MJX440 for TR4102/CW4020
User's Manual

Rev.1.10 2000/05/09

(English – PRELIMINARY)

This manual describes how to mount a MJX440 parallel interface (a PCI card or a PCMCIA card) and how to install a device driver for it.

Notes

- The copying of this manual, either wholly or in part, without the consent of Lightwell Corporation is prohibited.
- Lightwell disclaims any responsibility for any effects resulting from the use of this product.
- The specifications for this product and the contents of this manual are subject to change without notice.
- MS-DOS, Windows 95, Windows 98, Windows NT are registered trademarks of Microsoft.
- MULTI is a registered trademark of Green Hills Software.

Printed in Japan

Address: 5-20-12, Ogikubo, Siginami-ku, Tokyo, 167-0051, Japan
Telephone: 03-3392-3331
Fax: 03-3393-3878
E-mail: ZAXSupport@lightwell.co.jp
URL: http://www.lightwell.co.jp/ZAX/

May 2000

Congratulations on your recent purchase of MJX440 for TR4102/CW4020(“MJX440”).

This manual is composed of the following contents:

Chapter 1. Overview

This chapter describes the product configuration, provides an overview of MJX440, and explains the nomenclature of the various components.

Chapter 2. Setting the Parallel Interface

This chapter describes how to mount the parallel interface (a PCI card or a PCMCIA card) and how to install a device driver for it.

Chapter 3. Connecting the Hardware

This chapter describes how to connect the MJX440 to a host as well as how to connect the MJX440 to a target system.

Chapter 4. Installing the Software

This chapter describes how to install the software for the operation of MJX440.

Chapter 5. Setting the Environment for MJX440

This chapter describes how to set the environment before using the MJX440.

Chapter 6. Starting and Ending Software

This chapter describes how to start and end the software for the operation of the MJX440.

Chapter 7. MJX440 Commands

This chapter describes how to use the various MJX440 commands.

Chapter 8. Rapid Downloading

This chapter describes the procedure for effecting rapid downloading.

Appendixes

The Appendixes provide additional technical information, such as specifications and restrictions that are applicable to the target system.

もくじ

Chapter 1. Overview 6
1.1 Product Makeup 6
1.2 Overview of MJX440 9
1.3 Nomenclature 13
Chapter 2. Setting the Parallel Interface 16
Chapter 3. Connecting the Hardware 17
3.1 Connecting the MJX440 to the Host 17
3.2 Connecting an EJTAG probe 18
3.3 Connecting ROM Probes 19
3.4 Connecting External Trigger Cables 28
3.5 Connecting the Power Supply and Turning the Power On 30
Chapter 4. Installing the Software 31
Chapter 5. Setting the MJX440 Environment 37
Chapter 6. Starting and Terminating Software 39
Chapter 7. MJX440 Commands 43
ABORT 46
BATCH 47
BP 48
BP/A 51
BP/H 53
BP/S 54
CLEAR 55
CONFIG 56
DUMP 57
EXAMINE 58
FILL 59
GO 60
HISTORY 61
INIT 63
JOURNAL 64
LOAD 65
MOVE 66
PIN 67
QUIT 68
REGISTER 69

STEP 70
TRACE/M 71
TRACE/I 72
UNASM 74
VERSION 75
WAIT 76
XPIN 77
Chapter 8. Rapid Downloading 78
Appendix A. Specifications 79
Appendix B. Limits on Target Systems 80
Appendix C. EJTAG Connector 81
Appendix D. ROM Probes 83
Appendix E. Corresponding ROM Pin Assignment 86
Appendix F. LEDs 90
Appendix G. List of Register Names 91
Appendix H. MJX Binary File 92
Appendix I. Probing of the Target System 93
Appendix J. Configuration file details 100

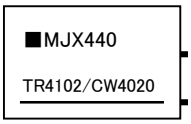
Chapter 1. Overview

This chapter describes the product configuration, provides an overview of MJX440, and explains the nomenclature of the various components.

1.1 Product Makeup

When shipped, each MJX440 for TR4102/CW4020 package contains the items indicated below. If you find any items missing, please contact Lightwell Corporation.

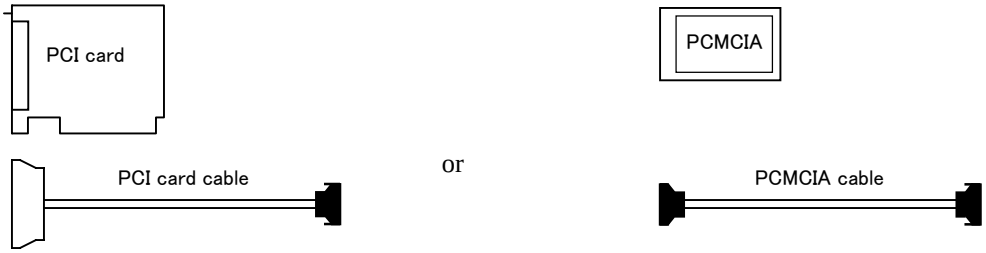
☐ MJX440 for TR4102/CW4020 system unit



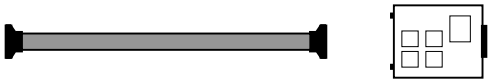
☐ AC Adapter

☐ AC cord

☐ Parallel Interface (PCI card, or PCMCIA card) and Parallel interface cable[*1]



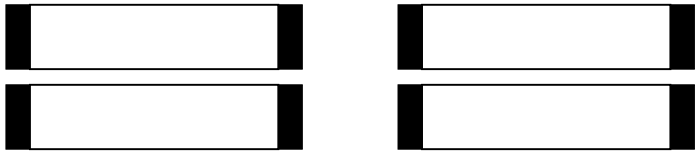
☐ EJTAG probe cable and EJTAG probe board



[*1] When shipped, the package contains a cord for either a PCI card or PCMICA card, but not both.

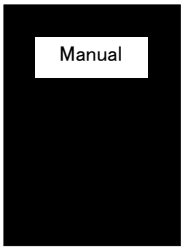
☐ ROM probe[*1]

☐ Four ROM probe cables



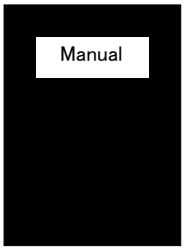
☐ Two External trigger cables

☐ Two Manuals



MJX440 for TR4102/CW4020

User's Manual



MJX440

Parallel Interface

Installation Manual

☐ Warranty card

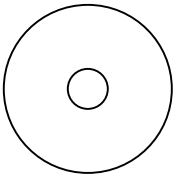
Warranty card

☐ User registration card

User registration card

【Important】Please fill out and mail the user registration card to Lightwell.

☐ CD-ROM



[*1] The type of a ROM probe included in the package depends on the type of ROM used in the system. For details, see "Table 1-1 Factory-shipped ROM probes" on the following page.

Following is a list of ROM probes that are included in the initial package, which depend on the type of ROM used in the product:

ROM type	ROM probe	
	ROM probe board	ROM plug
27010 27020 27040 27080 271000	J-101A×4	32pin×8
27C4000 16bit	J-102A×4	40pin×4
27C8000 16bit 27C16000 16bit	J-102A×4	42pin×4
271024 272048 274096	J-103A×4	40pin×4
27C4000 8bit	J-104A×4	40pin×8
27C8000 8bit 27C16000 8bit	J-104A×4	42pin×8

Table 1-1 Factory-shipped ROM probes

1.2 Overview of MJX440

MJX440 is an 52-pin Extended EJTAG connector equipped developmental aid device for the debugging of a TR4102 or CW4020 target system.

MJX440: Its Principal Features

- Because it uses the target system's EJTAG connector, MJX440 allows ready connection to the target system.
- MJX440 operates stably even when working with a high-speed CPU.
- MJX440 permits any in-cuicuit connection to the ROM and can debug programs running on ROM.
- MJX440 can be used in conjunction with the high-level language Debugger MULTI.
- The use of MJX440 commands permits the full use of the hardware on which MJX440 is based.
- MJX440 supports a realtime tracing function.
- MJX440 supports a hardware breakpoint function.
- MJX440 permits the rapid downloading of programs (440 kbytes/sec for EJTAG connection; 4Mbytes/sec for ROM in-circuit connection).

Minimum configuration

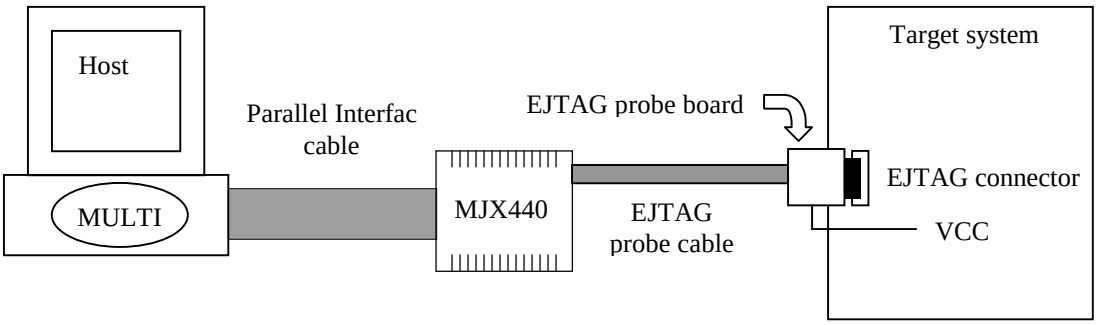


Figure 1-1. Minimum configuration[*1]

MJX440 operates in the minimum configuration showin in Figure 1-1. When used in its minimum configuration, MJX440 can debug programs that are stored in the RAM of the target system. In this operating mode, the program to be debugged is downloaded onto the RAM of the target system through the EJTAG probe cable.

[*1] In addition to EJTAG connector, VCC that is one of external trigger cable(1) signal should be connected.

In the minimum configuration, MJX440 can execute programs stored in the ROM, but it cannot download programs onto the ROM region or set software breakpoints. Debugging a ROM program requires a ROM in-circuit connection of MJX440.

ROM in-circuit connection

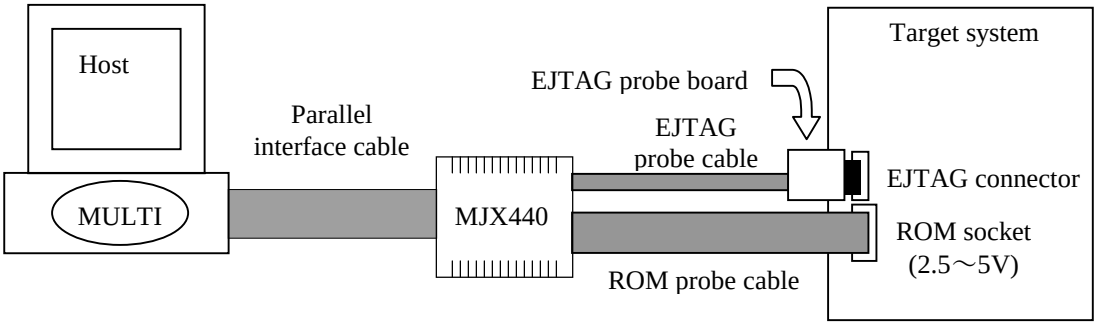


Figure 1-2. ROM in-circuit connection

Under a ROM in-circuit connection as shown in Figure 1-2, MJX440 can also debug programs that are stored in ROM. In this operating mode, the program to be debugged and stored in ROM is downloaded onto the emulation memory in the MJX440 rather than onto the RAM for the target system.

In this case, the ROM voltage is automatically recognized within the 2.5 ~ 5V range.

External trigger cable connection

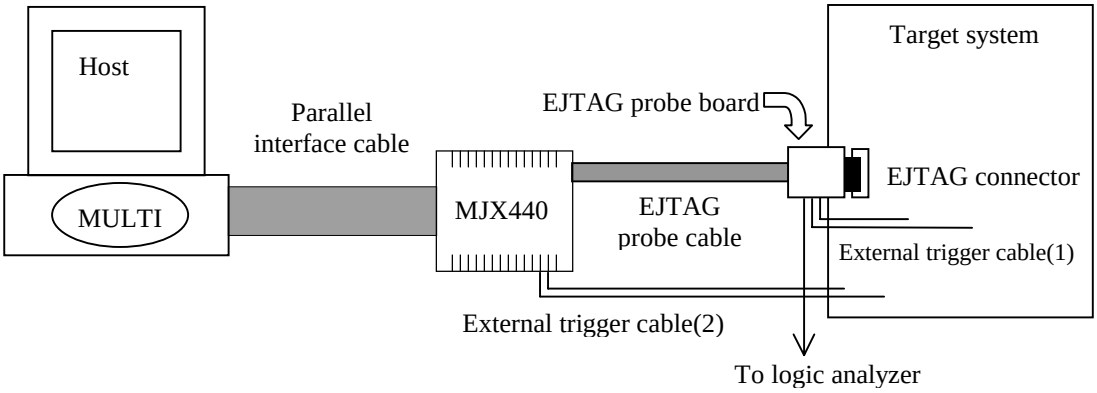


Figure 1-3. External trigger cable connection

As shown in Figure 1-3, external trigger cables can be used to implement the following functionality:

- Using the trace tigger as a logic analyzer trigger signal (External trigger cable(1) output)
- Storing target system signal status information in the realtime trace memory (External trigger cable(1) input)
- Displaying target system signal status information on the LED on the MJX440 (External trigger cable(2) input)

External trigger cable(1) connects with EJTAG probe board. External trigger cable(2) connects with MJX440 system unit.

Preliminary to using the MJX440

Before using the MJX440, you need to perform the preliminary steps described below, which can be performed by referring to Chapters 2 through 5. These steps need to be performed only once after the MJX440 is purchased.

- Setting the parallel interface
- Connecting the hardware
- Installing the software
- Setting the environment for the MJX440

Once these preliminary steps are completed, refer to Chapter 6 to start the software (MULTI or MJXDEBW) for operating the MJX440. Normal startup of the software indicates that the preliminary steps were successfully completed. If the software fails to start properly, check to see whether there was an error in performing the preliminary steps.

For a description of how to use the compiler or MULTI, see the respective manuals and release notes. An explanation of MJX440 commands for using functions specific to MJX440 is given in Chapter 7.

Refer to Chapter 8 for a description of how to effect a rapid downloading.

About MULTI

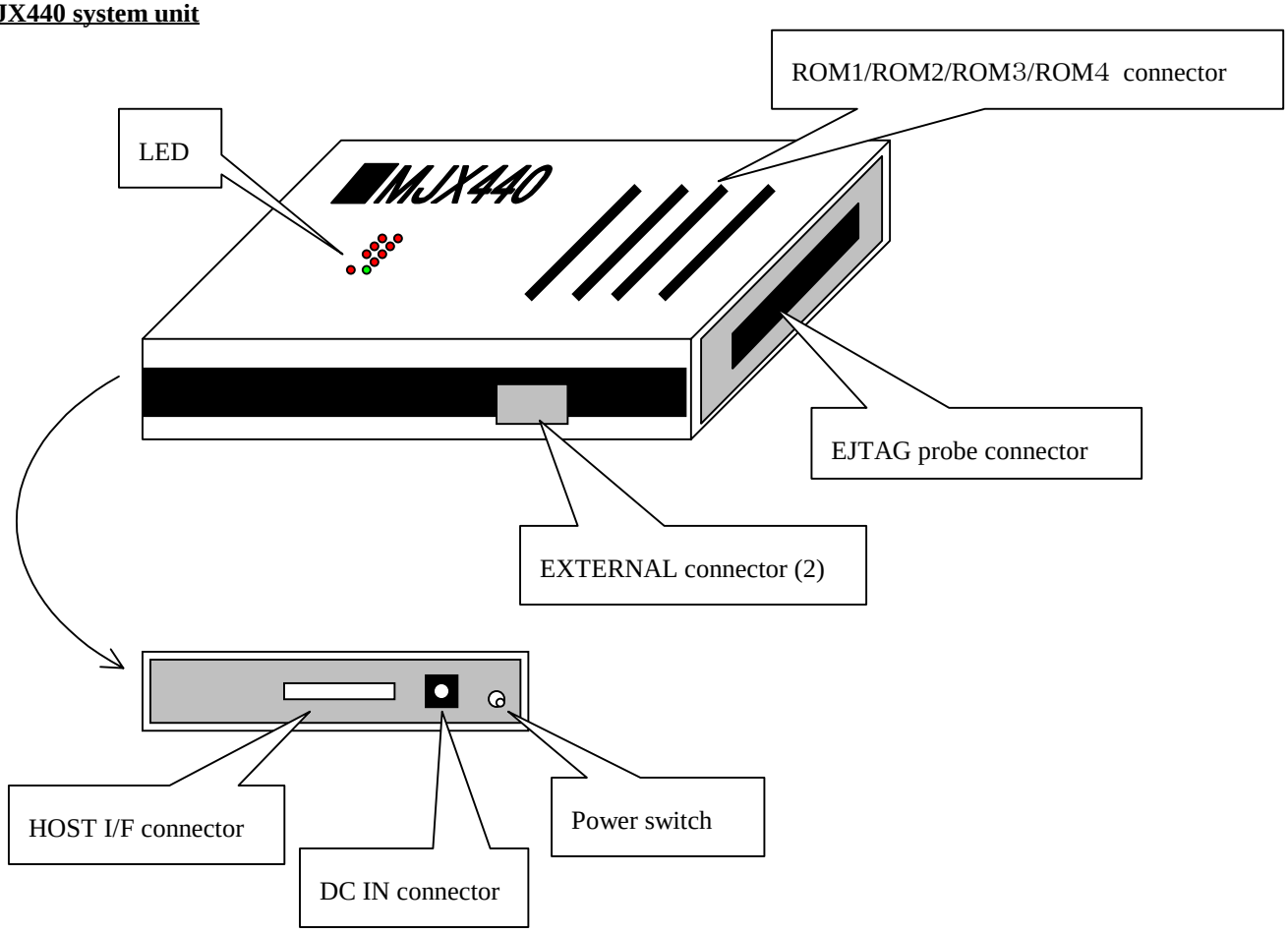
MULTI is a high-level language debugger that can be executed under various environments. You can run MULTI under the MJX440 environment by calling the server program MJXSERV.

About MJXDEBW

MJXDEBW refers to a quick debugger that supports MJX440 commands only. MJXDEBW may be a suitable choice when no high-level language debugging is needed or when the target system is to be checked by using the batch processing function.

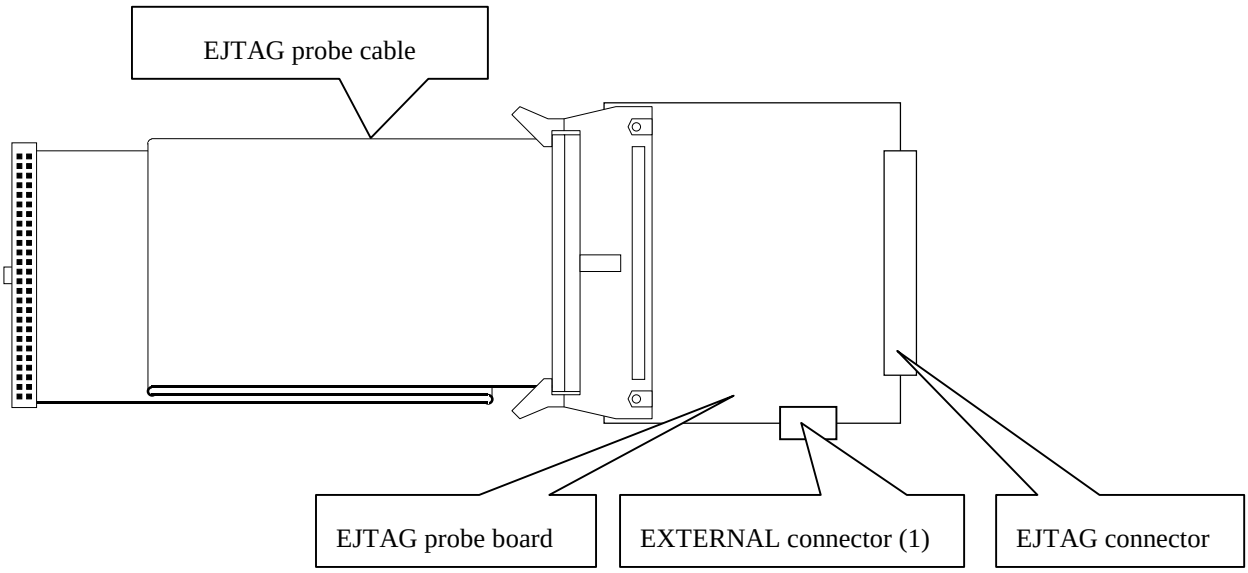
1.3 Nomenclature

MJX440 system unit



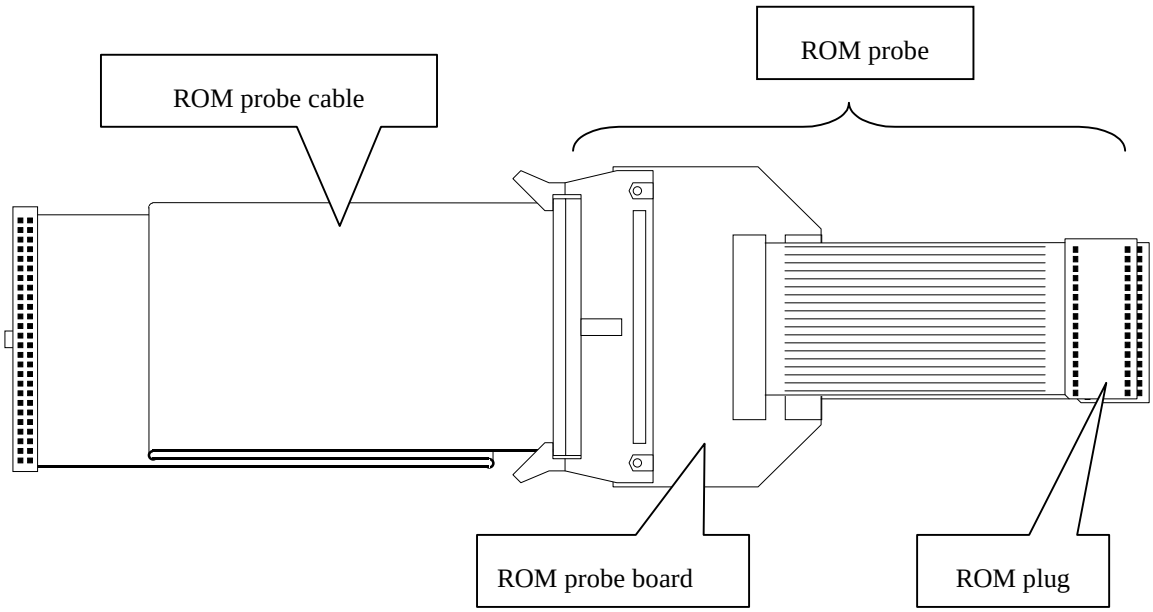
Power switch	This switch turn the power on and off.
DC IN connector	Connects the AC adapter.
HOST I/F connector	Connects the parallel interface cable.
EJTAG probe connector	Connects the EJTAG probe cable.
ROM1..ROM4 connector	Connects the ROM probe cable. The most inside part is a ROM1 connector, the most outside part is ROM4 connector.
EXTERNAL connector(2)	Connects External trigger cable(2) (dual line type).
LED	Displays power and connector plug-in status information.

