mode	0000_{16}-$FFFF_{16}$	0000_{16}-$013F_{16}$
M50734[2]	0000_{16}-$FFFF_{16}$	0000_{16}-$FFFF_{16}$
M37100[1] Single chip mode	8000_{16}-$FFFF_{16}$	0000_{16}-$7FFF_{16}$

Notes

1. Memory available to the debugger. Note that it is greater than the microcomputer ROM and RAM.

2. Some areas are inaccessible depending on the program counter. Those areas are shown in Table B.2.

Table B.2 Area Inaccessible in M50734/M37450 Microprocessor Mode

Area	Access
SFR (00D016-00FF16)	Inaccessible when PC is 00CE16-00FF16
External memory other than SFR (when bank 7 is internal)	Inaccessible when PC is 00CE16-00FF16 or is external
External memory other than SFR (when bank 7 is external)	Entire area is inaccessible

2. Difference in block stored with the O command

The block to be stored with the O command may need to be entered depending on the MCU.

This value is selected according to the address scope.

Table B.3 shows the input value according to MCU.

Table B.3 O Command Input Value by MCU

MCU Mode	Scope: Selection	Input Value
M50747 Single chip mode	8000 - 8FFF : 8 9000 - 9FFF : 9 A000 - AFFF : A B000 - BFFF : B C000 - CFFF : C D000 - DFFF : D E000 - EFFF : E F000 - FFFF : F	8-F
M50747 Microprocessor mode M50734	(0000 - 0FFF : 0) (1000 - 1FFF : 1) (2000 - 2FFF : 2) (3000 - 3FFF : 3) (4000 - 4FFF : 4) (5000 - 5FFF : 5) (6000 - 6FFF : 6) (7000 - 7FFF : 7) (8000 - 8FFF : 8) (9000 - 9FFF : 9) (A000 - AFFF : A) (B000 - BFFF : B) (C000 - CFFF : C) (D000 - DFFF : D) E000 - EFFF : E F000 - FFFF : F Value in ( ) differs, depending on the bank setting contents.	0-F
M50740 Single chip mode	F000 - FFFF : F	F
M50745 Single chip mode	F800 - EFFF : E F000 - FFFF : F	E or F
MCU Mode	Scope: Selection	Input Value

M50754 Single chip mode	F000 - EFFF : E F000 - FFFF : F	E or F
M50955 Single chip mode M37450 Single chip mode M37100 Single chip mode	8000 - 8FFF : 8 9000 - 9FFF : 9 A000 - AFFF : A B000 - BFFF : B B000 - BFFF : B C000 - CFFF : C D000 - DFFF : D E000 - EFFF : E F000 - FFFF : F	8-F
M50734 (board without RTT function) M37450 Microprocessor mode	(0000 - 0FFF : 0) (1000 - 1FFF : 1) (2000 - 2FFF : 2) (3000 - 3FFF : 3) (4000 - 4FFF : 4) (5000 - 5FFF : 5) (6000 - 6FFF : 6) (7000 - 7FFF : 7) (8000 - 8FFF : 8) (9000 - 9FFF : 9) (A000 - AFFF : A) (B000 - BFFF : B) (C000 - CFFF : C) (D000 - DFFF : D) (E000 - EFFF : E) (F000 - FFFF : F) (All area external?) Contents of () depends on the bank setting.	0-F

3. The difference between M50747 microprocessor mode and other modes.

- Reset (Z) command
  The contents of the banks set on the option board are displayed when the reset command is executed. Figure B.1 shows an example.

```
]Z<RET>
0000   RAM(2K or 8K)      8000   -----
2000   -----              A000   BANK0
4000   BANK1              C000   BANK2
6000   -----              E000   BANK3
]
```

Figure B.1 Z Command Execution Example in M50747 Microprocessor Mode

4. The difference between M50734 mode and other modes.

- Reset (Z) command
  The contents of the banks set on the option board are displayed when the reset command is executed. Figure B.2 shows an example of using RTT74 with PCA4034G02.

```
]Z<RET>
0XXX   BANK 0 External       2000   BANK 1 External
4XXX   BANK 2 External       6000   BANK 3 External
8XXX   BANK 4 External       A000   BANK 5 External
CXXX   BANK 8 External       E000   BANK 7 Internal
]
```

Figure B.2 Z Command Execution Example in M50734 Mode

- The display is the same as M50747 microprocessor mode when using the PCA4034 with a board without the RTT74 function, or when using the PCA4034R with RTT74. See Figure B.1.

