MDX700
User's Manual

English

Rev.2.21 1998/11/25

for

Parallel Interface MDX700

Ethernet Interface MDX700

Notes

- You may not reproduce this manual, either wholly or in part, without the consent of Lightwell Corporation.
- Lightwell Corporation disclaims responsibility for any adverse impact from the use of this product.
- The specifications for this product and the contents of the present manual are subject to change without notice.
- MS-DOS, Windows 95, and Windows NT are registered trademarks of Microsoft Corporation.
- SunOS and Solaris are registered trademarks of Sun Microsystems Corporation.
- MULTI is a registered trademark of Green Hills Software Corporation.
- SingleStep is a registered trademark of Software Development Systems Corporation.
- XRAY is a registered trademark of Mentor Graphics Corporation.

Address: 20-12 Ogikubo 5-chome Suginami-ku Tokyo 167 Japan
TEL: 03-3392-3331
FAX: 03-3393-3878
E-mail: ZAXSupport@lightwell.co.jp
URL http://www.lightwell.co.jp/ZAX/

Printed in Japan
November 1998

Congratulations on your recent purchase of the MDX700.

This manual is organized as follows:

CHAPTER 1: OVERVIEW

Provides an overview of the MDX700, and describes the operating environment., the product composition, the hardware configuration, and the software configuration for the MDX700.

CHAPTER 2: CONNECTING HARDWARE

Describes how to connect the MDX700 to a host system and to a target system.

CHAPTER 3: INSTALLING THE DEBUGGER

Describes how to install the Debugger.

CHAPTER 4: SETTING THE ENVIRONMENT FOR THE DEBUGGER

Describes how to set the environment necessary for using the Debugger, and, in particular, how to modify the configuration file (mdx.cfg).

CHAPTER 5: LAUNCHING THE DEBUGGER

Describes how to launch the Debugger.

CHAPTER 6: MDXDEB COMMANDS

Describes how to use commands for the quick Debugger MDXDEB.

CHAPTER 7: MDXCVT

Describes the operation of the file conversion tool MDXCVT that allows you to quickly download S-record files and IEEE695 files.

APPENDIXES

Provide specifications, explain constraints on the target system, and describe technical information, including supported ROMs and CPUs.

The following symbols, when used in a particular section in this manual, mean that the accompanying explanation is applicable only to the specific environment:

Parallel	Parallel interface for the MDX700
Ethernet	Ethernet interface for the MDX700

PC/AT	PC/AT-compatible host
PC-98	PC-98 host
SPARC	SPARCstation host

68000	68000 CPU family
ARM	ARM CPU family (including THUMB)
THUMB	THUMB CPU
MIPS	MIPS CPU family
PowerPC	PowerPC CPU family
SH	SuperH RISC engine CPU family (SH-1, SH-2, or SH-3)
SH-1	SH7030 series or SH7020 CPU series
SH-2	SH7600 CPU series
SH-3	SH7700 CPU series
V800	V800 CPU series (including V830, V850, and V850E)
V830	V830 CPU family
V850	V850 CPU family
V850E	V850E CPU family

CONTENTS

CHAPTER 1: OVERVIEW

This chapter provides an overview of the MDX700, and describes the operating environment., the product composition, the hardware configuration, and the software configuration for the MDX700.

Please read this chapter if you are a first-time user of the MDX700.

1.1 Overview of the MDX700

The MDX700 is a development aid device for use with embedded systems. Because it incorporates a ROM-based in-circuit method, the MDX700 offers the following features:

- The MDX700 is independent of the shapes of CPUs.
- The MDX700 is capable of accommodating a variety of CPUs through debugger changes.
- The MDX700 is capable of operating reliable, even in high-speed CPU target systems.
- The MDX700 offers a fast download capability (256KB/sec. Parallel)

The MDX700 operates with a debugger running on a host system. A variety of debuggers can be selected for the operation with the MDX700 to suit the particular CPU that is used and the environment for the developmental language that is employed. The MDX700 supports the following debuggers:

- MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 (68K/ARM/MIPS/PowerPC/SH/V800)
- MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1 (68K/ARM/MIPS/PowerPC/SH/V800)
- MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris (68K/ARM/MIPS/PowerPC/SH/V800)
- SingleStep 6.5 68K on Windows 3.1
- SingleStep 6.5 PowerPC on Windows 3.1
- XRAY68K 2.2a on MS-DOS PC/AT (product name: XHI68KMD)
- XRAY68K 2.2a on MS-DOS PC-98 (product name: XHI68KMD)
- XRAY68K 3.4 on SunOS/Solaris (product name: XHI68KMD)
- MDXDEB 3.5 on MS-DOS/Windows 3.1/95 (68K/ARM/MIPS/PowerPC/SH/V800)
- MDXDEB 3.5 on SunOS/Solaris (68K/ARM/MIPS/PowerPC/SH/V800)

CHAPTER 1. OVERVIEW

These debuggers provide basic debugging functions that allow you to download and execute user-generated programs, to set breakpoints, and to reference and overwrite variables registers.