EJTAG probe



EJTAG probe cable	Connects MJX440 and EJTAG probe board.
EJTAG probe board	This is the circuit board part of the EJTAG probe.
EJTAG connector	Connects to EJTAG connector on the target system.
EXTERNAL connector(1)	Connects External trigger cable(1) (single line type).

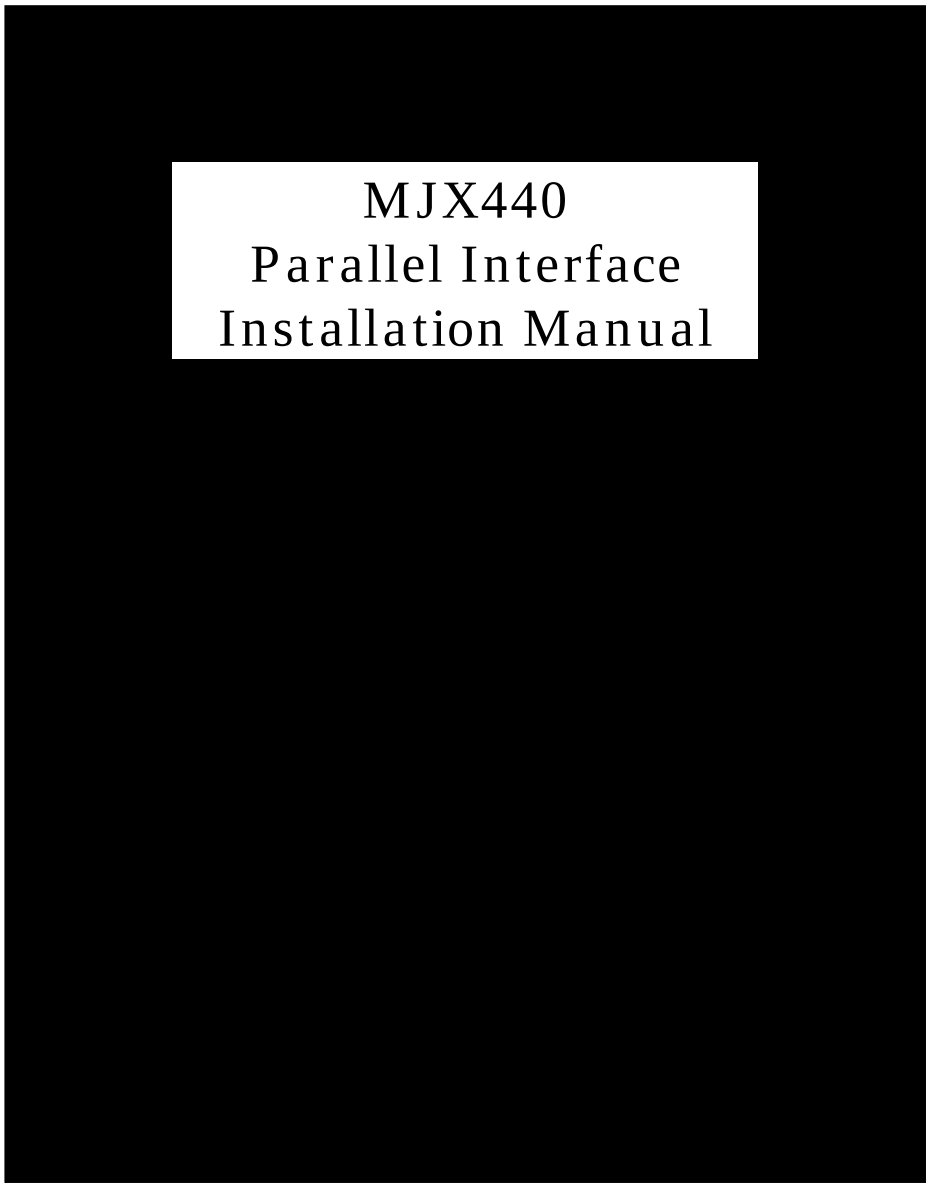
ROM probe



ROM probe cable	This cable connects the MJX440 to the ROM probe.
ROM probe	This refers to all probes that are connected to the target system ROM.
ROM probe board	This is the circuit board part of the ROM probe (with jumper settings).
ROM plug	This is a unit by which a device is connected to the ROM socket of the target system.

Chapter 2. Setting the Parallel Interface

To mount the parallel interface (a PCI or PCMCIA card) and to install the device driver for it, please refer to the following manual.



Chapter 3. Connecting the Hardware

This chapter describes how to connect the MJX440 to the host as well as how to connect the MJX440 to the target system.

【Important】Before connecting the MJX440 to other devices, be sure to turn off the power for the devices.

3.1 Connecting the MJX440 to the Host

Connect the PCI or PCMCIA card installed in the host system to the HOST I/F connector of the MJX440, using a parallel interface cable.

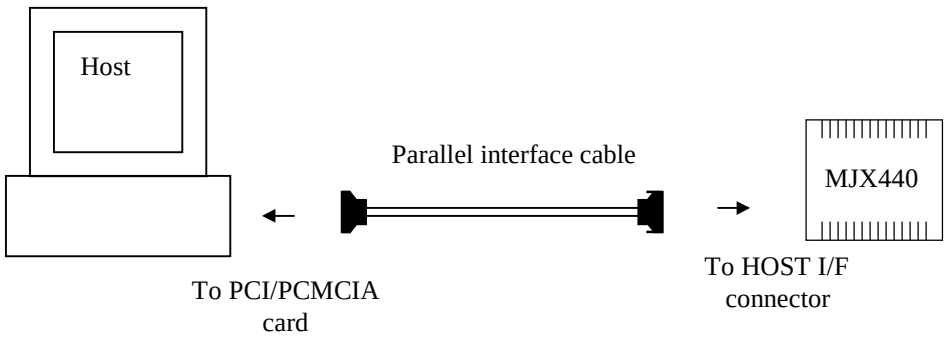


Figure 3-1. Connecting the MJX440 to the host

【Notes】The following limitations apply due to the thickness of the connector that connects the PCMCIA cable to the card:

- Even when there are two PCMCIA card slots, only one card can be used in some cases. There are also cases where a card can be inserted only into the bottom slot.
- An attempt to force the insertion of two cards can damage the PCMCIA card slot and the PCMCIA card connector part.
- PCMCIA card may not be used with PC which has only one PCMCIA card slot.

3.2 Connecting an EJTAG probe

Connect MJX440 system unit, EJTAG probe cable, EJTAG probe board, and the target system as shown in Figure 3-2.

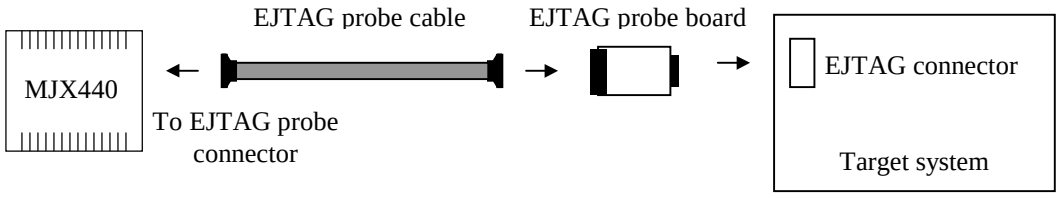


Figure 3-2. Connecting the EJTAG probe

【Note】To ensure the proper orientation of the connectors, make sure that the △ marks on the connectors match.

【Note】Be careful not to plug in the EJTAG probe in reverse. Make sure that the pin numbers on the connectors match. If connection was wrong, the system may be destroyed.

【Note】In addition to EJTAG connector, VCC that is one of external trigger cable 1's signal should be connected. See "3.4 Connecting External Trigger Cables".

3.3 Connecting ROM Probes

ROM probes need to be connected only when a ROM in-circuit connection must be perforemed.

First, set the jumper switches on your ROM brobe board according to the type of ROM to be used. Refer to “Appendix D. ROM Probes” for details.

Next, connect all the ROM probes that are supplied in the product package to the ROM probe cables.

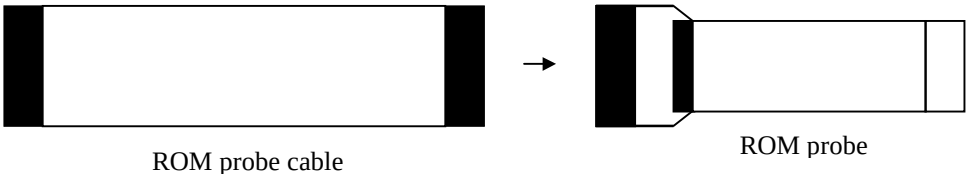


Figure 3-3. Connecting a ROM probe to a ROM probe cable (1)

In the next step, connect the ROM probe to the ROM socket on the target system, and connect the other end of the ROM probe cable to the ROM1/ROM2/ROM3/ROM4 connector on the MJX440.

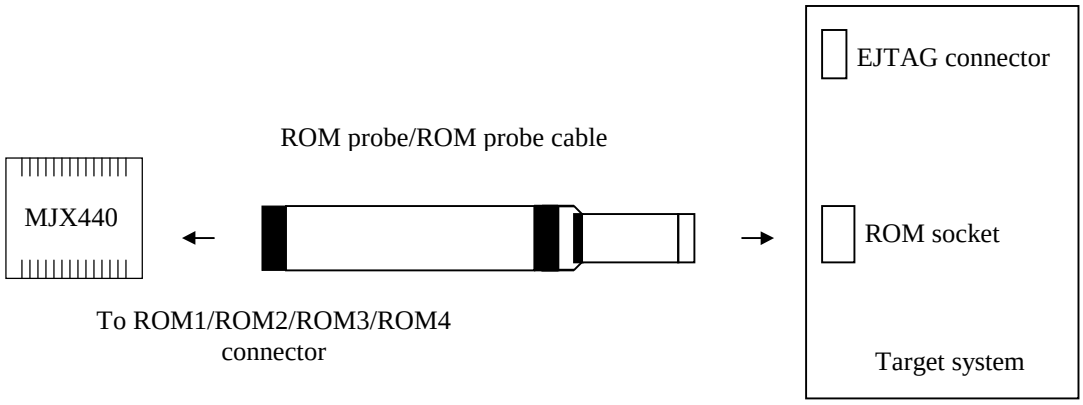


Figure 3-4. Connecting a ROM probe to a ROM probe cable (2)

How to connect a ROM probe depends on the following characteristics of the target system:

- ROM data bus width
- Number of ROM chips
- ROM access bus width

In Figures 3-5-1 through 3-5-11, select the connection diagram that matches your target system and connect the ROM probes to the ROM probe cables according to the diagram.

【Note】Be sure to turn off both the MJX440 and the target system.

【Note】Be careful not to plug in the ROM probe in reverse.

【Note】Distinguish the two ROM plugs that can be connected to ROM probes J-101A and J-104A by using the silk-printed characters (JROM1 or JROM2) that can be found on the ROM probe board.

One 8-bit bus width ROM chip and an 8-bit ROM access bus width:
(LED lit: ROM1)

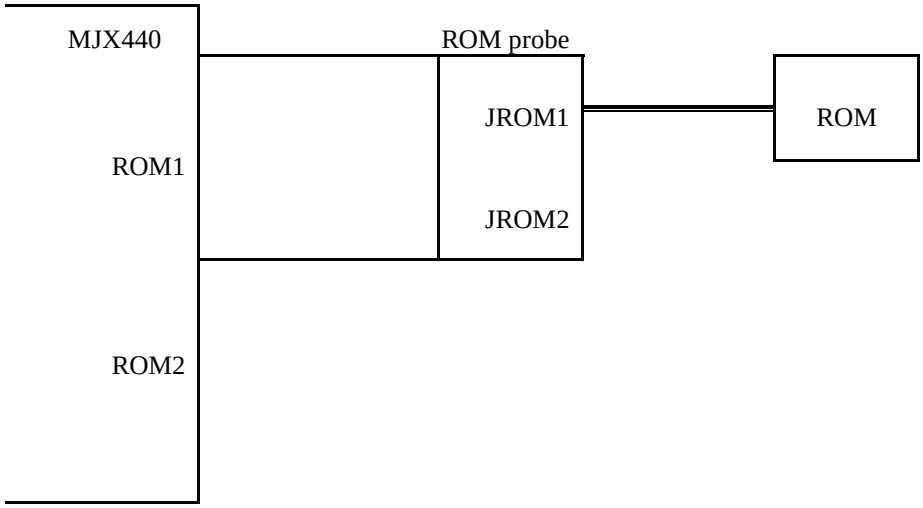


Figure 3-5-1. Connecting a ROM probe (1)

Two 8-bit bus width ROM chips and an 8-bit ROM access bus width:
(LEDs lit: ROM1)

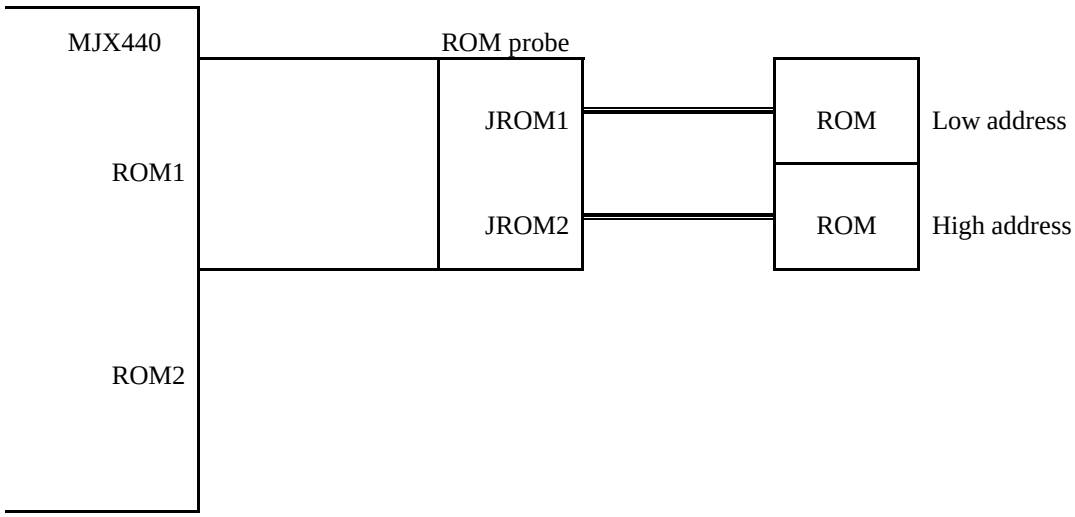


Figure 3-5-2. Connecting a ROM probe (2)

Four 8-bit bus width ROM chips and an 8-bit ROM access bus width:
(LEDs lit: ROM1, ROM2)

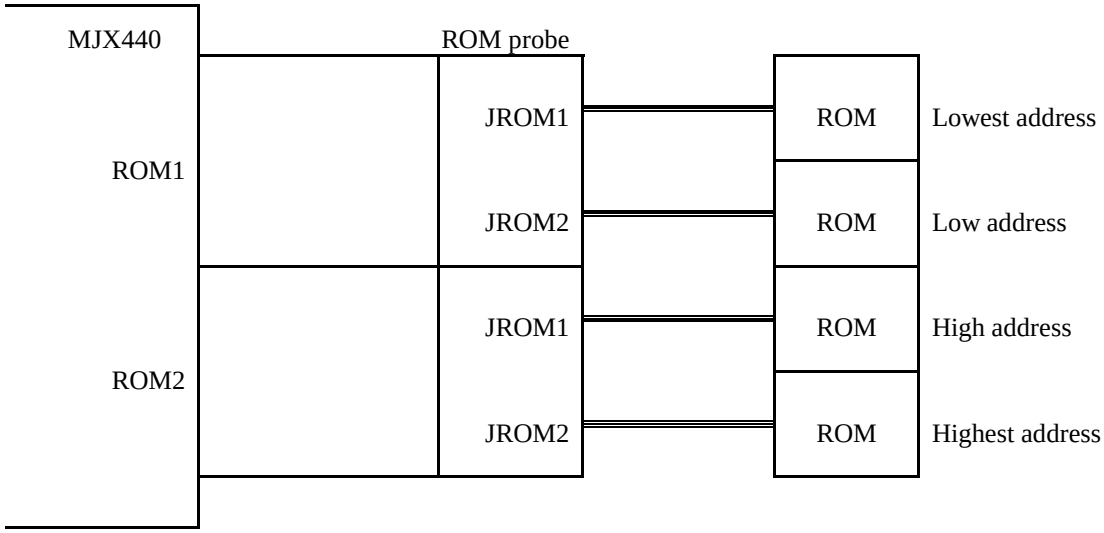


Figure 3-5-3. Connecting a ROM probe (3)

Two 8-bit bus width ROM chips and a 16-bit ROM access bus width:
(LEDs lit: ROM1)

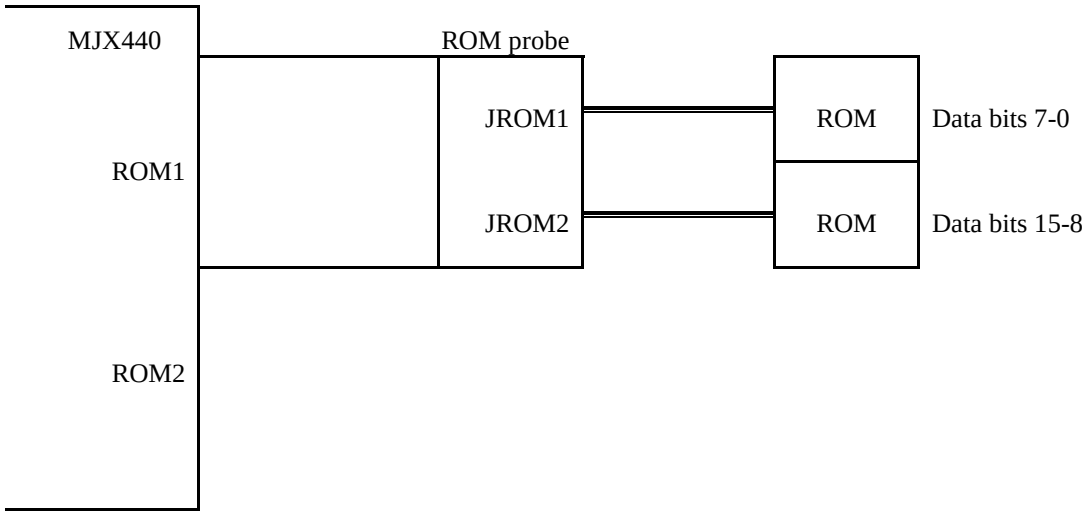


Figure 3-5-4. Connecting a ROM probe (4)

Four 8-bit bus width ROM chips and a 16-bit ROM access bus width:
(LEDs lit: ROM1, ROM2)

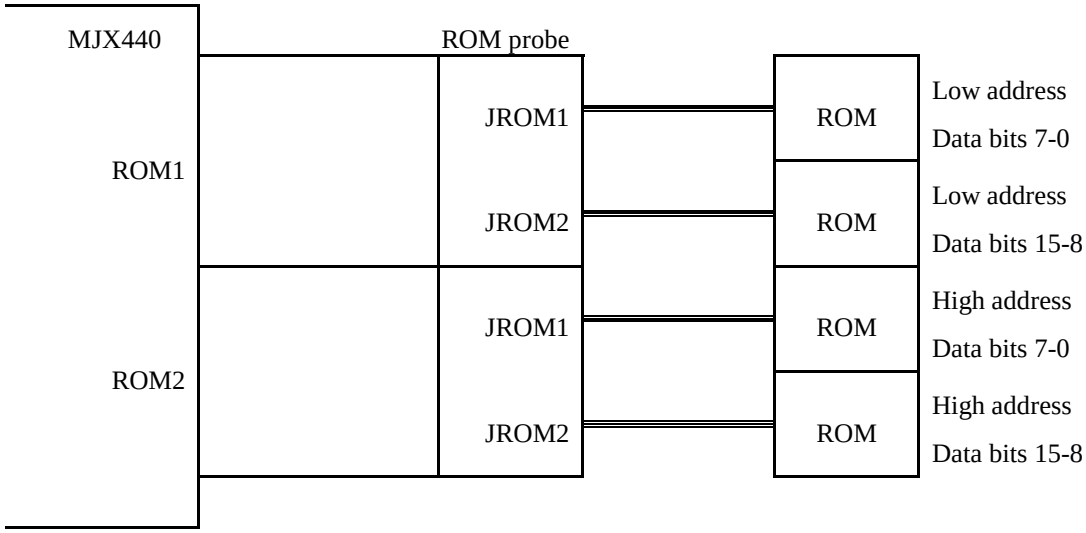


Figure 3-5-5. Connecting a ROM probe (5)

Four 8-bit bus width ROM chips and a 32-bit ROM access bus width:
(LEDs lit: ROM1, ROM2)