- Data memory area access, change command
  The D# and S# commands are used to access and change the contents of data memory in M50734 mode.

  The command syntax is the same as D and S commands, but "#" must be appended at the end of the command.

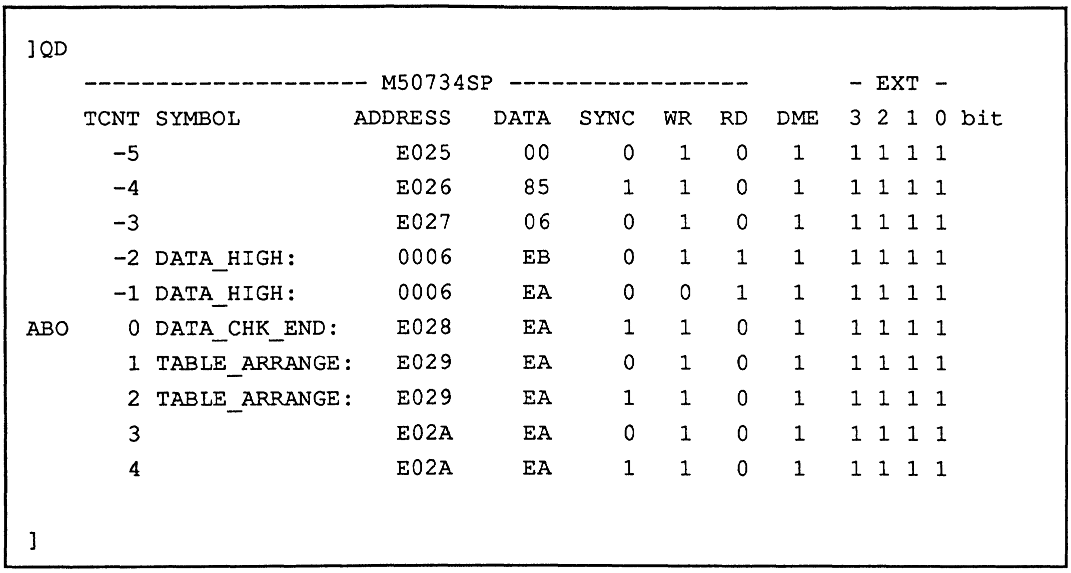
  The same symbols as program memory area can be used for data memory area.

```
]D# e000,e010
#E000 08 48 3F FF 1F FF 20 0C E1 68 28 40 00 00 00 00 .H?.....h(@....
#E010 A2   .
]S# E000
RESET:          #E000  08  52
                #E001  48  54
                #E002  3F  54
INITIAL:        #E003  FF  .

]
```

Figure B.3 Data Memory Access Display Example in M50734 Mode

- QD command Display
  The displayed result of the QD command in M50734 mode differs with other modes. In this mode, the count, symbol, address, data, sync signal, read signal, write signal, DME signal, and external signal (4 bit) are displayed.

  The difference with other modes is that the DME signal is added, read and write signals are separated, and only 4 bits of the external signals are displayed. Figure B.4 shows a display example.



```
]QD
        -------------------- M50734SP -----------------        - EXT -
        TCNT SYMBOL          ADDRESS  DATA  SYNC  WR  RD  DME  3 2 1 0 bit
          -5                   E025    00     0   1   0   1   1 1 1 1
          -4                   E026    85     1   1   0   1   1 1 1 1
          -3                   E027    06     0   1   0   1   1 1 1 1
          -2 DATA_HIGH:        0006    EB     0   1   1   1   1 1 1 1
          -1 DATA_HIGH:        0006    EA     0   0   1   1   1 1 1 1
ABO        0 DATA_CHK_END:     E028    EA     1   1   0   1   1 1 1 1
           1 TABLE_ARRANGE:    E029    EA     0   1   0   1   1 1 1 1
           2 TABLE_ARRANGE:    E029    EA     1   1   0   1   1 1 1 1
           3                   E02A    EA     0   1   0   1   1 1 1 1
           4                   E02A    EA     1   1   0   1   1 1 1 1

]
```

Figure B.4 Realtime Trace Access and Display in M50734 Mode

- Access disabled area

  When a board without RTT functions is used with the PCA4034G02 board and data is written 1 byte below the access disabled area, an error occurs and the MCU is automatically reset.