The MDX700, which is connected to the ROM socket for the target system, can also be used to access RAM, I/O devices, and other non-ROM resources. Thus, the MDX700 allows you to download user programs into the RAM area and to write data to I/O devices.

Before using the MDX700 and any of the associated debuggers, you must perform the preliminary actions listed below. Please refer to Chapters 2 through 4 for a description of how to perform these actions.

- Connecting the MDX700 to the host system
- Connecting the MDX700 to the target system
- Installing the debugger
- Setting the environment for the debugger
- Setting the IP address for the MDX700 Ethernet
- Registering the MDX700's IP address and the host name Ethernet

Refer to Chapter 5 for a description of how to launch the debugger. If the debugger fails to start, make sure that all the preliminary actions have been performed correctly. If the problem still persists, refer to Appendix O, "Troubleshooting".

For using the MDXDEB as a debugger, refer to Chapter 6. When using a debugger other than the MDXDEB, refer to the release notes and the manual for that debugger.

If a problem occurs when the debugger is being used, refer to Appendix C, "Notes", and Appendix K, "Running User Program with the Monitor Program".

Refer to Chapter 7 for a description of how to use the MDXCVT, which is a file conversion tool that is supplied with the debugger. The MDXCVT converts S-record files and IEEE695 files into a format that permits their fast downloading.

1.2 Operating Environment

The following describes the environment in which the MDX700 and the debugger (MDXDEB) can operate.

Parallel

● Host	A PC/AT or compatible to which an ISA board can be connected A PC-98 to which a C-BUS board can be connected
● OS	MS-DOS, Windows 3.1, or Windows 95
● Memory	8Mb or larger
● Hard disk	1Mb or larger available space
● Target system	16Kb free space for ROM 4Kb free space for RAM RESET and NMI signal line connectivity desirable

Ethernet

● Host	SPARCstation
● OS	SunOS 4.x, Solaris 5.x
● Memory	8Mb or larger
● Hard disk	1Mb or larger available space
● Target system	16Kb free space for ROM 4Kb free space for RAM RESET and NMI signal line connectivity desirable

Debuggers other than the MDXDEB may require different operating conditions. Refer to the release notes for the desired debugger for details.

Refer to the compiler manual for a description of the operating environment that is required by the compiler.

For a description of the operating environment for the target system, refer to Appendix B, "Target System Constraints".

1.3 Product Composition

As a product, the MDX700 is composed of the following components. If any of the components are missing in the package you purchased, please contact Lightwell Corporation. The voltage converter adapter MDX003 is an optional item, which may be purchased separately.