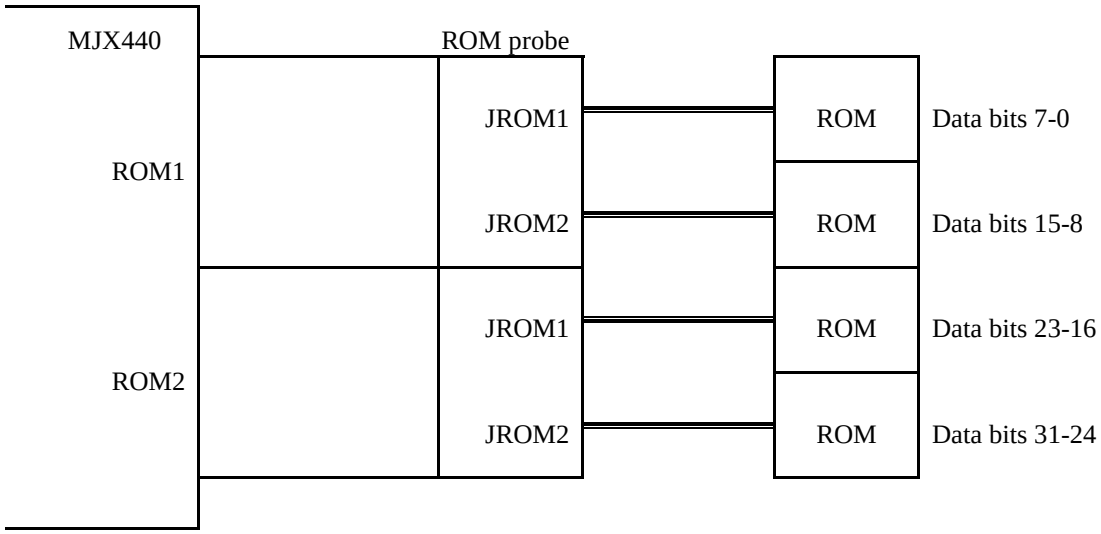


Figure 3-5-6. Connecting a ROM probe (6)

One 16-bit bus width ROM chip and a 16-bit ROM access bus width:
(LED lit: ROM1)

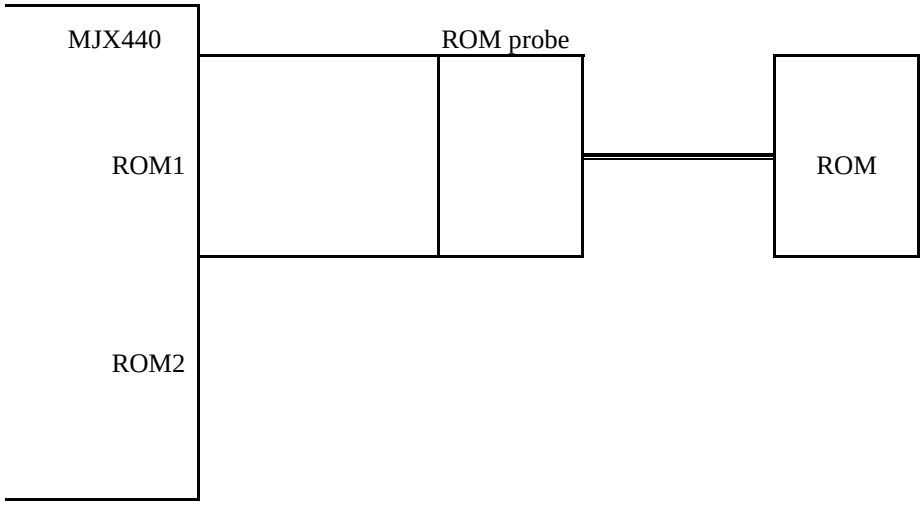


Figure 3-5-7. Connecting a ROM probe (7)

Two 16-bit bus width ROM chips and a 16-bit ROM access bus width:
(LEDs lit: ROM1, ROM2)

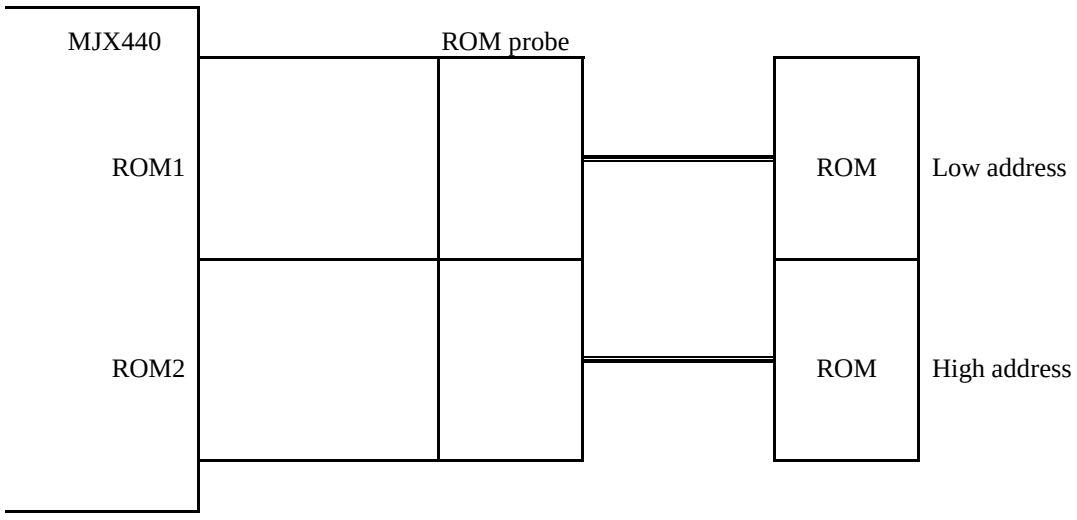


Figure 3-5-8. Connecting a ROM probe (8)

Two 16-bit bus width ROM chips and a 32-bit ROM access bus width:
(LEDs lit: ROM1, ROM2)

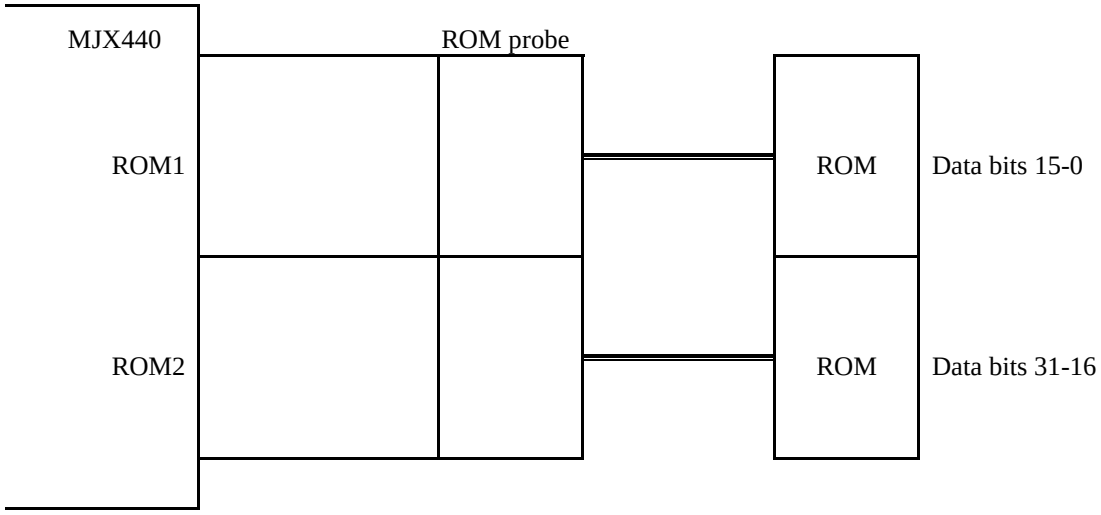


Figure 3-5-9. Connecting a ROM probe (9)

Four 16-bit bus width ROM chips and a 32-bit ROM access bus width:
(LEDs lit: ROM1, ROM2, ROM3, ROM4)

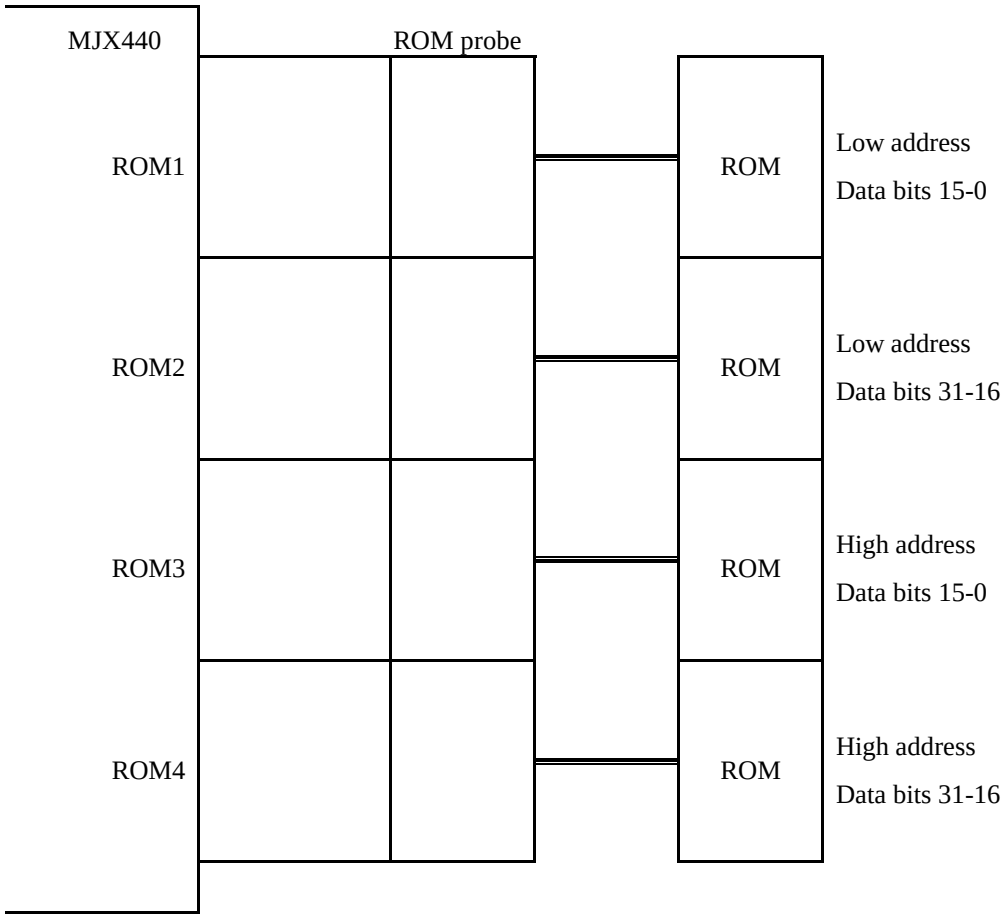


Figure 3-5-10. Connecting a ROM probe (10)

Four 16-bit bus width ROM chips and a 64-bit ROM access bus width:
(LEDs lit: ROM1, ROM2, ROM3, ROM4)

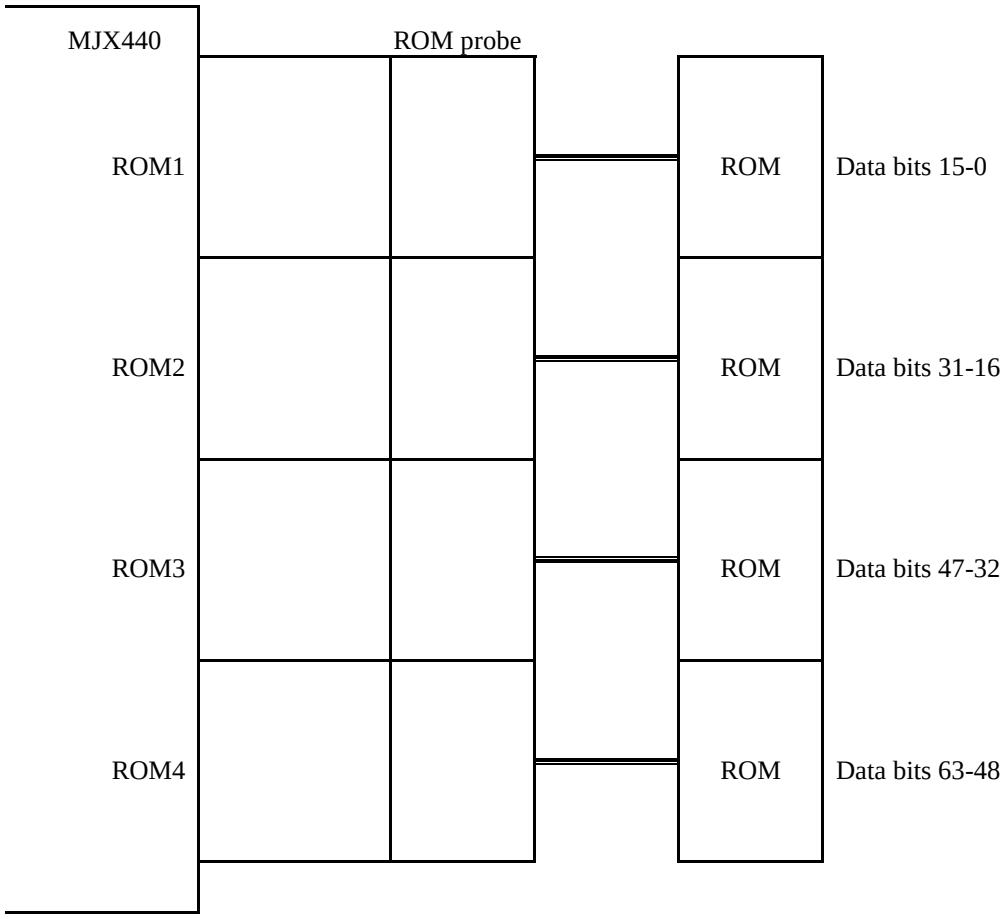


Figure 3-5-11. Connecting a ROM probe (11)

3.4 Connecting External Trigger Cables

【IMPORTANT】The VCC signal of the external trigger cable(1) must be connected to the power line (3.3V) on the target system. The connection of the other signal is optional.

When displaying the status of target system signals on the LED or using MJX440-output trace trigger signals as input into the logic analyzer, use external trigger cables. If these functions are not used, connection of external trigger cables is not necessary.

There are single line type of External trigger cable(1) and dual line type of External trigger cable(2). External trigger cable(1) connects to EXTERNAL connector(1) on EJTAG probe board, and External trigger cable(2) connects to EXTERNAL connector(2) on MJX440 system unit.

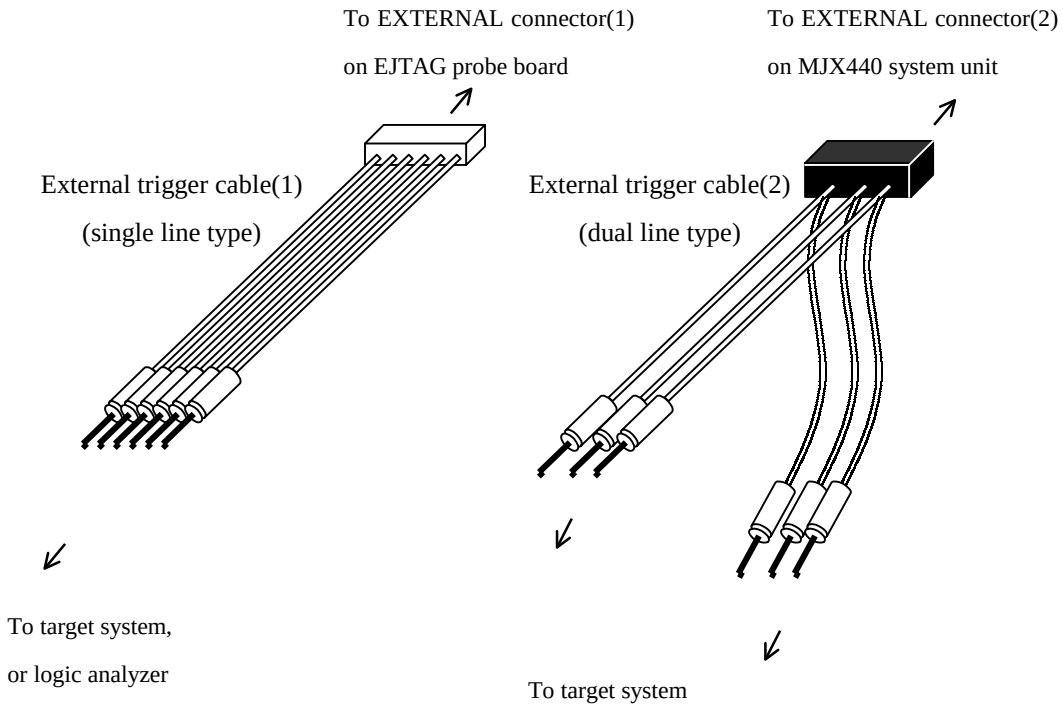


Figure 3-5. Connecting external trigger cables

Details of the signals passing through the external trigger cable are shown below, where signals are color-coded in terms of clip and cable colors:

Signal	Color	I/O	Funtionality
EXTIN1	Clip: yellow Cable: brown	TTL input	For the signals EXTIN1, EXTIN2 and EXTIN3, the signal status is recorded during realtime tracing.
EXTIN2	Clip: yellow Cable: red	TTL input	
EXTIN3	Clip: yellow Cable: orange	TTL input	
TRGOUT-	Clip: green Cable: green	3.3V output	Generates LOW one clock pulse when a trace trigger is passed. This signal can be used as trigger input into the logic analyzer.
VCC	Clip: red Cable: red		This signal is connected to the VCC(3.3V) outlet of the target system.
GND	Clip: black Cable: black		This signal is connected to the GND outlet of the target system.

Table 3-1 External trigger cable(1) (single line type, connects to EJTAG probe board)

Signal	Color	I/O	Funtionality
EXTIN1	Clip: yellow Cable: brown	TTL input	If the connected signal is at the HIGH level, the corresponsing LED is lit.[*1] Signal status can be display by XPIN command of MJX440 command set. Signal status is not recorded during realtime tracing.
EXTIN2	Clip: yellow Cable: red	TTL input	
EXTIN3	Clip: yellow Cable: orange	TTL input	
EXTOUT1	Clip: red Cable: brown	3.3V O.D. output[*2]	Generates either the LOW or HIGH level when asserted by the XPIN command of the MJX440 command set.
EXTOUT2	Clip: red Cable: red	3.3V O.D. output[*2]	
GND	Clip: black Cable: black		This signal is connected to the GND outlet of the target system.

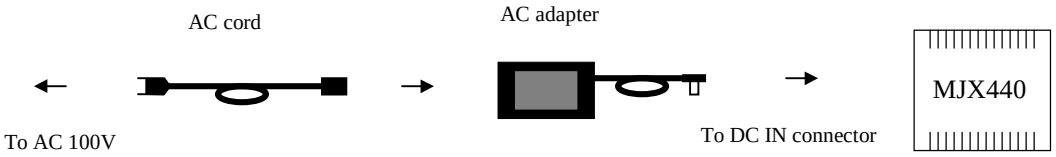
Table 3-2 External trigger cable(2) (dual line type, connects to MJX440 system unit)

[*1] On the MJX440 system unit, signal names are indicated in abbreviation, e.g., EXTIN3→EXI 3.

[*2] "O.D." stands for "open drain".

3.5 Connecting the Power Supply and Turning the Power On

After connecting all cables, connect the AC cord and the AC adapter to the MJX440, making sure that the power switch for the MJX440 is off.



Turn on the devices in the following sequence:

1. Host
2. MJX440
3. Target system

Similarly, turn off the devices in the following sequence:

1. Target sytem
2. MJX440
3. Host

【Important】Turning the power on or off in the incorrect sequence can damage the equipment.

【Important】Do not connect or disconnect any of the devices when turning the power on.

Chapter 4. Installing the Software

This chapter describes how to install the software for the operation of the MJX440.

Please install the software described below:

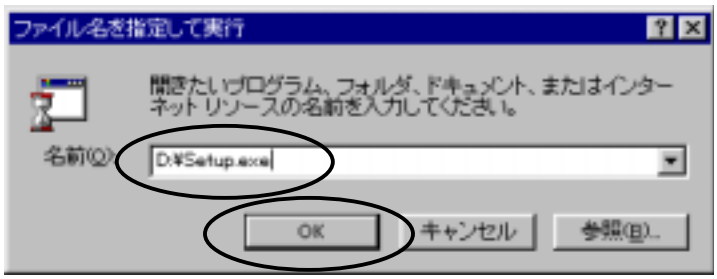
1. Install Green Hills Software's integrated development environment MULTI.

2. Load the CD-ROM labeled "*MJX440 for MIPS/EJTAG Tools*" in the CD-ROM drive.

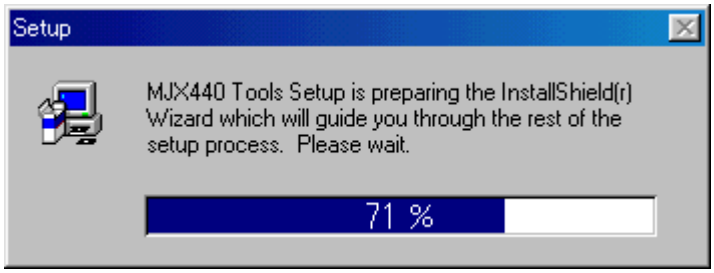
   If the setup program automatically started, skip 3. and 4.