  This error occurs because the address is automatically incremented when data is written in memory and consequently an inaccessible area is accessed.

5. Difference between M37450 microprocessor mode and other modes

   The differences from other modes in M37450 microprocessor mode is the same as the case of a board without RTT functions (Board: PCA4034G02) in M50734 mode. Refer to “Differences from other modes in M50734 mode”.

   The difference between the M37450 microprocessor mode and other modes is the same as the case for board (option board: PCA4034G02) without RTT function in M50734 mode. Refer to "Difference between M50734 mode and other modes."

APPENDIX C. SETTING THE BAUD RATE

This appendix describes how to set the baud rate of the PCs supported by RTT74. The operating system (OS) is MS-DOS (or PC-DOS).

C.1 NEC PC-9801 Series and PC-98XA/XL/LT

RTT74 can be used with NEC[1] PC-9801 series and PC-98XA/XL/LT computers.

1. Setting serial I/O mode and transmission speed

   The transmission speed can be set by using the MS-DOS SPEED command. An example is shown below.

   ```
   A>SPEED R0 9600 B8 PN S2<RET>
   ```

   If XON is specified with the SPEED command, it is necessary to clear XON. In this case the command is entered as follows:

   ```
   A>SPEED R0 9600 B8 PN S2 NONE<RET>
   ```

   (a) R0: standard RS-232C interface.
   (b) 9600: set data transfer speed to 9600
   (c) B8: set data length to 8 bits
   (d) PN: no parity check
   (e) S2: set stop bits to 2
   (f) NONE: no XON/XOFF control

C.2 IBM PC/XT/AT

RTT74 can be used with IBM[2] PC/XT/AT

- Setting the serial I/O mode and transmission speed.

  The transmission speed of RTT74 is automatically set to 9600 bps when IBM PC/XT/AT is started. Therefore, the transmission speed of PC4000E must be set to 9600 bps with jumper switches on the internal board.

  Refer to the PC4000E operation manual for further information.

[1] NEC: Nippon Electric Corporation
[2] IBM: International Business Machines

APPENDIX D. CABLE CONNECTION

The following are the cable connections of computers supported by RTT74.

D.1 NEC PC-9801 series and PC-98XA/XL/LT

RTT74 can be used with NEC PC-9801 series and PC-98XA/XL/LT computers.

- Connection between PC-9801 series or PC-98XA/XL/LT computers and the PC 4000E

  The connection between the PC-9801 series or PC-98XA/XL/LT computers and the PC 4000E is made with a cable attached to the PC 4000E. Refer to the PC 4000E operation manual for further information.

D.2 IBM PC/XT/AT

RTT74 can be used with the IBM PC/XT/AT computers.

To connect an IBM PC/XT/AT and PC4000E, the cable attached to the PC4000E must be modified as follows.

1. Obtain a D-sub 25 pin female connector.

2. Open the connector housing marked "HOST" on the cable attached to the PC 4000E, and pull out the D-sub male 25 P connector.

3. Cut each wire connected to the male connector and connect each wire to the same pin of the female connector. In other words the male and female connectors are replaced without changing the pin connections.

4. Insert the replaced female connector in the housing.

After the cable is modified, connect the IBM PC/AT/XT and the PC4000E according to the PC4000E operation manual.

[Precautions when using a 9-pin cable]

- When a personal computer is equipped with a 9 pin type connector, such as the Mitsubishi Electric MAXY, connection can be made with the following cables.

- The type name of the RS9-pin cable is "MSCH-TOOL-H" and its specification is "RS9-pin cable".

- Fig. D.1 shows the wiring diagram.

- The PC4000E can be connected to MAXY with its serial I/O switch JP1 set to its factory setting MULTI16. (Different from IBM-PC setting.)