3. Select "**Start**" and "**Run (R) …**" to display the "Run" dialog.

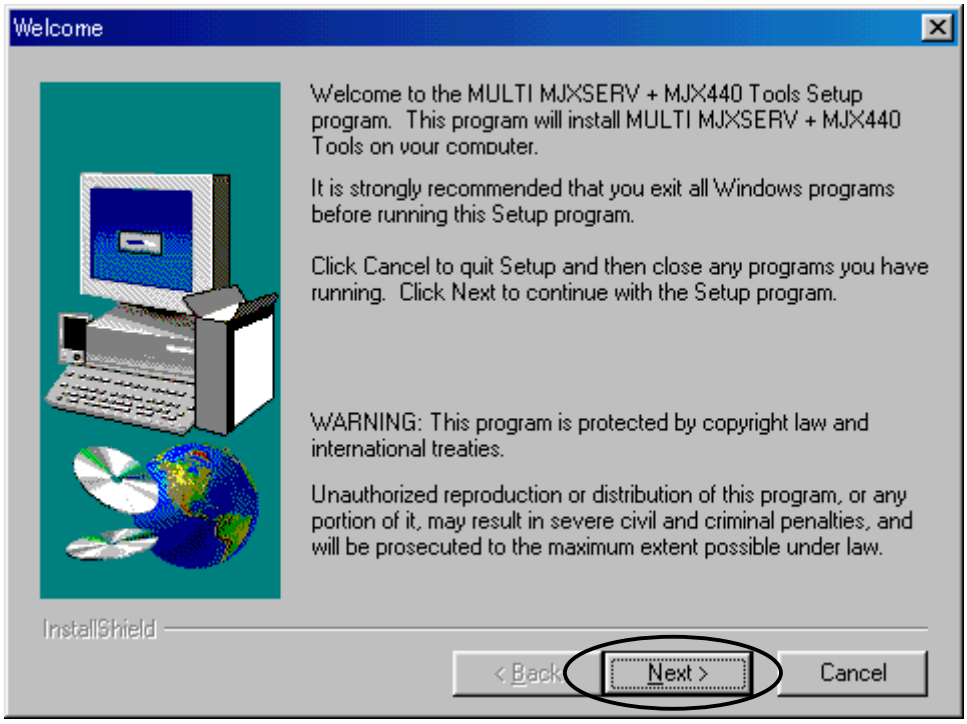
4. In "**Open (O**)", specify the setup program Setup.exe and click on "**OK**". (assuming that your CD-ROM drive is D: drive, specify "**D:¥Setup.exe**").



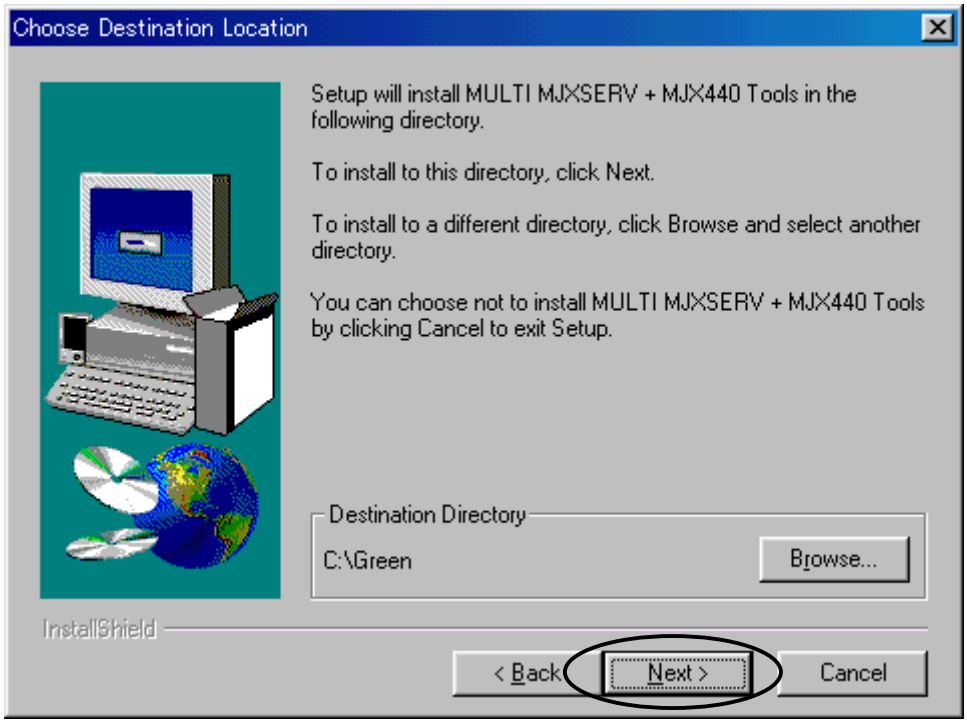
5. The setup program starts.



6. When the "**Welcome**" dialog appears, click on "**Next >**":

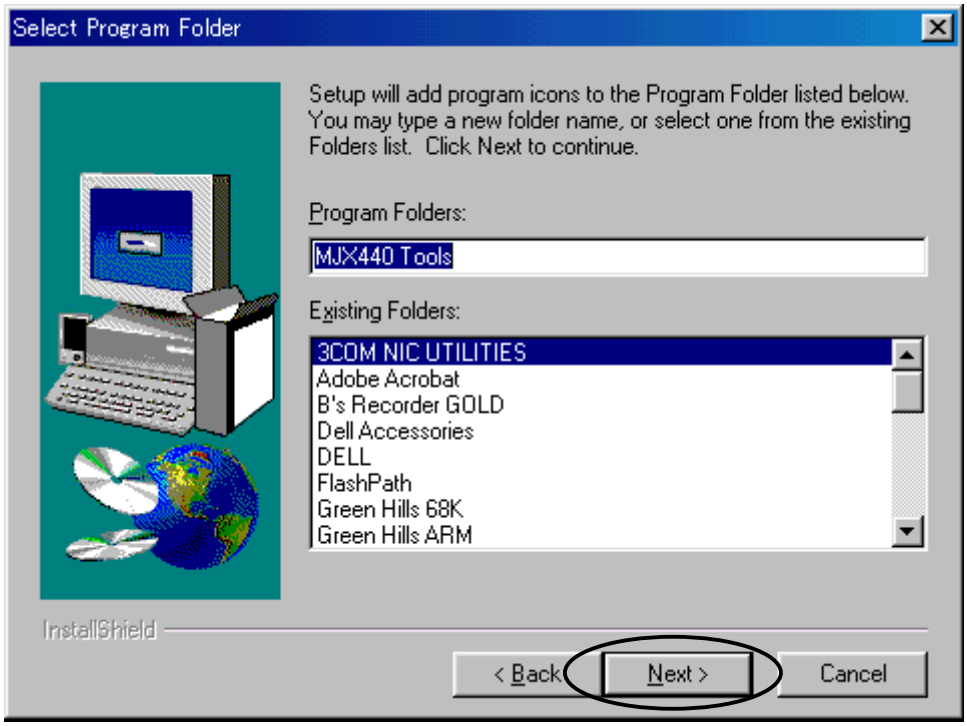


7. When the "**Choose Destination Location**" dialog appears, specify the destination by clicking on the "**Browse...**" button. Be sure to specify the directly in which MULTI was previously installed. (Default directory: "C:¥Green")



Click on "**Next >**".

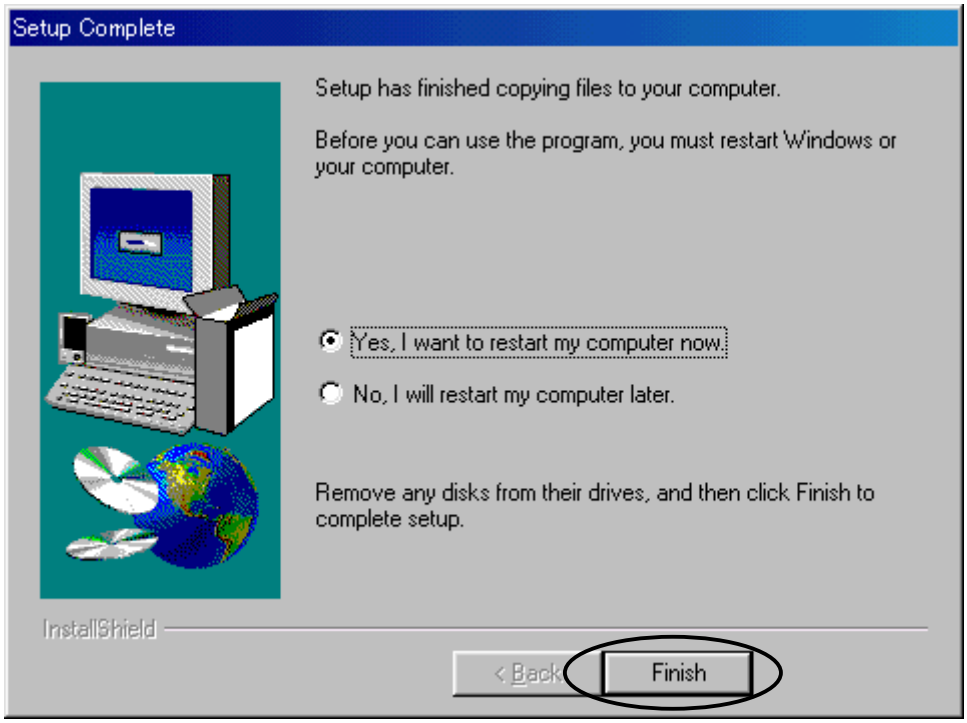
8. When the "**Select Program Folder**" dialog appears, click on the "**Next >**" button.



9. Starting the intallation process.

10. The installation finishes, displaying created program folder.

11. When the "**Setup Complete**" dialog appears, click on the "**Finish**" button to restart the system.



This process installs the following files:

MJX4020.INI	Sample of MJXDEBW conficuration file (CW4020)
MJX4102.INI	Sample of MJXDEBW conficuration file (TR4102)
MJX440.ICO	MJX440 icon file
MJXASM.DLL	assembler/disassembler library
MJXCFG.EXE	Configuration support tool MJXCFG (for creating configuration files)
MJXCFG.CNT	MJXCFG help file
MJXCFG.HLP	〃
MJXCVT.EXE	MJXCVT MJX binary file conversion program MJXCVT
MJXDBAPI.DLL	MJXSERV-MJXDEBW communication library
MJXDEBW.EXE	Quick debugger MJXDEBW
MJXEJTAG.DLL	TR4102/CW4020 library
MJXNB85E.DLL	NB85E library
MJXNB85E.INI	Sample of MJXDEBW conficuration file (NB85E)
MJXSERV.EXE	MULTI server program for MJX440
MJXV831.DLL	V831 library
MJXV831.INI	Sample of MJXDEBW conficuration file (V831)
MJXV832.DLL	V832 library
MJXV832.INI	Sample of MJXDEBW conficuration file (V832)
MSVCP60.DLL	Microsoft Visual C++ runtime library
MSVCRT.DLL	Microsoft Visual C++ runtime library
WNTIX.EXE	Installation program (not used)

The structure of software files is shown below:

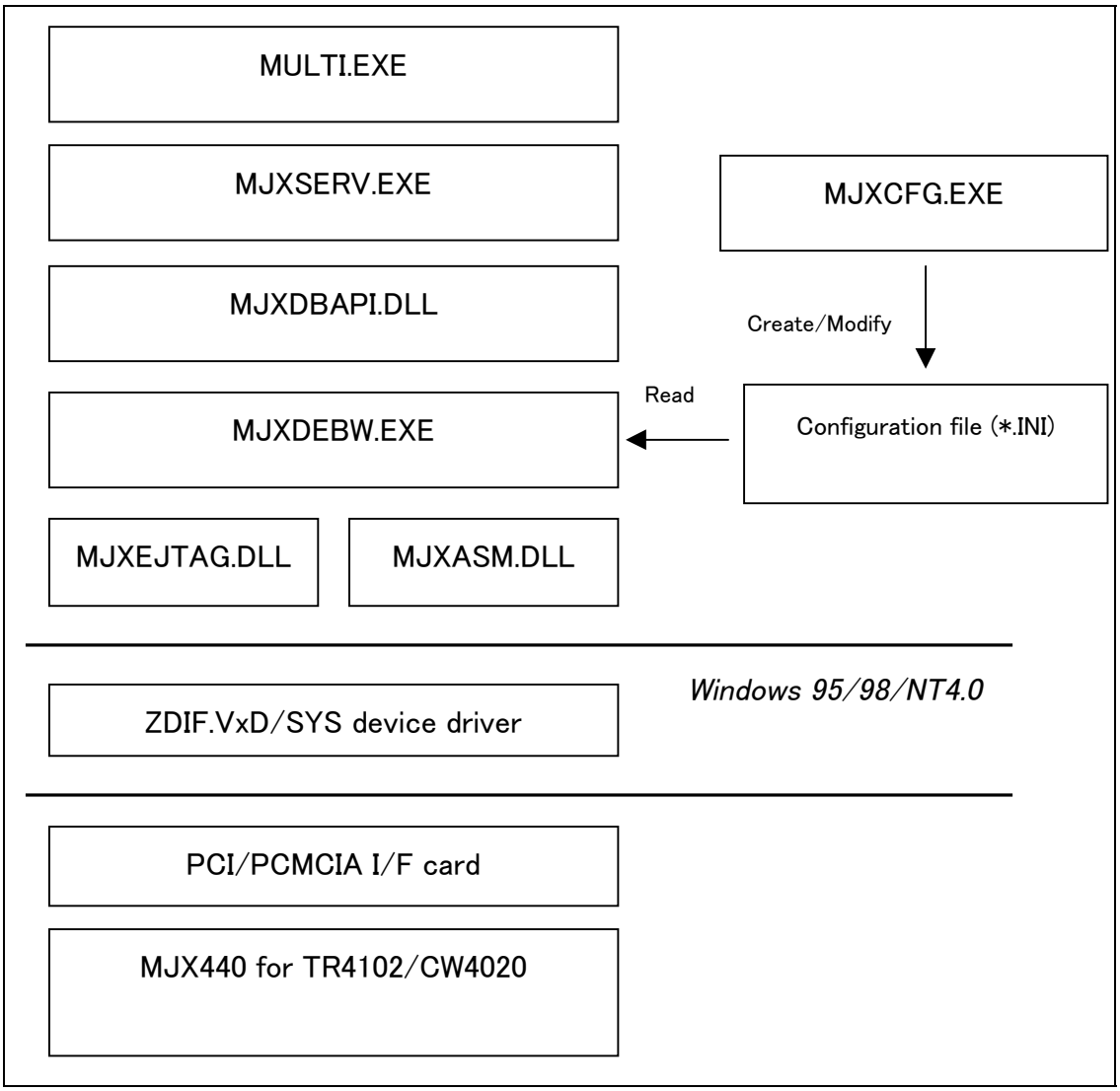


Figure 4-1. Structure of software file

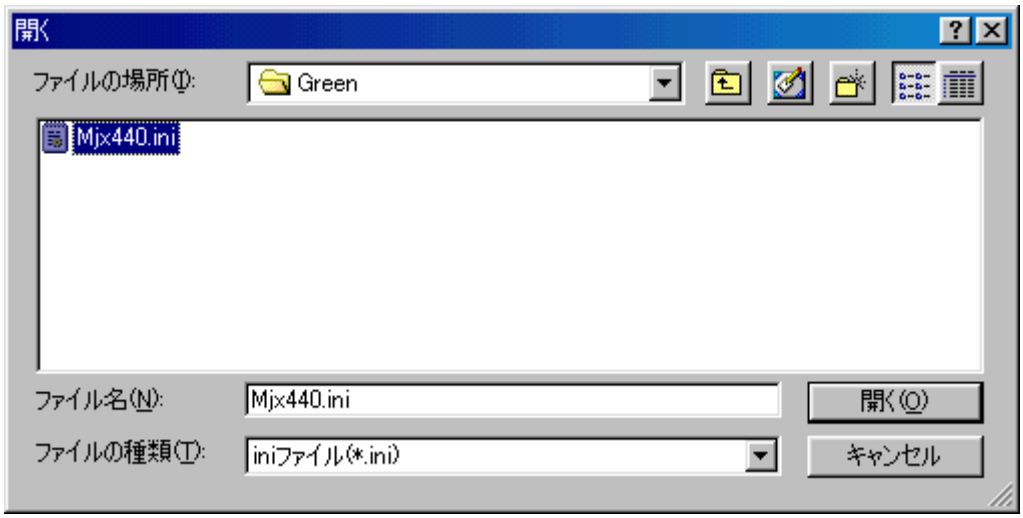
- MULTI.EXE calls MJXSERV.EXE as a child process.
- MJXSERV.EXE calls MJXDEBW.EXE as a child process through MJXDBAPI.DLL.
- MJXDEBW.EXE can work independently, not only a child process.

Chapter 5. Setting the MJX440 Environment

This chapter describes how to set the environment that is necessary for the operation of the MJX440.

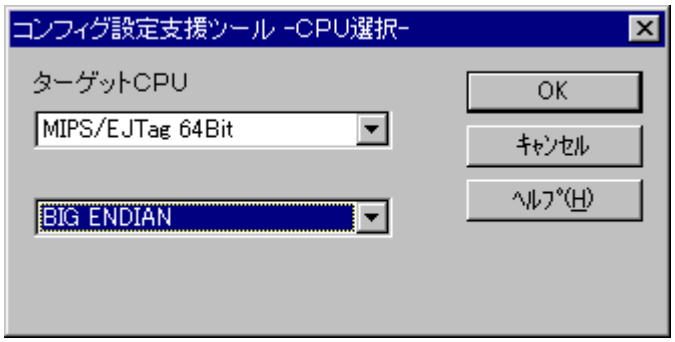
The environment for the MJX440 can be set by using the configuration support tool MJXCFG. After turning on all the devices, launch MJXCFG from the start menu according to the following procedures:

1. Start menu
2. Program (P)
3. MJX440 Tools
4. MJXCFG Configuration Tool



Then, specify the configuration file in which the environmental parameters are to be stored. You should normally specify MJX440.INI.

After configuration file specified, the following diag will be displayed.



Select the target system CPU and endian, the click [OK] button.[*1]

After the target CPU specified, the following diag will be displayed.

When the dialog box appears, set the appropriate fields according to the MJX440 and the target system environment to be used, and then press the [OK] button.

Pressing the [Test] button in the MJX440 tag and the ROM tag button causes the system to test the compatibility between the fields that are set, generating an error message in the event of an error.

[*1] Be sure to select “MIPS/TR4102” or “MIPS/CW4020”.

Chapter 6. Starting and Terminating Software

This chapter describes how to start and terminate software for the operation of the MJX440.

【Note】Before starting software, “Chapter 5. Setting MJX440 Environment” must be completed.

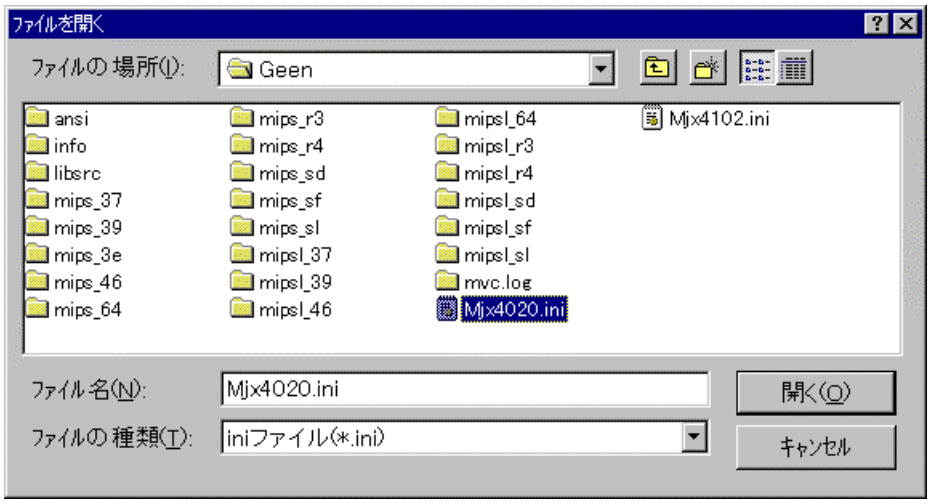
【Note】Only one software program for the operation of the MJX440 can be executed at a time.

MULTI

To operate the MJX440 using MULTI, after starting the MULTI system, use the following command to remote-connect the MJXSERV. [*1]

remote mjxserv

After the above command entered, the following dialog will be displayed for configuration file input. Then, specify the configuration file suit for the target system.[*2]

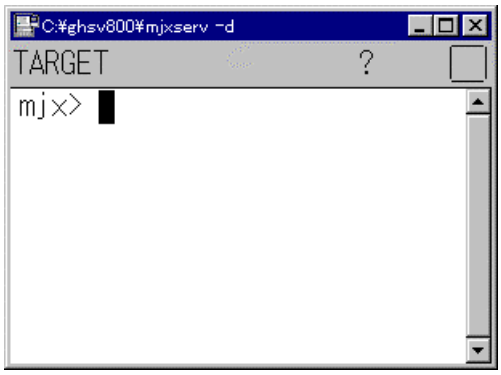


[*1] To operate the MJX440 from a builder, specify the server name mjxserv and press the REMOTE button.

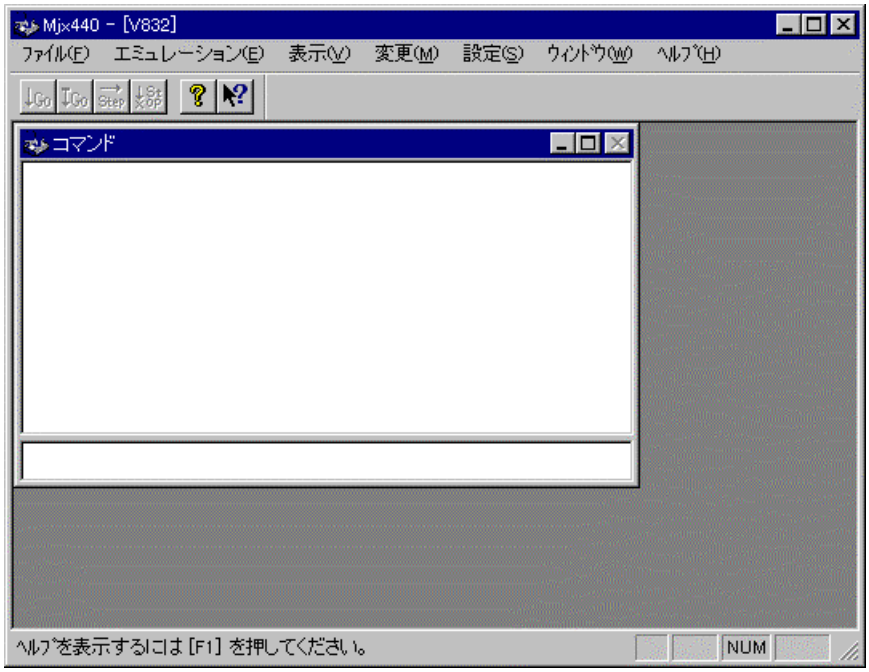
[*2] If taking a long time for specifying configuration file in dialog, MULTI displays a message “Server Message Timed Out, Terminate Connection?”. After this message was displayed, click No button.

When remote-connect is successfully finished, MULTI target window and MJXDEBW window are displayed.

MULTI target window



MJXDEBW window



In this case, the MJX440.INI file is used as a configuration file. Any other configuration file should be specified explicitly. The following command, for example, uses the MJX4020.INI configuration file:

remote mjxserv mjx4020.ini

To terminate the program, enter the *quit* command:

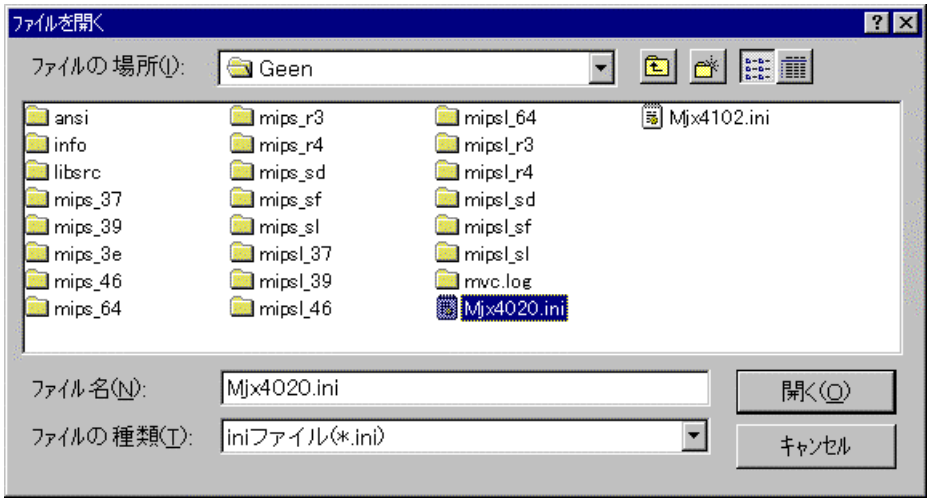
quit

MJXDEBW

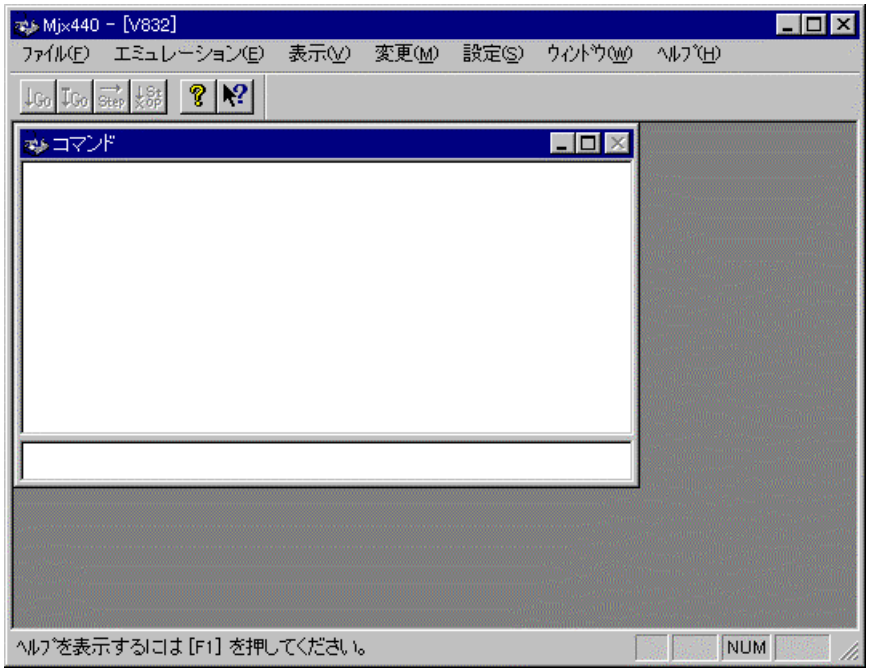
To operate the MJX440 using MJXDEBW, launch MJXDEBW.EXE from the start menu by performing the following sequence of actions:

1. Start menu
2. Program (P)
3. MJX440 Tools
4. MJXDEBW Debugger

After MJXDEBW are started, the following dialog will be displayed for configuration file input. Then, specify the configuration file suit for the target system.



When MJXDEBW is started successfully, MJXDEBW window are displayed.



The configuration file can be specified in the argument of MJXDEBW. The following command, for example, uses the file MJX4020.INI:

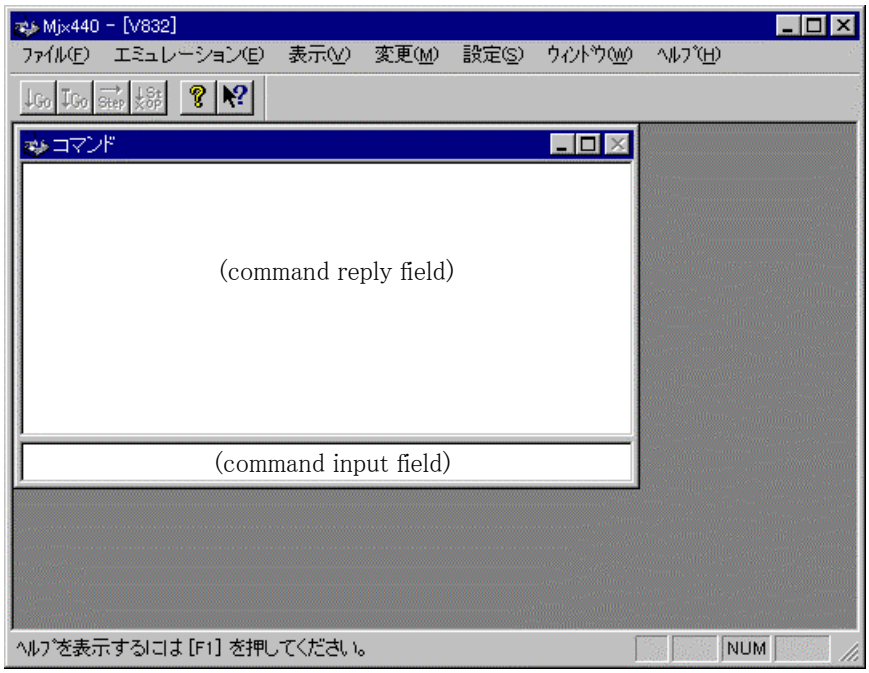
MJXDEBW mjx4020.ini

To terminate the program, simply enter the *quit* command:

quit

Chapter 7.　MJX440 Commands

MJX440 commands can be entered in command input field of MJXDEBW window.



The “Command” sub-window in MJXDEBW window has the following functions.

- The lower part is command input field for command input.
- The upper part is command reply field for displaying command execution result.
- The number of lines in command reply field can be expanded by selecting View menu-Option-View.

Following is a list of available MJX440 commands:

ABORT	Stops a user program.
BATCH	Executes the MJX440 commands that are coded in a batch file.
BP	Displays, sets, and resets breakpoints.
BP/A	Sets data access breakpoints.
BP/H	Sets instruction hardware breakpoints.
BP/S	Sets instruction software breakpoints.
CLEAR	Clear command reply field.
CONFIG	Displays and modifies the MJX440 configuration.
DUMP	Displays memory contents.
EXAMINE	Modifies memory contents.
FILL	Fills memory contents.
GO	Executes a user program.
HISTORY	Displays the result of realtime tracing.
INIT	Reinitializes the MJX440.
JOURNAL	Writes the results of execution of a command to a file.
LOAD	Downloads a file into memory.
MOVE	Block-transfers memory contents.
PIN	Enables/disables a pin.
QUIT	Terminates MJXDEBW.
REGISTER	Displays/modifies the contents of a register.
STEP	Executes a user program in steps.
TRACE/M	Displays/sets the realtime trace mode.
TRACE/I	Displays/sets the realtime trace condition.
UNASM	Disassembles and displays memory contents.
VERSION	Displays software version information.
WAIT	Waits until a user program stops
XPIN	Displays/sets the status of signals serviced by external trigger cables.

Abbreviation of command names

The name of a command can be shortened to any length as long as it remains distinct from all other commands.

AB	This means the same as **ABORT**.
D	This means the same as **DUMP**.
CL	This means the same as **CLEAR**.
CO	This means the same as **CONFIG**.
C	This abbreviation, unable to distinguish between **CLEAR** and **CONFIG**, is illegal.

Notes on using MULTI

The following command is ignored in the MULTI target window:

QUIT	Terminates MJXDEBW.

Whenever possible, the use of the following command in the MULTI target window should be avoided:

BP/S	Sets instruction software breakpoints.
GO	Executes a user program.
STEP	Executes a user program in steps.
REG reg=val	Modifies the contents of a register.

Execution of these commands can compromise the compatibility between MULTI and MJXSERV. Therefore, functions such as controlling the execution of a user program or overwriting a register should be performed using a MULTI command.

Even when modifying register on MULTI, MJXDEBW register are not updated immediately. MJXDEBW register will be updated right before execution of a user program.

ABORT

Stops a user program.

Format:

ABORT	Stops a running user program.

Arguments:

Example:

GO	(Executes a user program.)
ABORT	(Stops a user program)

Remarks:

BATCH

Executes the MJX440 commands that are coded in a batch file.

Format:

BATCH ***file*** Reads the batch file ***file*** line by line and executes the contents of the file as MJX440 commands.

Arguments:

file Name of the batch file in which MJX440 commands are coded (a text file).

Example:

BAT INIT.TXT (Executes the **INIT.TXT** file as a batch file.)

Remarks:

- The **BATCH** command can be nested up to four nesting levels.
- The line beginning with ";" in a batch file is a comment line.
- The **QUIT** in a batch file serves to terminate the **BATCH** command only; it does not terminate MJXDEBW.

BP

Displays, sets, and resets breakpoints.

Format:

BP [A|H|S]	Displays breakpoints.
BP N|AN|HN|SN	Displays numbers of breakpoins。
BP/C *num*|*|A*|H*|S*	Clears breakpoints.
BP/D *num*|*|A*|H*|S*	Disables breakpoints.
BP/E *num*|*|A*|H*|S*	Enables breakpoints.

Arguments:

A	Specifies data access breakpoints.
H	Specifies instruction hardware breakpoints.
S	Specifies instruction software breakpoints.
	When **A**, **H**, or **S** not specified, specifies all breakpoints.
N	Specifies number of all breakpoints.
AN	Specifies number of data access breakpoints.
HN	Specifies number of instruction hardware breakpoints.
SN	Specifies number of instruction software breakpoints.
num	Specifies a breakpoint number. (decimal)
*****	Specifies all breakpoints.
A*	Specifies all data access breakpoints.
H*	Specifies all instruction hardware breakpoints.
S*	Specifies all instruction software breakpoints.

Example:

BP	(Displays all breakpoints)
BP N	(Displays number of all breakpoints)
BP/C S*	(Clear all instruction software breakpoints)
BP/D H*	(Disable all instruction hardware breakpoints)
BP/E A*	(Enable all data access breakpoints)

Remarks:

The breakpoints are displayed in the following format.

```
No. Type ASID Address  AddrMask C/T R/W       Data           DataMask     Enb
  1 S/W   --  11111111    --    --  --         --               --        Enb
  2 INST 0012 00222222 FF000000 --  --         --               --        Dis
  3 ACCS  --  00003333    --     T   W  ________11111111 FFFFFFFF00000000 Enb
  4 PROC  --  00444444    --    --  WC         --        FFFFFFFFFFFF0000 Enb
```

The fields are described below. The field "--" means not specified field.

No. Breakpoint number. (decimal; beginning with 1)

Type Breakpoint type. See below for details.

- **S/W** an instruction software breakpoint.
- **INST** an instruction hardware breakpoint
- **ACCS** a data access breakpoint
- **PROC** a processor bus breakpoint
- **CPLX** a complex breakpoint

ASID Address Space ID of breakpoint address (hexadecimal)

Address Breakpoint address. The physical address is used in processor bus breakpoint. The virtual address is used in the other breakpoint. (hexadecimal)

AddrMask Breakpoint address mask. Disable bit is 1, Enable bit 0. (hexadecimal)

C/T Complex breakpoint, or trace trigger. See below for details.

- **C** used for a part of a complex breakpoint
- **T** used for a trace trigger
- **--** used for a normal breakpoint

R/W Data access. See below for details.
(Case of a data access breakpoint)

R data reading breakpoint

W data writing breakpoint

R/W data access breakpoint[*1]

(Case of a processor bus breakpoint)

R reading instruction/data in uncache area breakpoint

W writing instruction/data in uncache area breakpoint

WC writing instruction/data in cache area breakpoint

(No meaning in case of other breakpoints)

Data Breakpoint data. This field is only valid in case of data access breakpoint or processor bus breakpoint. (hexadecimal)

DataMask Breakpoint data mask. Disable bit is 1, Enable bit 0. (hexadecimal)

Enb Breakpoint enable/disable. See below for details.

Enb Enable

Dis Disable

[*1] data reading or data writing breakpoint

BP/A

Sets data access breakpoints.

Format:

BP/A[/R|/W] [*asid:*]*addr* [*mask*|*!mask*] [,[/B|/W|/L|/D] *value* [*vmask*|*!vmask*]

Sets data access breakpoints.

Arguments:

/R	Specifies data reading breakpoint
/W	Specifies data writing breakpoint
	When /R and /W not specified, specifies both access.
asid	Address Space ID (hexadecimal. Not used when not specified)
addr	Specifies breakpoint address (hexadecimal)
mask	Specifies disable bits for ***addr*** (hexadecimal)
!mask	Specifies enable bits for ***addr*** (hexadecimal)
/B	Specifies 8-bit for ***value***
/W	Specifies 16-bit for ***value***
/L	Specifies 32-bit for ***value***
/D	Specifies 64-bit for ***value***
	When **/B**, **/W**, **/L**, and **/D** not specified, automatically recognized by digit number of ***value***. (two digit = 8-bit)
value	Specifies breakpoint data (hexadecimal)
vmask	Specifies enable bits for ***value*** (hexadecimal)
!vmask	Specifies disable bits for ***value*** (hexadecimal)

Example:

BP/A 1000	(Breaks on data access at address 0x1000.)
BP/A F000 00FF	(Breaks on data access in address 0xF000-0xF0FF.)
BP/A F000 !FF00	(Breaks on data access in address 0xF000-0xF0FF.)
BP/A 10000, /L 22	(Breaks on data access at address 0x10000 with 32-bit data 0x00000022.)
BP/A /R 1000, 22	(Breaks on data reading at address 0x1000 with 8-bit data 0x22)
BP/A /R 1000, 4400 00FF	(Breaks on data reading at address 0x1000 with 16-bit data 0x4400-0x44FF)

BP/A /R 1000, 4400 !FF00

(Breaks on data reading at address 0x1000 with 16-bit data 0x4400-0x44FF)

BP/A /W 1000, 22 (Breaks on data writing at address 0x1000 with 8-bit data 0x22.)

Remarks:

- By using data access breakpoints, the user program can be stopped after reading or writing data at specified address.
- Specify enable bit to 0, disable bit to 1 in ***mask*** and ***vmask***.
- Specify enable bit to 1, disable bit to 0 in ***!mask*** and ***!vmask***.
- If data access breakpoint is set as a trace trigger, maximum number of breakpoins are decreased. (See **TRACE/I** command)
- The maximum number of instruction hardware breakpoins can be displayed by "**BP AN**"command.
- The data access breakpoints ared performed by CPU's Debug Support Unit (DSU) function.

BP/H

Sets instruction hardware breakpoints.

Format:

BP/H [*asid:*]*addr* [*mask*|*!mask*]

Sets instruction hardware breakpoints.

Arguments:

asid	Address Space ID (hexadecimal. Not used when not specified)
addr	Specifies breakpoint address (hexadecimal)
mask	Specifies disable bits for ***addr*** (hexadecimal)
!mask	Specifies enable bits for ***addr*** (hexadecimal)

Example:

BP/H F000	(Breaks before instruction execution at address 0xF000.)
BP/H F000 00FF	(Breaks before instruction execution in address 0xF000-F0FF.)
BP/H F000 !FF00	(Breaks before instruction execution in address 0xF000-F0FF.)

Remarks:

- By using instruction hardware breakpoints, the user program can be stopped before executing instruction on specified address.
- Specify enable bit to 0, disable bit to 1 in ***mask***.
- Specify enable bit to 1, disable bit to 0 in ***!mask***.
- If instruction hardware breakpoint is set as a trace trigger, maximum number of breakpoins are decreased. (See **TRACE/I** command)
- The maximum number of instruction hardware breakpoins can be displayed by "**BP HN**"command.
- The instruction hardware breakpoints ared performed by CPU's Debug Support Unit (DSU) function.

BP/S

Sets instruction software breakpoints.

Format:

BP/S *addr* Sets instruction software breakpoints.

Arguments:

addr Specifies breakpoint address (hexadecimal)

Example:

BP/S F0000000 (Breaks before instruction execution at address 0xF0000000.)

Remarks:

- By using instruction software breakpoints, the user program can be stopped before executing instruction on specified address.
- Instruction software breakpoints are can be set on RAM area, or ROM area which ROM in-circuited.
- A maximum of 128 instruction software breakpoints can be set.

CLEAR

Clear command reply field.

Format:

CLEAR Clear command reply field.

Arguments:

Example:

CLEAR

Remarks:

CONFIG

Displays and modifies the MJX440 configuration.

Format:

CONFIG	Displays the current MJX440 configuration.
CONFIG/S ***item*****=*****value***	Assigns MJX440 configuration item ***item*** to ***value***.

Arguments:

item Specifies one of the following configuration items:

JCLOCK	JTAG clock
TOPADDR	ROM starting address

value Values assigned to configuration items

20 or **40**	JCLOCK (Hz)
Hexadecimal address	TOPADDR

Example:

CONFIG	(Displays MJX440 configuration information.)
CONFIG/S JCLOCK=20	(Sets the JTAG clock at 20MHz.)
CONFIG TOPADDR=20000	(Sets the ROM starting address at 0x20000.)

Remarks:

DUMP

Displays memory contents.

Format:

DUMP[/B|/W|/L||/D] [[*asid:*]*addr1*,[*addr2*]]

Displays memory contents from addresses ***addr1*** through ***addr2***.

Arguments:

/B	Specifies 8 bits.
/W	Specifies 16 bits.
/L	Specifies 32 bits.
/D	Specifies 64 bits.
asid	Address Space ID (hexadecimal. The current value used when not specified)
addr1	Starting memory display address (hexadecimal)
addr2	Ending memory display address (hexadecimal. ***addr1*+3F** used when not specified)

Example:

DUMP/B 1000	(Displays 64-byte momory starting at address 0x1000 in increments of 8 bits.)
DUMP/L 2000,20FF	(Dsiplays memory contents from address 0x2000 through address 0x20FF in increments of 16 bits.)
DUMP	(Displays the continuation of the previous **DUMP** command.)

Remarks:

- Displays 64 bytes of memory contents if ***addr2*** is omitted.
- Displays the continuation of the previous **DUMP** command if ***addr1*** is omitted.
- If a size is omitted, the size specified in the previous command execution is applied.

EXAMINE

Modifies memory contents.

Format:

EXAMINE[/B|/W|/L/|/D] [*asid:*]*addr*

Interactively changes the memory contents beginning at address ***addr***.

EXAMINE[/B|/W|/L/|/D] [*asid:*]*addr*=*data1*[,*data2*…]

Changes the memory contents at address ***addr*** to data ***data1*[,*data2*…]**.

Arguments:

/B	Specifies 8 bits.
/W	Specifies 16 bits.
/L	Specifies 32 bits.
/D	Specifies 64 bits.
asid	Address Space ID (hexadecimal. The current value used when not specified)
addr	Starting memory change address (hexadecimal)

data1 data2

Memory change data (hexadecimal)

Example:

EXAMINE/B 1000=55	(Changes the memory contents at address 0x1000 to 8-bit data 0x55.)
EXAMINE/W 3000=1,2,3	(Changes the memory contents starting from address 0x3000 to 16-bit data 0x0001, 0x0002, and 0x0003.)
EXAMINE/L 2000	(Interactively changes the memory contents starting from address 0x2000.)

00002000 00000000 11223344

00002004 00000000 55667788

00002008 00000000 . (Terminates when a period is encountered.)

Remarks:

- In the interactive mode, this command terminates when a period is entered.
- If a size is omitted, the size specified in the previous command execution is applied.

FILL

Fills memory contents.

Format:

FILL[/B|/W|/L||/D] [*asid:*]*addr1*,{*addr2*},*data*

Fills the memory from address ***addr1*** through address ***addr2*** with data ***data***.

Arguments:

/B	Specifies 8 bits.
/W	Specifies 16 bits.
/L	Specifies 32 bits.
/D	Specifies 64 bits.
asid	Address Space ID (hexadecimal. The current value used when not specified)
addr1	Memory fill starting address (hexadecimal)
addr2	Memory fill ending address (hexadecimal)
data	Fill data (hexadecimal)

Example:

FILL/B 0,3FF,FF	(Fills the memory from 0x0 through 0x3FF with 8-bit data 0xFF.)
FILL/W 1000,1FFF,0	(Fills the memory from 0x1000 through 0x1FFF with 16-bit data 0x000.)

Remarks:

- This command cannot fill more than 8M bytes of memory (a limit item).

GO

Executes a user program.

Format:

GO [[*asid:*]*addr*] Executes a user program from address ***addr***.

Arguments:

asid Address Space ID (hexadecimal. Not used when not specified)

addr User program starting address (hexadecimal)

Example:

GO 1000 (Executes the user program starting from address 0x1000.)

GO (Executes a user program from the current PC.)

Remarks:

- This command should not be executed from within a MULTI target window.
- When the argument ***addr*** is omitted, the command executes the user program from the PC.

HISTORY

Displays the result of realtime tracing.

Format:

HISTORY	Displays a range of packet numbers in which realtime trace results are stored.
HISTORY/D [*start*[,*end*]]	Displays the results of realtime tracing from packet number ***start*** through ***end*** in disassembling.
HISTORY/P [*start*[,*end*]]	Displays the results of realtime tracing from packet number ***start*** through ***end*** in units of packets.