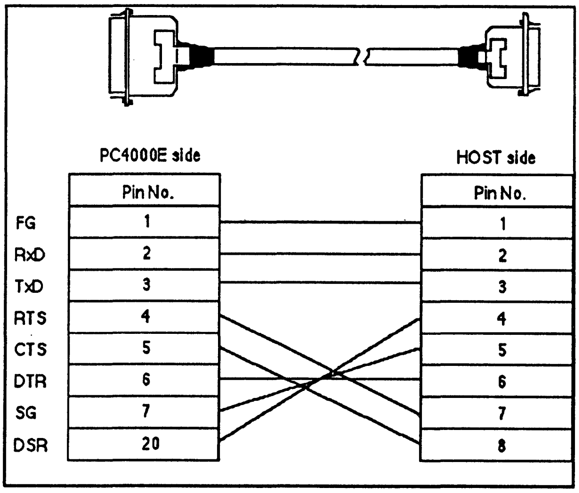


Figure D.1 RS9-pin Cable Connection

APPENDIX E. RTT74 SPECIFICATIONS

E.1 Standard Environment

Conditions listed in Table E.1 are presumed to be under the standard MS-DOS environment.

Table E.1 MS-DOS standard environment

Item	Specification
MS-DOS version	V.3.1
Memory capacity	The user available memory capacity is 256 KB. This value is the result of the MS-DOS standard CHKDSK command.

E.2 RTT74 Specifications

The RTT74 specification under standard MS-DOS environment is as follows:

- Number of characters that can be entered in a command line

  127 characters maximum (the rest is ignored)

- Allowed number of symbols

  Six files (TEST1.A74-TEST6.A74) are created each containing 1000 global label definitions with as shown in the program in Figure E.1. Each label is 8 characters long.

  Each file is assembled and the six files are linked to create a SYM file (TEST.SYM) containing information for 6000 symbols.

  When this file is loaded, approximately 3300 global labels are available when there is 256K bytes of free space with the CHKDSK command as shown in Figure E.3.

```
;        TEST1.A74
         .section       p
:JCFU0001:     nop
:ZQSQ0002:     nop
:LMIR0003:     nop
              :
              :
              :
:GPSU0998:     nop
:ASAV0999:     nop
:SPHR1000:     nop
         .end
```

Figure E.1 Global Symbol Registration Sample Program (TEST1.A74)

```
A>SRA74 TEST1<RET>
MELPS 740 SRA74 V.1.01.01C
Copyright 1989, 1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

now processing pass 1  ( TEST1.A74 )
----*----*
now processing pass 2  ( TEST1.A74 )
----*----*
ERROR   COUNT            00000
WARNING COUNT            00000
STRUCTURED STATEMENT     00000 LINES
TOTAL   LINE ( SOURCE ) 01003 LINES
TOTAL   LINE ( OBJECT ) 01003 LINES
COMMENT LINE ( SOURCE ) 00001 LINES
COMMENT LINE ( OBJECT ) 00001 LINES
OBJECT  SIZE ( p      ) 01000 (03E8) BYTES
A>SRA74 TEST2<RET>
        :
A>SRA74 TEST3<RET>
        :
A>SRA74 TEST4<RET>
        :
A>SRA74 TEST5<RET>
        :
A>SRA74 TEST6<RET>
        :
A>LINK74<RET>
MELPS 740 LINKER V.1.01.01C
Copyright 1989,1990, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

Relocatable files (.R74) >> TEST1 TEST2 TEST3 TEST4 TEST5 TEST6
Libraries         (.LIB) >>
Section information      >> P=C000
Command parameter        >> -S -FTEST
now processing pass 1
processing "TEST1.R74"
processing "TEST2.R74"
processing "TEST3.R74"
processing "TEST4.R74"
processing "TEST5.R74"
processing "TEST6.R74"
```

```
now processing pass 2
processing "TEST1.R74"
processing "TEST2.R74"
processing "TEST3.R74"
processing "TEST4.R74"
processing "TEST5.R74"
processing "TEST6.R74"
TOTAL ROM SIZE 6000 (1770H) BYTES
TOTAL RAM SIZE 0 (0H) BYTES
```