Arguments:

start	Starting display packet number (decimal. The number 0 or continuous number of packet used when not specified)
end	Ending display packet number (decimal)

Example:

HISTORY	(Displays a range of packet numbers in which realtime trace results are stored.)
HISTORY/D 0,70	(Displays the results of realtime tracing from packet numbers 0 through 70 in disassembling.)
HISTORY/P -10,0	(Displays the results of realtime tracing from packet numbers -10 through 0 in units of packets.)

Remarks:

- Use **TRACE** command to set realtime tracing mode.
- Specify ***start*** *and* ***end*** in terms of offset values, for which the packet number 0 is the starting value. Depending on the trace conditions that are employed, the packet number 0 corresponds with the following program location point:

  In case of the following trace conditions, program starting point is packet number 0.

  Begin monitor

  End monitor

  Begin trigger

  In case of the following trace conditions, trace trigger point is packet number 0.

  End trigger

Mid-trigger

Inner trigger

- When realtime trace mode is REALTIME mode, incomplete address may be displayed.

INIT

Reinitializes the MJX440.

Format:

INIT	Reinitializes the MJX440.

Remarks:

- This command also resets the CPU on the target system.

JOURNAL

Writes the results of execution of a command to a file.

Format:

JOURNAL[/A|/W] ***file*****[,***mode*****[,***echo*****]]**

Writes the results of command execution to file ***file***.

JOURNAL/E Terminates file output and closes the file.

Arguments:

/A	Append output specification	
/W	New output specification (default)	
file	Output file name specification	
mode	Output mode specification	
	IN	Writes commands only.
	OUT	Writes command execution results only.
	ALL (default)	Writes both command and command execution results.
echo	Echo mode specification	
	OFF	No screen display of file output
	ON (default)	Screen display of file output

Example:

JOURNAL TEST.TXT	(Writes command execution results to the **TEST.TXT** file.)
JOURNAL/E	(Terminates the file output and closes the file.)
JOURNAL/A TEST.TXT	(Appends command execution results to the **TEST.TXT** file.)

Remarks:

LOAD

Downloads a file into memory.

Format:

LOAD ***file*****[,*****offset*****]**	Downloads into memory file ***file*** of MJX binary[*1], S-record, Intel hex, or COFF format.
LOAD/R ***file*****[,*****offset*****]**	Downloads into memory MJX binary file ***file*** through emulation memory.

Arguments:

file	File to be downloaded
offset	Offset address (default 0)

Example:

LOAD PROG1.ABS	(Downloads the file **PROG1.ABS** into memory.)
LOAD PROG1.ABS,2000	(Downloads the file **PROG1.ABS** into memory with offset address +0x2000.)
LOAD/R PROG2.MJX	(Downloads the file **PROG2.MJX** into memory through emulation memory.)

Remarks:

- The file format is automatically recognized.
- If a file name extension is omitted, the extension **.mjx** is supplied by default.
- The **LOAD/R** command may not be able to download an MJX binary file correctly if the file contains records that point to emulation memory.

[*1] For a description of MJX binary files, see “Chapter 8. Rapid Downloading”.

MOVE

Block-transfers memory contents.

Format:

MOVE[/B|/W|/L|/D] [*asid:*]*addr1*,{*addr2*},[*asid:*]*addr3*

Block-transfers memory contents from addresses ***addr1*** through ***addr2*** to add ress ***addr3***.

Arguments:

/B	Specifies 8 bits
/W	Specifies 16 bits
/L	Specifies 32 bits
/D	Specifies 64 bits
asid	Address Space ID (hexadecimal. The current value used when not specified)
addr1	Source memory starting address (hexadecimal)
addr2	Source memory ending address (hexadecimal)
addr3	Destination memory address (hexadecimal)

Example:

MOVE 1000,10FF,2000 (Block-transfers memory contents from the addresses 0x1000 through 0x10FF to the address 0x2000.)

Remarks:

- This command cannot block transfer memory contents exceeding 8M bytes (a limit item).

PIN

Enables/disables a pin.

Format:

PIN	Displays pin enabled/disabled status.
PIN ***pinname*****=EN|DI**	Enables or disables pin ***pinname***.

Arguments:

pinname (pin name); specify one of the following pins:

INT	(**INT0** throuth **INT5**)
NMI	

EN	Enable
DI	Disable

Example:

PIN	(Displays the pin enabled/disabled status.)
PIN INT=DI	(Disables the pin **INT0** through **INT5**.)

Remarks:

QUIT

Terminates MJXDEBW.

Format:

QUIT	Terminates MJXDEBW.

Remarks:

- The **QUIT** command in a batch file terminates the BATCH command only; it does not terminate MJXDEBW.

REGISTER

Displays/modifies the contents of a register.

Format:

RESISTER	Displays the contents of all registers.
RESISTER/GP	Displays the contents of all general registers.
RESISTER/CP0	Displays the contents of all coprocessor 0 registers.
RESISTER/F	Displays the contents of all floating-point registers.
RESISTER ***reg***	Displays the contents of register ***reg***.
RESISTER/GP ***num***	Displays the contents of a general register numbered ***num***.
RESISTER/CP0 ***num***	Displays the contents of a coprocessor register numbered ***num***.
RESISTER/F ***num***	Displays the contents of a floating-point register numbered ***num***.
RESISTER ***reg=data***	Changes the contents of register ***reg*** to data ***data***.
RESISTER/GP ***num=data***	Changes the contents of general register numbered ***num*** to data ***data***.
RESISTER/CP0 ***num=data***	Changes the contents of coprocessor 0 register numbered ***num*** to data ***data***.
RESISTER/T [*tlbnum*]	Displays TLB entries.

Arguments:

reg	Register name specification (See “Appendix G. List of Register Names”.)
num	Register number specification (decimal. 0 throuth 31)
data	Register change data (hexadecimal)
tlbnum	TLB entry number (decimal. 0 through max TLB entry number. All entry used when not specified)

Example:

REGISTER	(Displays the contents of all registers.)
REGISTER R1	(Displays the contents of register R1.)
REGISTER PC=2000	(Changes the contents of register PC to data 0x2000.)
REGISTER/T	(Displays all TLB entries.)

Remarks:

STEP

Executes a user program in steps.

Format:

STEP [*num*] Executes a user program by *num* steps.

Arguments:

num Step count specification (decimal; default: 1)

Example:

STEP (Executes a user program by 1 step.)

STEP 10 (Executes a user program by 10 steps)

Remarks:

TRACE/M

Displays/sets the realtime trace mode.

Format:

TRACE/M FL	Set trace mode to FULL mode.
TRACE/M RT	Set trace mode to REALTIME mode.
TRACE	Displays trace mode and all trace conditions.

Arguments:

Example:

TRACE	(Displays trace mode)
TRACE/M FL	(Sets trace mode to FULL mode)

Remarks:

- The defference between FULL mode and REALTIME mode described below.

FULL mode	Stores all branch instructions information in the realtime trace memory	If many branch instructions are executed in short time, the user program may execute slowly.
REALTIME mode	Stores branch instructions information in the realtime trace memory as many as possible.	Even if many branch instructions are executed in short time, the user program can execute in realtime. Some branch instructions information may be lacked in the realtime trace memory, so the realtime tracing may not be completed.

TRACE/I

Displays/sets the realtime trace condition.

Format:

TRACE/I BM[,*length*]	Sets execution trace conditions on the begin monitor.
TRACE/I EM	Sets execution trace conditions on the end monitor.
TRACE/I BT,*num*	Sets execution trace conditions on the begin trigger.
TRACE/I ET,*num*	Sets execution trace conditions on the end trigger.
TRACE/I MT,*length*,*num*	Sets execution trace conditions on the mid trigger.
TRACE/R	Disables all trace conditions.
TRACE	Displays trace mode and all trace conditions.

Arguments:

length	Tracing word length specification from trace trigger (decimal; default: 524288)
num	Breakpoint number which is used as trace trigger (decimal)

Example:

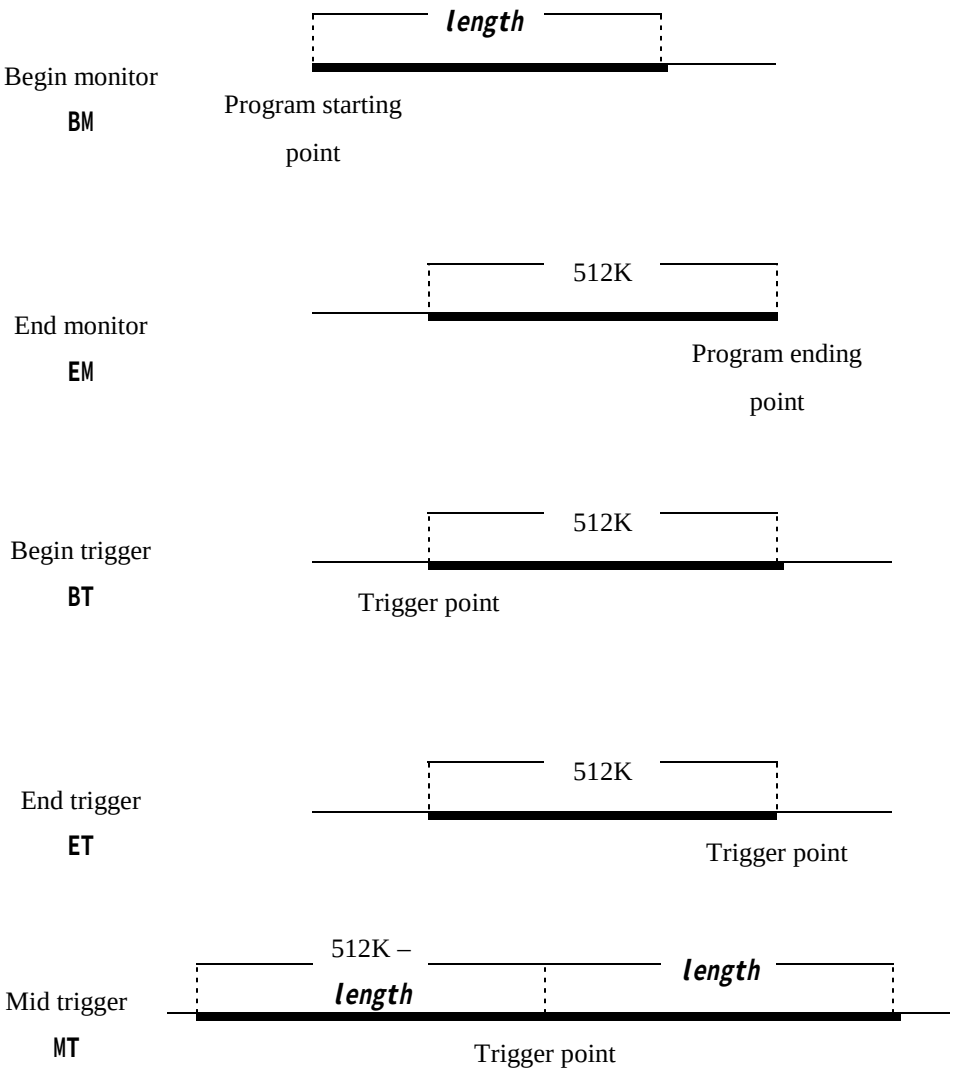
TRACE/I BM	(Sets execution trace conditions on the begin monitor.)
TRACE/I BT,1	(Sets execution trace conditions on the begin trigger, and assigns trace trigger to breakpoint number 1.)
TRACE/I MT,2048,3	(Sets execution trace conditions on the mid trigger, assigns trace trigger to breakpoint number 3, and sets tracing length from trace trigger to 2048.)
TRACE	(Disables all trace conditions.)
TRACE/R	(Disables all trace conditions.)

Remarks:

- The following two kinds of breakpoint can be used as trace trigger. If they assigned to trace trigger, breakpoint function does not work.

  Data access breakpoints (**BP/A**)
  Instruction hardware breakpoints (**BP/H**)

Tracing range:



UNASM

Disassembles and displays memory contents.

Format:

UNASM [[*asid:*]*addr*[,*count*]]

Displays the memory contents including ***count*** instructions starting from address ***addr***, in the disassembled form.

Arguments:

asid	Address Space ID (hexadecimal. The current value used when not specified)
addr	Disassembled display memory starting address (hexadecimal)
count	Number of disassembled instructions for display (decimal; default: 16)

Example:

UNASM 1000	(Displays the memory contents including 16 instructions starting from the address 0x1000 in the disassembled form.)
UNASM	(Displays the continuation of the previous **UNASM** command.)

Remarks:

- Displays memory contents including 16 instructions in the disassembled form when ***count*** is omitted.
- Displays the continuation of the previous **UNASM** command when ***addr*** is omitted.

VERSION

Displays software version information.

Format:

VERSION Displays the following version information.

MJXDEBW	MJXDEBW.EXE software
DLL	DLL library depending on CPU
DRIVER	device driver
MJX440	MJX440 hardware
CPU	CPU Debug Support Unit (DSU)

Example:

VERSION

Remarks:

WAIT

Waits until a user program stops

Format:

WAIT [*time*]	Waits until a user program stops. If a user program does not stop in specified time period, MJX440 stops a user program.

Arguments:

time Specifies waiting time (decimal. Millisecond. The ∞ used when not specified)

Example:

GO	(Executes a user program.)
WAIT 1000	(Waits until a user program stops. If a user program does not stop in one second, MJX440 stops a user program)

Remarks:

- The waiting time can be specified by millisecond, but it is not so accurate.

XPIN

Displays/sets the status of signals serviced by external trigger cables.

Format:

XPIN	Displays the status of signals serviced by an external trigger cable.
XPIN ***ch*****,*****level***	Sets output signal ***ch*** from the external trigger cable to ***level***.

Arguments:

ch	Pin specification for external trigger cable output signals	
	1	EXTOUT1
	2	EXTOUT2
level	Output pint level specification	
	0	LOW level
	1	HIGH level

Example:

XPIN	(Displays the status of signals serviced by an external trigger cable.)
XPIN 1,0	(Sets the external trigger cable output signal EXTOUT1 to the LOW level.)

Remarks:

- The output signals are negative logic.

Chapter 8. Rapid Downloading

This chapter describes the procedures for effecting rapid downloading.

By creaging an MJX binary file, you can rapidly download programs at the following rates:

- EJTAG cable connection: 440Kbytes/second (RAM region)
- ROM in-circuit connection: 4Mbytes/second (ROM region)

MJX binary files can be created by using the file conversion program MJXCVT, which converts S-records into an MJX binary file. A description of how to use this program, executed from the MS-DOS prompt, follows:

Using MJXCVT

```
mjxcvt [-o offset] infile [outfile]
```

-o offset	Add the offset address to the output file.
infile	Input file name
outfile	Output file name (default: ***infile*** with the extension .mjx)

Downloading an MJX binary file

The MJX binary file created by MJXCVT can be downloaded by issuing the LOAD command in the MJX440 command set:

```
load myfile.mjx
```

Appendix A. Specifications

System unit dimensions	43mm(H) x 149mm(W) x 186mm(D)	
Weight	1100g	
Power supply (AC adapter)	Input: AC100～240V 50Hz/60Hz	
	Output: DC 5V±5% 3.0A	
ROM probe cable	300mm	
External trigger cable	300mm	
Operating temperature range	0℃～35℃	
Storage temperature range	-10℃～55℃	
Ambient humidity range	30%～85%	
Compatible ROM	See “Appendix D. ROM Probes”.	
Number of ROM chips	1, 2, 4, and 8 8-bit chips; 1, 2, and 4 16-bit chips	
Emulation memory size	16Mbytes	
Access time	50nsec from CS	
Interface	Parallel (PCI or PCMCIA card)	
Target interface	EJTAG connector	
	ROM socket	
Supported CPU	TR4102/CW4020	
Downloading rate	440Kbytes/sec (EJTAG)、	
	4Mbytes/sec (ROM in-circuit connection)	
Supported debugger	Green Hills MULTI	
Breakpoint function	Instruction fetch	× 2
	Software	× 100
	Memory access	× 4
Trace function	Trace memory:	48bit × 128K
	Trace clock:	125MHz max (depends on CPU)
	Trace condition:	6 sets
	Time stam:	32bit
Limits	See “Appendix B. Limits on Target Systems”.	

Appendix B. Limits on Target Systems

In order to use the MJX440, be sure that your target system meets the following conditions:

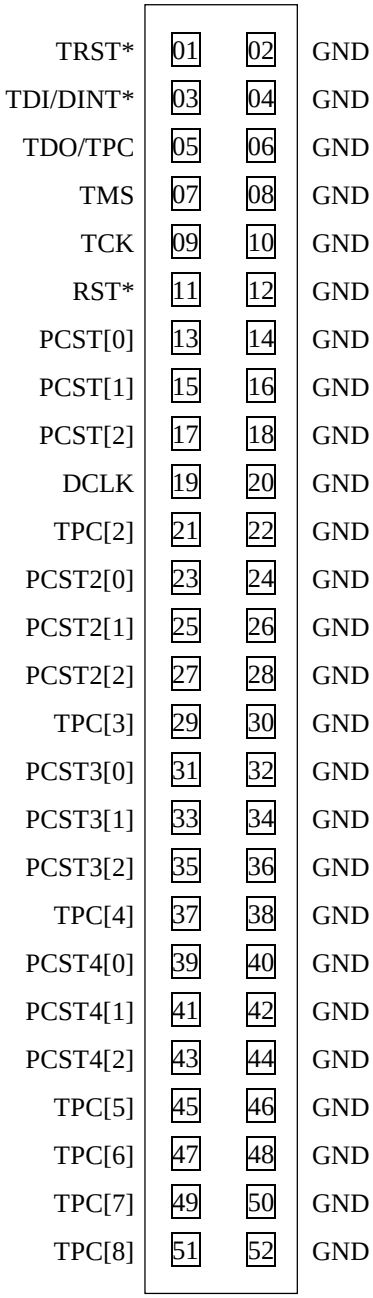
- The target system is equipped with connectors in compliance with 52pin Extended EJTAG specifications. (See “Appendix C. EJTAG Connector”)
- The VCC signal of External trigger cable(1) can be connected to the power line on the target system.

In order to use a target system with a ROM in-circuit connection, the target system must meet the following conditions:

- The target system is equipped with a ROM socket.
- The target system contains ROM in a bank configuration.
- If multiple ROM chips are installed, all ROM address signals in the target system are identical, or ROM starting addresses are continuous.

Appendix C. EJTAG Connector

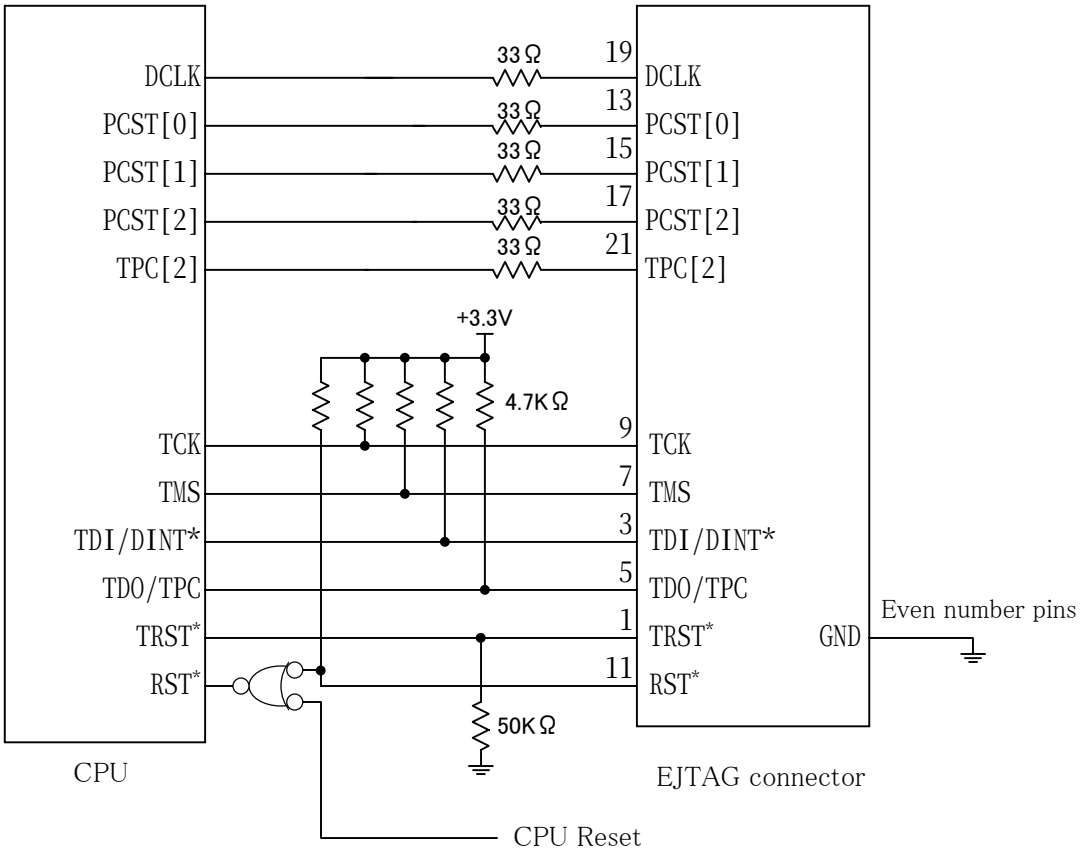
Pin assignment



Recommended connector

- PRECIDIP Corporation, 852-10-052-10-001 (straight)
- PRECIDIP Corporation, 852-10-052-20-001 (right angle)
- PRECIDIP Corporation, 852-10-052-30-001 (straight, surface mount)

Recommended circuitry of EJTAG connector in the target system



Notes

- The patterns between the CPU and the EJTAG connector should be as short as possible (less than 50 mm).
- The patterns between TCLK～TCK and between DCLK～DCLK should be GND-shielded.
- Not all pins are drawn in the above circuitry. The missing PCST and TPC pins are the same as drawn in the above circuitry.
- The EJTAG connector has no VCC pin. So, VCC signal of External trigger cable(1) should be connected to VCC(3.3V) in the target system. It detects status of target system power.