Figure E.2 Assemble and Link Example

```
A>CHKDSK<RET>

Volume RAM-DRIVE   created 1991-3-4 3:00

  2022912 bytes total disk space
        0 bytes in 1 hidden files
   490496 bytes in 18 user files
  1532416 bytes available on disk

   655360 bytes total memory
   256384 bytes free

A>RTT74 -HOST=IBMPC -MCU=M37471 TEST.SYM<RET>
MELPS 740/M38000 DEBUGGER CONTROL SOFTWARE V.2.10.00C
Copyright 1989, 1990, 1991, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SOFTWARE CORPORATION
All Rights Reserved.

HOST MACHINE --> IBM PC/XT/AT

DEBUGGER SYSTEM  ---> PC4600
MONITOR          ---> V.1.01-0
MCU              ---> M37471

Out of heap space
GLOBAL SYMBOL LOADED
TEST.SYM LOAD END --> 3305 SYMBOLS DEFINED
>
```

Figure E.3 Execution Example

E.3 Precautions when Using RTT74

1. Program location used for debugging

   The source file must be coded so that the final address is FFFFH.

   (Example)

   With M37410M4-XXXFP, the ROM address is 2000-3FFF, however the source file must be set to 3FFFH to FFFFH.

2. Stack size required by the debugger

   During debugging, the option board uses 1 to 6 bytes of stack memory. The contents of the memory pointed by the stack pointer cannot be accessed.

   Refer to the board's operation manual for more information.

INDEX

CONTACT ADDRESSES FOR FURTHER INFORMATION

JAPAN

Semiconductor Marketing Division
Mitsubishi Electric Corporation
2-3, Marunouchi 2-chome
Chiyoda-ku, Tokyo 100, Japan
Telex: 24532 MELCO J
Telephone: (03) 3218-3473
(03) 3218-3499
Facsimile: (03) 3214-5570

Overseas Marketing Manager
Kita-Itami Works
4-1, Mizuhara, Itami-shi,
Hyogo-ken 664, Japan
Telex: 526408 KMELCO J
Telephone: (0727) 82-5131
Facsimile: (0727) 72-2329

HONG KONG

Mitsubishi Electric (H.K.) Ltd.
41st fl., Manulife Tower, 169,
Electric Road, North Point, Hong Kong
Telex: 60800 MELCO HX
Telephone: 510-0555
Facsimile: 510-9830, 510-9822,
510-9803

SINGAPORE

MELCO SALES SINGAPORE PTE, LTD.
307 Alexandra Road #05-01/02
Mitsubishi Electric Building
Singapore 0315
Telex: RS 20845 MELCO
Telephone: 4732308
Facsimile: 4738944

TAIWAN

MELCO-TAIWAN CO., Ltd.
1st fl., Chung-Ling Bldg.,
363, Sec. 2, Fu-Hsing S Road,
Taipei R.O.C.
Telephone: (02) 735-3030
Facsimile: (02) 735-6771
Telex: 25433 CHURYO "MELCO-TAIWAN"

U.S.A.

NORTHWEST
Mitsubishi Electronics America, Inc.
1050 East Arques Avenue
Sunnyvale, CA 94086
Telephone: (408) 730-5900
Facsimile: (408) 730-4972

SAN DIEGO
Mitsubishi Electronics America, Inc.
16980 Via Tazon, Suite 220
San Diego, CA 92128
Telephone: (619) 451-9618
Facsimile: (619) 592-0242

DENVER
Mitsubishi Electronics America, Inc.
4600 South Ulster Street
Metropoint Building, 7th Floor
Denver, CO 80237
Telephone: (303) 740-6775
Facsimile: (303) 694-0613

SOUTHWEST
Mitsubishi Electronics America, Inc.
991 Knox Street
Torrance, CA 90502
Telephone: (213) 515-3993
Facsimile: (213) 217-5781

SOUTH CENTRAL
Mitsubishi Electronics America, Inc.
1501 Luna Road, Suite 124
Carrollton, TX 75006
Telephone: (214) 484-1919
Facsimile: (214) 243-0207