Appendix D. ROM Probes

ROM probe board J-101A jumper settings

ROM plug	Jumper settings	ROM size	Supported ROM	ROM maker
32pin (27010)	JMP1 1-2 shorted JMP2 1-2 shorted	128K x 8bit (0x20000byte)	HN27C101AG μPD27C1001AD TC571000D TC571000AD TC57H1000AD M5M27C101K MBM27C1001-nnZ 27010 27C010 Am27C010	Hitachi NEC Toshiba Toshiba Toshiba Mitsubishi Fujitsu intel intel AMD
32pin (27020)	JMP1 1-2 shorted JMP2 1-2 shorted	256K x 8bit (0x40000byte)	μPD27C2001D M5M27C201K Am27C020	NEC Mitsubishi AMD
32pin (27040)	JMP1 1-2 shorted JMP2 1-2 shorted	512K x 8bit (0x80000byte)	HN27C4001G μPD27C4001DZ TC574000D TC574000DI M5M27C401K MBM27C4001-nnZ 27040 Am27C040	Hitachi NEC Toshiba Toshiba Mitsubishi Fujitsu intel AMD
32pin (27080)	JMP1 1-2 shorted JMP2 1-2 shorted	1024K x 8bit (0x100000byte)	Am27C080	AMD
32pin (271000)	JMP1 2-3 shorted JMP2 1-2 shorted	128K x 8bit	HN27C301AG μPD27C1000AD TC571001D TC571001AD TC57H1001AD M5M27C100K MBM27C1000-nnZ	Hitachi NEC Toshiba Toshiba Toshiba Mitsubishi Fujitsu

ROM probe board J-102A jumper settings

ROM plug	Jumper settings	ROM size	Supported ROM	ROM maker
40pin (27C4000 16bit)	JMP1 1-2 shorted	256K x 16bit (0x80000byte)	HN27C4000G Am27C400	Hitachi AMD
42pin (27C8000 16bit)	JMP1 1-2 shorted	512K x 16bit (0x100000byte)	μPD27C8000 Am27C800	NEC AMD
42pin (27C16000 16bit)	JMP1 1-2 shorted	1024K x 16bit (0x200000byte)		

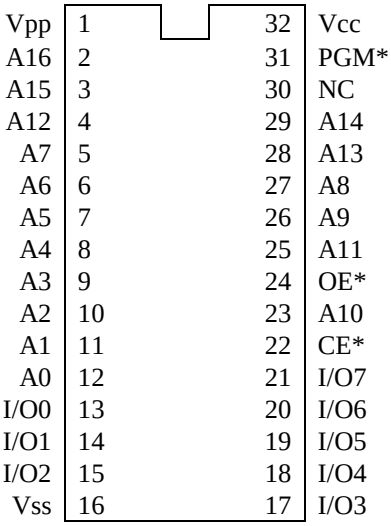
ROM probe board J-103A jumper settings

ROM plug	Jumper settings	ROM size	Supported ROM	ROM maker
40pin (271024)	JMP1 1-2 shorted	64K x 16bit (0x20000byte)	HN27C1024HG μPD27C1024D μPD27C1024AD TC57H1024D TC57H1024AD MBM27C1024-nnZ 27210 27C210 Am27C1024	Hitachi NEC NEC Toshiba Toshiba Fujitsu intel intel AMD
40pin (272048)	JMP1 1-2 shorted	128K x 16bit (0x40000byte)	Am27C2048	AMD
40pin (274096)	JMP1 1-2 shorted	256K x 16bit (0x80000byte)	HN27C4096G HN27C4096HG HN27C4096AG HN27C4096AHG TC574096D MBM27C4096-nnZ 27240 Am27C4096	Hitachi Hitachi Hitachi Hitachi Toshiba Fujitsu intel AMD

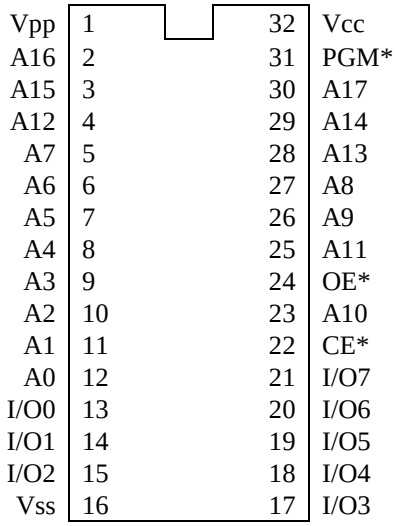
ROM probe board J-104A jumper settings

ROM maker	Jumper settings	ROM size	Supported ROM	ROM maker
40pin (27C4000 8bit)	JMP1 1-2 shorted	512K x 8bit (0x80000byte)	HN27C4000G Am27C400	Hitachi AMD
42pin (27C8000 8bit)	JMP1 1-2 shorted	1024K x 8bit (0x100000byte)	μ PD27C8000 Am27C800	NEC AMD
42pin (27C16000 8bit)	JMP1 1-2 shorted	2048K x 8bit (0x200000byte)		

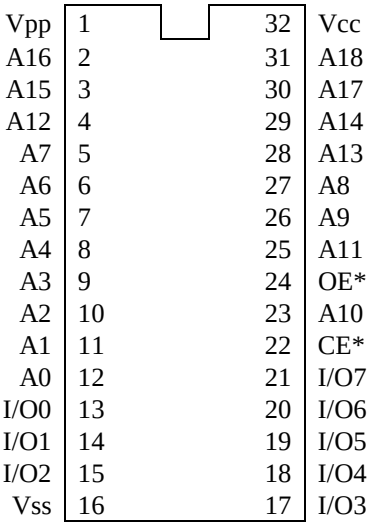
Appendix E. Corresponding ROM Pin Assignment



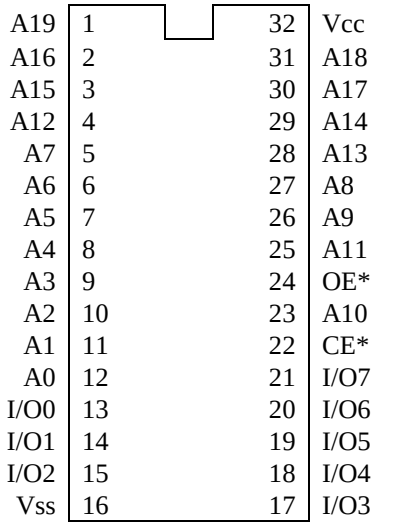
J-101A 27010



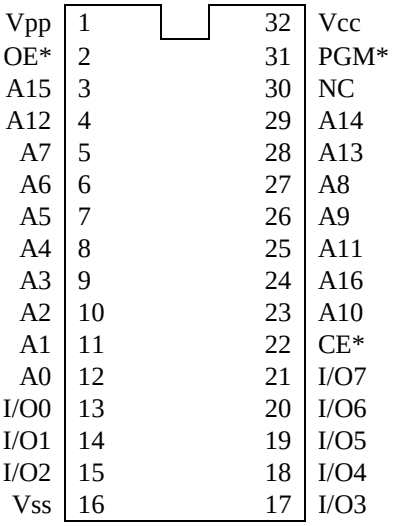
J-101A 27020



J-101A 27040



J-101A 27080



J-101A 271000

A17	1	40	A8
A7	2	39	A9
A6	3	38	A10
A5	4	37	A11
A4	5	36	A12
A3	6	35	A13
A2	7	34	A14
A1	8	33	A15
A0	9	32	A16
CE*	10	31	BYTE*/Vpp
Vss	11	30	Vss
OE*	12	29	I/O15/A-1
I/O0	13	28	I/O7
I/O8	14	27	I/O14
I/O1	15	26	I/O6
I/O9	16	25	I/O13
I/O2	17	24	I/O5
I/O10	18	23	I/O12
I/O3	19	22	I/O4
I/O11	20	21	Vcc

J-102A/J-104A
27C4000

A18	1	42	NC
A17	2	41	A8
A7	3	40	A9
A6	4	39	A10
A5	5	38	A11
A4	6	37	A12
A3	7	36	A13
A2	8	35	A14
A1	9	34	A15
A0	10	33	A16
CE*	11	32	BYTE*/Vpp
Vss	12	31	Vss
OE*	13	30	I/O15/A-1
I/O0	14	29	I/O7
I/O8	15	28	I/O14
I/O1	16	27	I/O6
I/O9	17	26	I/O13
I/O2	18	25	I/O5
I/O10	19	24	I/O12
I/O3	20	23	I/O4
I/O11	21	22	Vcc

J-102A/J-104A
27C8000

A18	1	42	A19
A17	2	41	A8
A7	3	40	A9
A6	4	39	A10
A5	5	38	A11
A4	6	37	A12
A3	7	36	A13
A2	8	35	A14
A1	9	34	A15
A0	10	33	A16
CE*	11	32	BYTE*/Vpp
Vss	12	31	Vss
OE*	13	30	I/O15/A-1
I/O0	14	29	I/O7
I/O8	15	28	I/O14
I/O1	16	27	I/O6
I/O9	17	26	I/O13
I/O2	18	25	I/O5
I/O10	19	24	I/O12
I/O3	20	23	I/O4
I/O11	21	22	Vcc

J-102A/J-104A
27C16000

Vpp	1	40	Vcc
CE*	2	39	PGM*
I/O15	3	38	NC
I/O14	4	37	A15
I/O13	5	36	A14
I/O12	6	35	A13
I/O11	7	34	A12
I/O10	8	33	A11
I/O9	9	32	A10
I/O8	10	31	A9
Vss	11	30	Vss
I/O7	12	29	A8
I/O6	13	28	A7
I/O5	14	27	A6
I/O4	15	26	A5
I/O3	16	25	A4
I/O2	17	24	A3
I/O1	18	23	A2
I/O0	19	22	A1
OE*	20	21	A0

J-103A 271024

Vpp	1	40	Vcc
CE*	2	39	PGM*
I/O15	3	38	A16
I/O14	4	37	A15
I/O13	5	36	A14
I/O12	6	35	A13
I/O11	7	34	A12
I/O10	8	33	A11
I/O9	9	32	A10
I/O8	10	31	A9
Vss	11	30	Vss
I/O7	12	29	A8
I/O6	13	28	A7
I/O5	14	27	A6
I/O4	15	26	A5
I/O3	16	25	A4
I/O2	17	24	A3
I/O1	18	23	A2
I/O0	19	22	A1
OE*	20	21	A0

J-103A 272048

Vpp	1	40	Vcc
CE*	2	39	A17
I/O15	3	38	A16
I/O14	4	37	A15
I/O13	5	36	A14
I/O12	6	35	A13
I/O11	7	34	A12
I/O10	8	33	A11
I/O9	9	32	A10
I/O8	10	31	A9
Vss	11	30	Vss
I/O7	12	29	A8
I/O6	13	28	A7
I/O5	14	27	A6
I/O4	15	26	A5
I/O3	16	25	A4
I/O2	17	24	A3
I/O1	18	23	A2
I/O0	19	22	A1
OE*	20	21	A0

J-103A 274096

Appendix F.　LEDs

PWR (緑)	Lit when power is on
EJTAG (赤)	Lit when externl trigger cable(1) VCC signal is at HIGH level
EXI1 (赤)	Lit when externl trigger cable(2) EXTIN1 signal is at HIGH level
EXI2 (赤)	Lit when externl trigger cable(2) EXTIN2 signal is at HIGH level
EXI3 (赤)	Lit when externl trigger cable(2) EXTIN3 signal is at HIGH level
ROM1 (赤)	See Figures 3-5-1 through 3-5-11.
ROM2 (赤)	Same as the above
ROM3 (赤)	Same as the above
ROM4 (赤)	Same as the above

Appendix G. List of Register Names

(general registers)

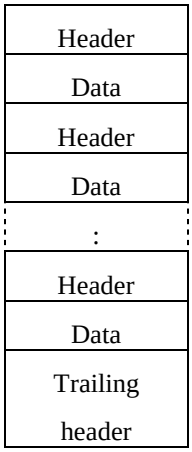
R0	CP0_0	PC
R1	CP0_1	HI
R2	CP0_2	LO
R3	CP0_3	
R4	CP0_4	
R5	CP0_5	
R6	CP0_6	
R7	CP0_7	
R8	CP0_8	
R9	CP0_9	
R10	CP0_10	
R11	CP0_11	
R12	CP0_12	
R13	CP0_13	
R14	CP0_14	
R15	CP0_15	
R16	CP0_16	
R17	CP0_17	
R18	CP0_18	
R19	CP0_19	
R20	CP0_20	
R21	CP0_21	
R22	CP0_22	
R23	CP0_23	
R24	CP0_24	
R25	CP0_25	
R26	CP0_26	
R27	CP0_27	
R28	CP0_28	
R29	CP0_29	
R30	CP0_30	
R31	CP0_31	

(floating-point registers)

FGR0	FPR0	FCR0
FGR1	FPR1	FCR31
FGR2	FPR2	
FGR3	FPR3	
FGR4	FPR4	
FGR5	FPR5	
FGR6	FPR6	
FGR7	FPR7	
FGR8	FPR8	
FGR9	FPR9	
FGR10	FPR10	
FGR11	FPR11	
FGR12	FPR12	
FGR13	FPR13	
FGR14	FPR14	
FGR15	FPR15	
FGR16	FPR16	
FGR17	FPR17	
FGR18	FPR18	
FGR19	FPR19	
FGR20	FPR20	
FGR21	FPR21	
FGR22	FPR22	
FGR23	FPR23	
FGR24	FPR24	
FGR25	FPR25	
FGR26	FPR26	
FGR27	FPR27	
FGR28	FPR28	
FGR29	FPR29	
FGR30	FPR30	
FGR31	FPR31	

Appendix H. MJX Binary File

Binary file organization



Header (16 bytes)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
‘M’	‘J’	‘1’	00	len1	len2	len3	00	00	00	00	adr1	adr2	adr3	adr4	00

len1: data byte length (MSB)

len2: data byte length

len3: data byte length (LSB)

adr1: logical address (MSB)

adr2: logical address

adr3: logical address

adr4: logical address (LSB)

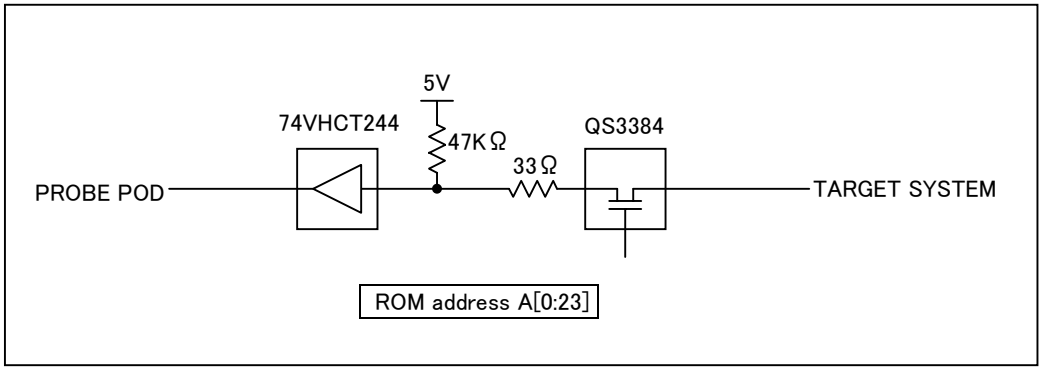
Data (varable data byte length)

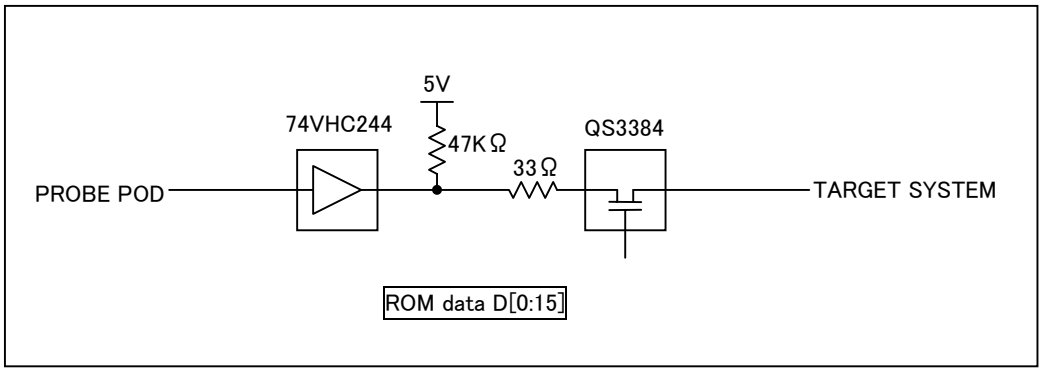
XX	XX	XX	XX	XX	XX	XX	…	XX	XX	XX	XX	XX	XX	XX	

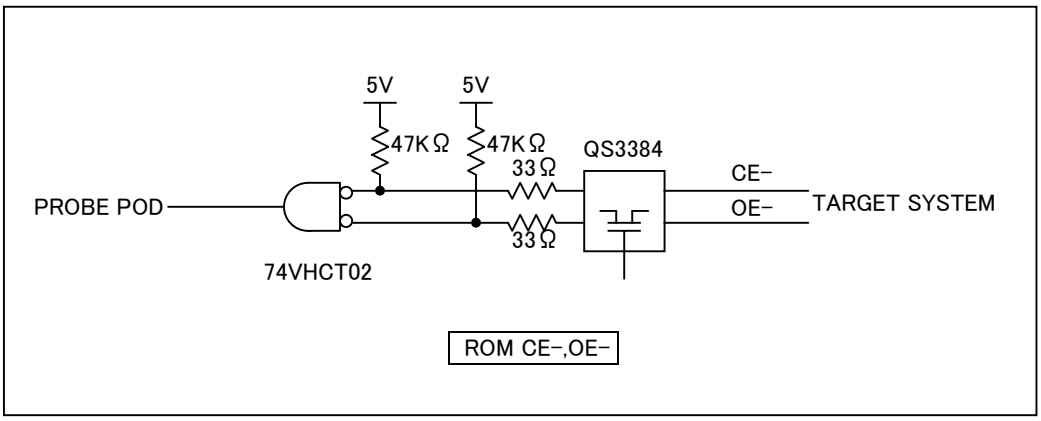
Trailing header (16 bytes)

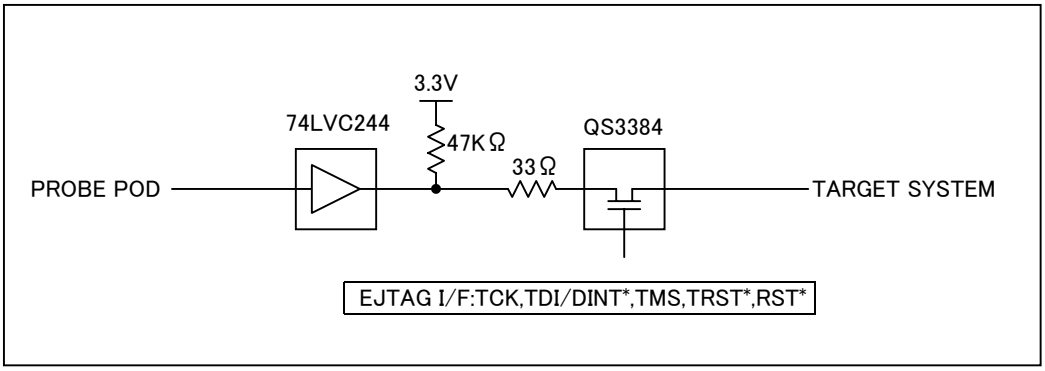
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
‘M’	‘J’	‘1’	00	00	00	00	00	00	00	00	00	00	00	00	00

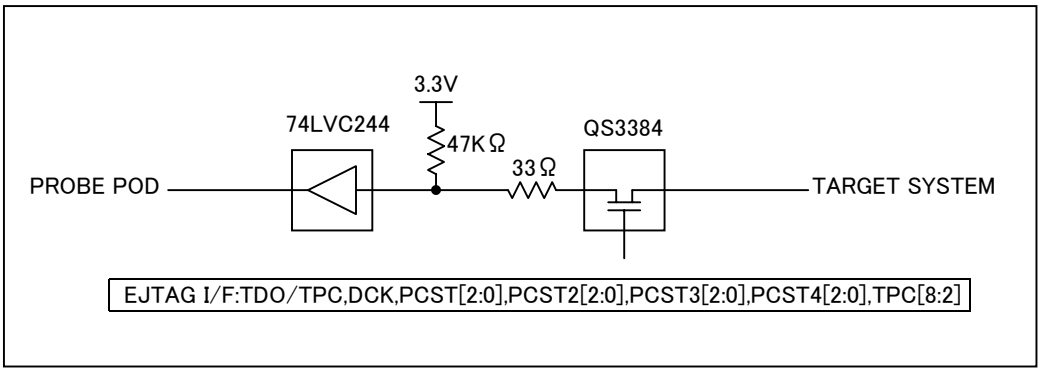
Appendix I. Probing of the Target System

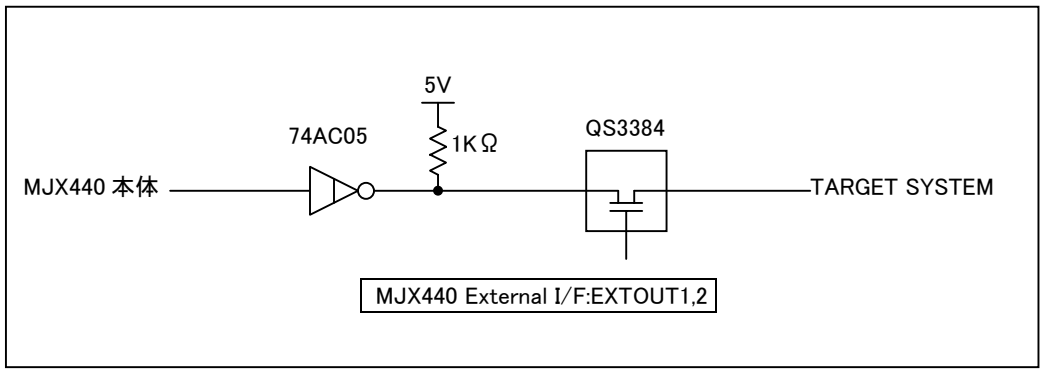


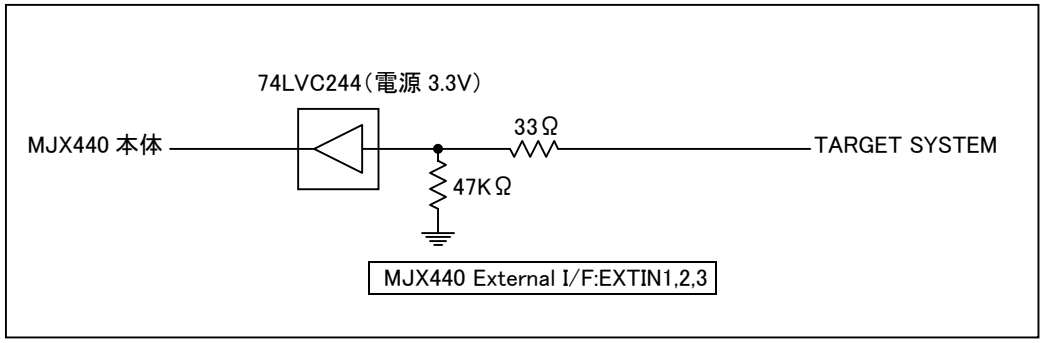


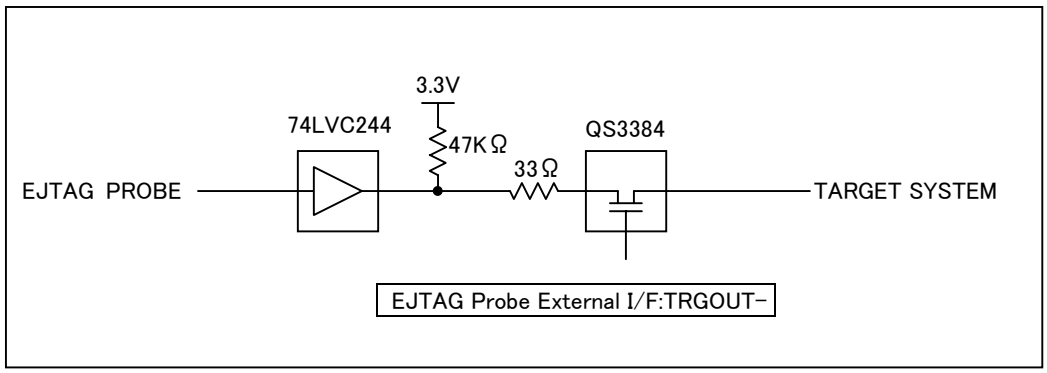

5V
5V
47KΩ
47KΩ
33Ω
33Ω
QS3384
CE−
OE−
TARGET SYSTEM
PROBE POD
74VHCT02
ROM CE−,OE−

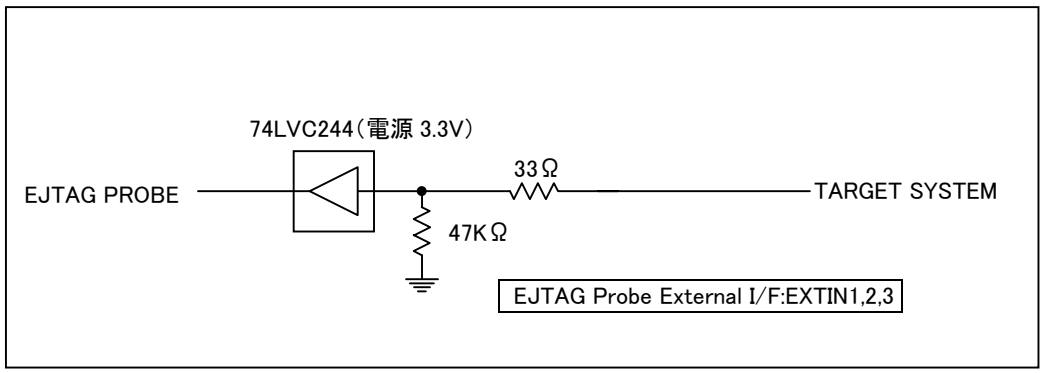

3.3V
74LVC244
47KΩ
33Ω
QS3384
PROBE POD
TARGET SYSTEM
EJTAG I/F:TCK,TDI/DINT*,TMS,TRST*,RST*

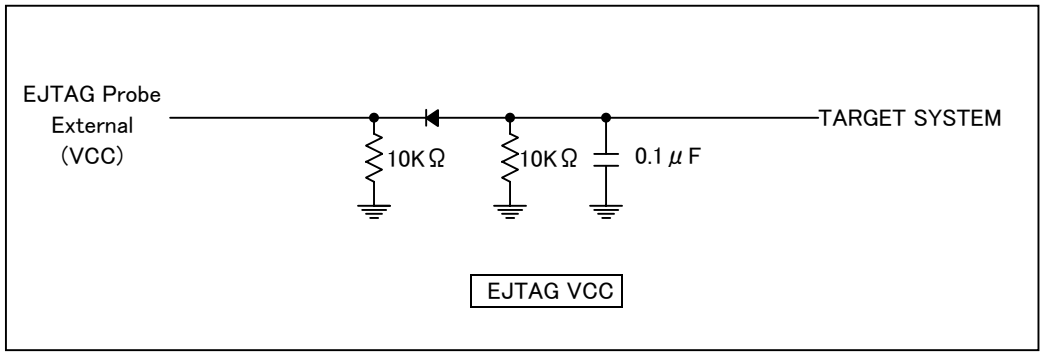

3.3V
74LVC244
47KΩ
33Ω
QS3384
PROBE POD
TARGET SYSTEM
EJTAG I/F:TDO/TPC,DCK,PCST[2:0],PCST2[2:0],PCST3[2:0],PCST4[2:0],TPC[8:2]

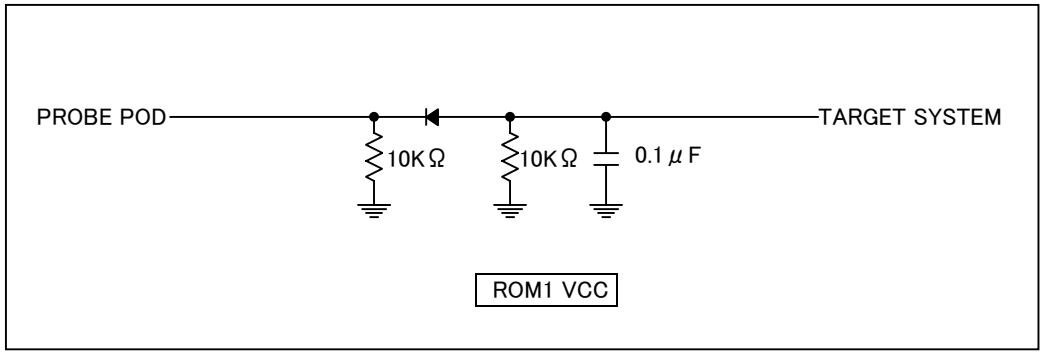

5V
74AC05
1KΩ
QS3384
MJX440 本体
TARGET SYSTEM
MJX440 External I/F:EXTOUT1,2

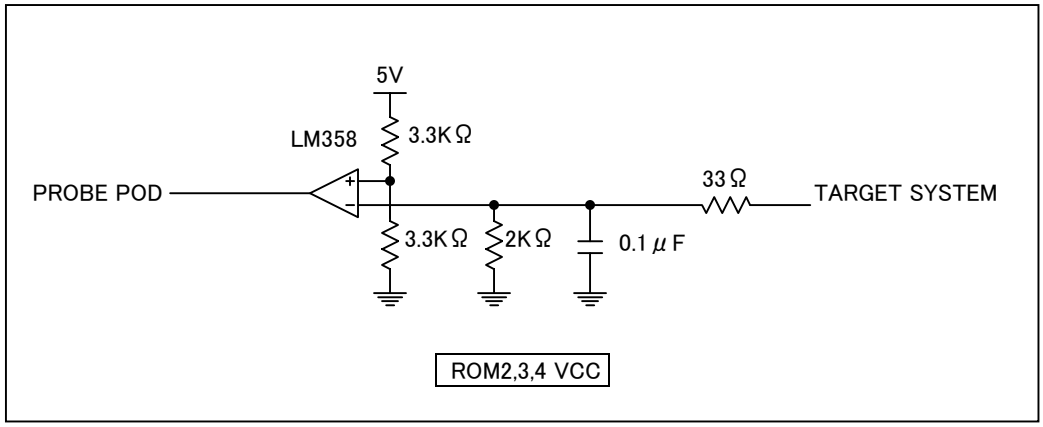

74LVC244（電源 3.3V）
MJX440 本体
33Ω
TARGET SYSTEM
47KΩ
MJX440 External I/F:EXTIN1,2,3


3.3V
74LVC244
47KΩ
QS3384
33Ω
EJTAG PROBE
TARGET SYSTEM
EJTAG Probe External I/F:TRGOUT-

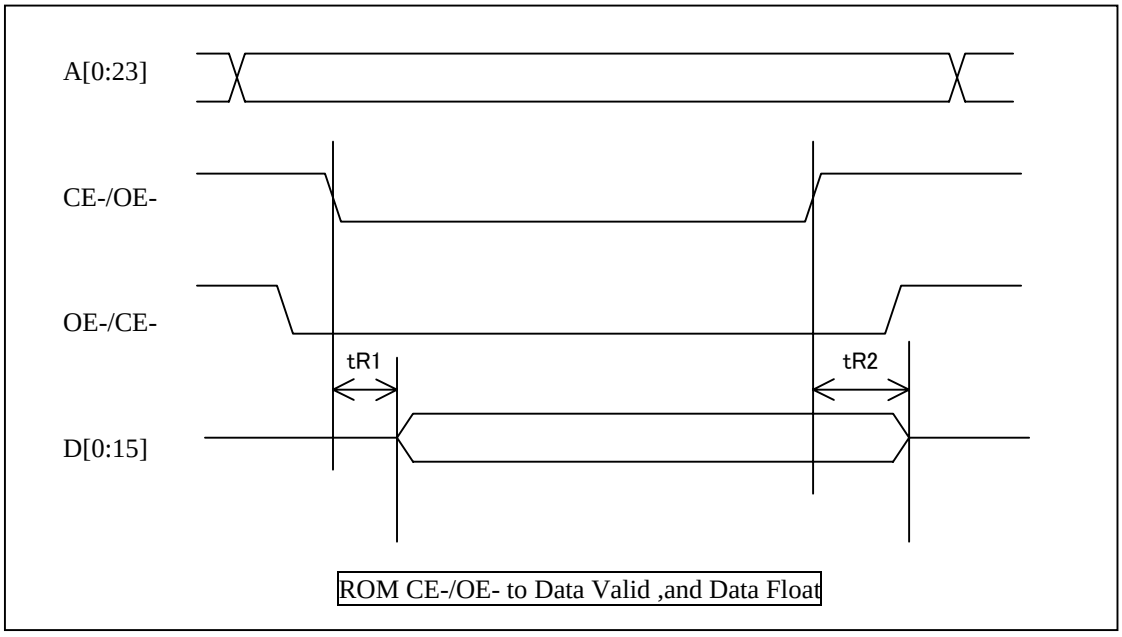

74LVC244（電源 3.3V）
33Ω
EJTAG PROBE
TARGET SYSTEM
47KΩ
EJTAG Probe External I/F:EXTIN1,2,3


EJTAG Probe
External
(VCC)
10KΩ
10KΩ
0.1μF
TARGET SYSTEM
EJTAG VCC

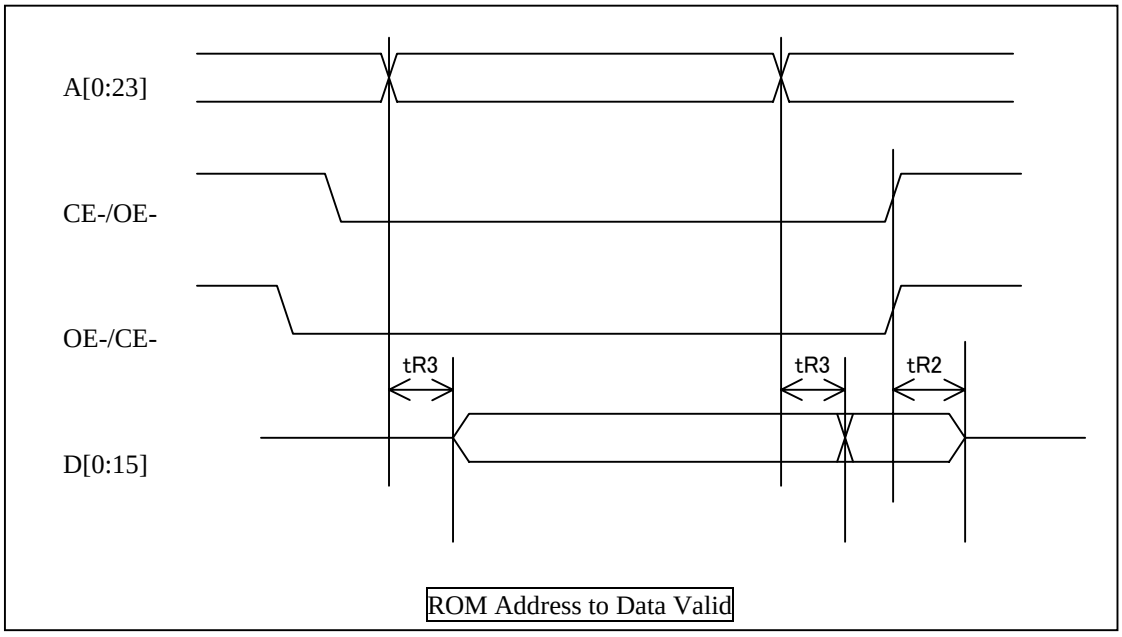

PROBE POD
10KΩ
10KΩ
0.1μF
TARGET SYSTEM
ROM1 VCC


5V
LM358
3.3KΩ
PROBE POD
33Ω
TARGET SYSTEM
3.3KΩ
2KΩ
0.1μF
ROM2,3,4 VCC

符号	項　　目	PROBE POD(TYP)
tR1	CE-/OE- to Data Valid Delay	50nS
tR2	CE-/OE- to Data Float Delay	30nS
tR3	Address to Data Valid Delay	50nS



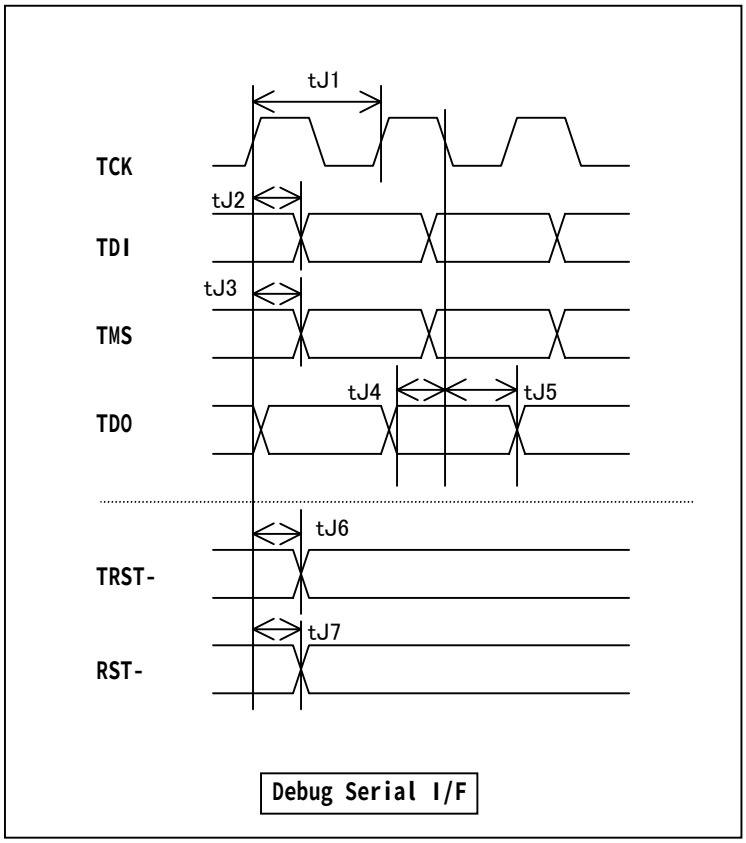
ROM CE-/OE- to Data Valid ,and Data Float



ROM Address to Data Valid

符号	項　　目	PROBE		
		MIN	TYP	MAX
tJ1	TCK Clock Period		25	
tJ2	TDI Valid Delay	5	16	
tJ3	TMS Valid Delay	5	12	
tJ4	TDO in Setup Time	20		
tJ5	TDO in Hold Time	0		
tJ6	TRST- Active Delay		5	
tJ7	RST- Active Delay		6	

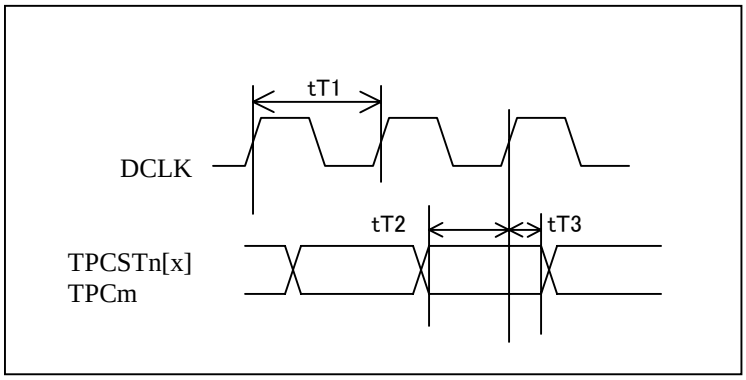
All values are in the target system.



EJTAG (TRACE)

符号	項　　目	PROBE		
		MIN	TYP	MAX
tT1	DCLK Clock Period			125MHz
tT2	Trace data in Setup Time	8*		
tT3	Trace data in Hold Time	0		

All values are in the target system.



The tT2 is 8～23nsec. It sampled with 1-clock delay.

Appendix J. Configuration file details

Sample of Configuration file:

```
[TARGET]
CPUTYPE=3                       ;; 3=mips_32bit_cpu, 4=mips_64bit_cpu
ENDIAN=1                        ;; 0=big_endian, 1=little_endian

MIPSPHYADDR=32                  ;; addrss_bus_phisical_width : 32 or 64
MIPS16ENA=0                     ;; MIPS16 (compressed 16bit code) enable : 0=disable, 1=enable

[MJX440]
VERSION=120                     ;;
JCLOCK=1                        ;; JTAG clock_speed : 0=40MHz(fast), 1=20MHz(slow)
EMMVOL=3                        ;; ROM_Emulation_MeMory volume : 0=no_emm, 1=4MB, 2=8MB, 3=16MB

[ROMEMU]
TOPADDR=0000000000000000        ;; ROM_emulation_memory top_address :
ROMIMAGE=FFFFFFFFFFFFFFFF       ;; ROM_emulation_memory image
ROMTYPE=8                       ;;  0= 1Mbit_16bit,  1= 1Mbit_8bit,  2= 2Mbit_16bit,  3= 2Mbit_8bit,
                                ;;  4= 4Mbit_16bit,  5= 4Mbit_8bit,  6= 8Mbit_16bit,  7= 8Mbit_8bit,
                                ;;  8=16Mbit_16bit,  9=16Mbit_8bit, 10=32Mbit_16bit, 11=32Mbit_8bit,
                                ;; 12=64Mbit_16bit
ROMCNT=2                        ;; number of total ROM IC/sockets : 0=0,1=1,2=2,3=4, 4=8
BUSWIDTH=0                      ;; bus width of ROM_emulation_memory : 0=32bit,1=16bit,2=8bit,3=64bit

[ENVIRONMENT]
```

Items in configuration file:

CPUTYPE **CPU data bus width** ——— Specifies CPU data bus width. If MIPS64 architecture CPU is used in 32 bit mode, specify 32 bit.

3	32 bit (MIPS32 architecture)
4	64 bit (MIPS64 architecture)

ENDIAN **CPU endian** ——— Specifies CPU endian.

0	big endian
1	little endian

MIPSPHYADDR **CPU physical address width** ——— Specify CPU physical address width. If MIPS64 architecture CPU is used in 32 bit mode, specify 64 bit.

32	32 bit physical address
64	64 bit physical address

MIPS16ENA **Use of MIPS16 instructions** ——— Specify use of MIPS16 instructions.

0	Not use MIPS16 instructions
1	Use MIPS16 instructions

VERSION **Do not change**

JCLOCK **JTAG clock rate** ——— Specify JTAG clock rate. Specify 20MHz when MJX440 does not work in 40MHz.

0	40MHz
1	20MHz

EMMVOL **Emulation memroy size** ——— Specify emulation memory size in MJX440.

0	No emulation memory
1	4M bytes

2 8M bytes

3 16M bytes

TOPADDR **ROM starging address** ——— Specify in-circuited ROM starting address. Use logical address.

ROMIMAGE **ROM image starting address** ——— Specify ROM image starting address. If no ROM image area or ROM image area is not used, specify the address **FFFFFFFFFFFFFFFF**.

By using ROMIMAGE, ROM image area can be used the same as ROM area.

ROMTYPE **ROM type** ——— Specify in-circuited ROM type. See "Appendix D ROM Probes."

0 64K x 16Bit

1 128K x 8Bit

2 128K x 16Bit

3 256K x 8Bit

4 256K x 16Bit

5 512K x 8Bit

6 512K x 16Bit

7 1M x 8Bit

8 1M x 16Bit

9 2M x 8Bit

10 2M x 16Bit

11 4M x 8Bit

12 4M x 16Bit

ROMCNT **Number of ROMs** ——— Specify number of in-circuited ROMs.

0 No ROM in-circuit

1 one ROM

2 tow ROMs

3 four ROMs

4 eight ROMs

BUSWIDTH **ROM access bus width** ——— Specify in-circuited ROM access bus width. If two or more ROMs are accessed at the same time, specify bus width of all ROMs.

0	32 bit
1	16 bit
2	8 bit
3	64 bit