NORTHERN
Mitsubishi Electronics America, Inc.
15612 Highway 7 #243
Minnetonka, MN 55345
Telephone: (612) 938-7779
Facsimile: (612) 938-5125

NORTH CENTRAL
Mitsubishi Electronics America, Inc.
800 N. Bierman Circle
Mt. Prospect, IL 60056
Telephone: (312) 298-9223
Facsimile: (312) 298-0567

NORTHEAST
Mitsubishi Electronics America, Inc.
200 Unicorn Park Drive
Woburn, MA 01801
Telephone: (617) 932-5700
Facsimile: (617) 938-1075

MID-ATLANTIC
Mitsubishi Electronics America, Inc.
800 Cottontail Lane
Somerset, NJ 08873
Telephone: (201) 469-8833
Facsimile: (201) 469-1909

SOUTH ATLANTIC
Mitsubishi Electronics America, Inc.
2500 Gateway Center Blvd., Suite 300
Morrisville. NC 27560
Telephone: (404) 368-4850
Facsimile: (404) 662-5208

SOUTHEAST
Mitsubishi Electronics America, Inc.
Town Executive Center
6100 Glades Road #210
Boca Raton, FL 33433
Telephone: (407) 487-7747
Facsimile: (407) 487-2046

CANADA
Mitsubishi Electronics America, Inc.
6185 Ordan Drive, Unit #110
Mississauga, Ontario, Canada L5T 2E1
Telephone: (416) 670-8711
Facsimile: (416) 670-8715

Mitsubishi Electronics America, Inc.
300 March Road, Suite 302
Kanata, Ontario, Canada K2K 2E2
Telephone: (416) 670-8711
Facsimile: (416) 670-8715

GERMANY

Mitsubishi Electric Europe GmbH
Headquarters:
Gothear Str. 8
4030 Ratingen 1, Germany
Telex: 8585070 MED D
Telephone: (02102) 4860
Facsimile: (02102) 486-115

Munich Office:
Arabellastraße 31
8000 München 81, Germany
Telex: 5214820
Telephone: (089) 919006-09
Facsimile: (089) 9101399

FRANCE

Mitsubishi Electric Europe GmbH
55, Avenue de Colmar
92563 Rueil Malmaison Cedex
Telex: 632326
Telephone: 47087871
Facsimile: 47513622

ITALY

Mitsubishi Electric Europe GmbH
Centro Direzionale Colleoni
Palazzo Cassiopea 1
20041 Agrate Brianza I-Milano
Telephone: (039) 636011
Facsimile: (039) 6360120

SWEDEN

Mitsubishi Electric Europe GmbH
Lastbilsvägen 6B
5-19149 Sollentuna, Sweden
Telex: 10877 (meab S)
Telephone: (08) 960468
Facsimile: (08) 966877

U.K.

Mitsubishi Electric (U.K.) Ltd.
Travellers Lane
Hatfield
Herts AL10 8XB, England, U.K.
Telephone: (0044) 7072 76100
Facsimile: (0044) 7072 78692

AUSTRALIA

Mitsubishi Electric Australia Pty. Ltd.
348 Victoria Road
Rydalmere Nsw 2116, Australia
Private Bag No.2 Rydalmere Nsw 2116
Telex: MESYDAA 126614
Telephone: (02) 684-7200
Facsimile: (02) 638-7072

MITSUBISHI SEMICONDUCTORS
Debug Control Software
for the MELPS 740 Series
RTT 74 Ver 2.10 Second Edition

Mar. First Edition 1992
Editioned by
Committee of editing of Mitsubishi Semiconductor USER'S MANUAL

Published by
Mitsubishi Electric Corp., Semiconductor Marketing Division

MITSUBISHI SEMICONDUCTORS

Debug Control Software for the MELPS 740 Series RTT74 Ver 2.10 Second Edition

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: MITSUBISHI DENKI BLDG., MARUNOUCHI, TOKYO 100 TELEX: J24532 CABLE: MELCO TOKYO

If these products or technologies fall under Japanese and/or COCOM strategic restrictions, diversion contrary thereto is prohibited.

H-EA024-A KI-9203 Printed in Japan (ROD)

New publication, effective Mar. 1992.
Specifications subject to change without notice.