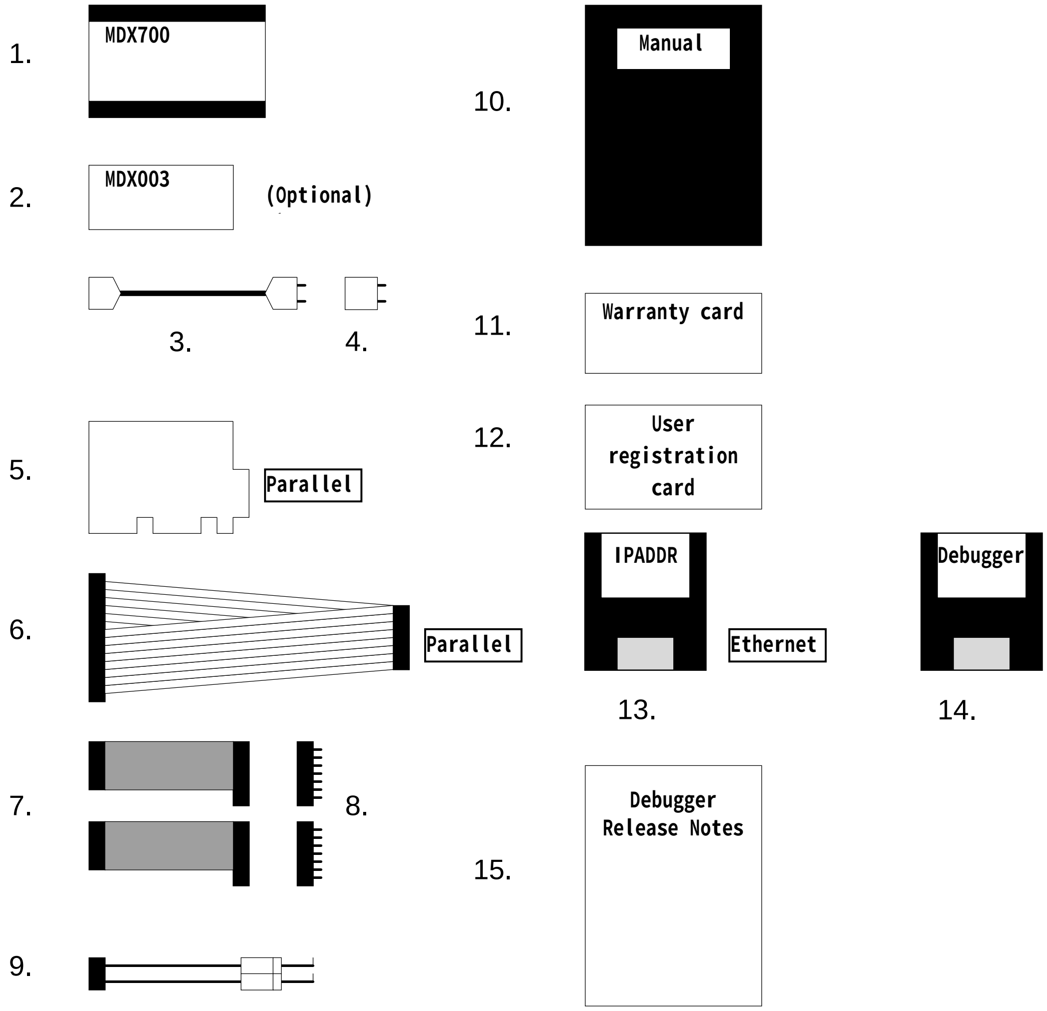


Figure 1-1 Product composition

1. MDX700 system unit
2. MDX003 (optional)
3. Power cable
4. 3P-2P conversion socket
5. Parallel interface board (ISA or C-BUS board) Parallel
6. Parallel interface cable Parallel
7. ROM cable (see the following pages)
8. ROM probe (see the following pages)
9. External trigger cable
10. User's manual
11. Warranty card
12. User registration card
13. Floppy disk for IP address setup program Ethernet
14. Debugger floppy disk
15. Debugger resource notes

NOTE: The sign Parallel or Ethernet indicates that the product is included only in the MDX700 with the identified interface specification.

NOTE: See the following pages for ROM cable and ROM probe product compositions.

IMPORTANT: Please fill out and mail your user registration card to Lightwell Corporation.

In the product composition for the MDX700, the ROM cables and the ROM probes may vary according to the particular configuration of ROM. The table below shows the configurations of ROM and the corresponding ROM cables/ROM probes. In the table, the figures indicate the number of ROM cables or ROM probes that are included in the product package.

ROM	ROM cable		ROM probe	
configuration	RC28AD	RC28D	B106	B107
27256 x 1	1	-	1	-
27256 x 2	1	1	2	-
27256 x 4	1	3	4	-
27512 x 1	1	-	-	1
27512 x 2	1	1	-	2
27512 x 4	1	3	-	4

Table 1-1-1 Correspondence between ROM configuration and ROM cables/ROM probes (1)

ROM	ROM cable		ROM probe					
configuration	RC32AD	RC32D	B108	B109	B111	B112	B117	B119
27010 x 1	1	-	1	-	-	-	-	-
27010 x 2	1	1	2	-	-	-	-	-
27010 x 4	1	3	4	-	-	-	-	-
27020 x 1	1	-	-	-	1	-	-	-
27020 x 2	1	1	-	-	2	-	-	-
27020 x 4	1	3	-	-	4	-	-	-
27040 x 1	1	-	-	-	-	1	-	-
27040 x 2	1	1	-	-	-	2	-	-
27040 x 4	1	3	-	-	-	4	-	-
271000 x 1	1	-	-	1	-	-	-	-
271000 x 2	1	1	-	2	-	-	-	-
271000 x 4	1	3	-	4	-	-	-	-
27C4000/8bit x1	1	-	-	-	-	-	1	-
27C4000/8bit x2	1	1	-	-	-	-	2	-
27C4000/8bit x4	1	3	-	-	-	-	4	-
27C8000/8bit x1	1	-	-	-	-	-	-	1
27C8000/8bit x2	1	1	-	-	-	-	-	2
27C8000/8bit x4	1	3	-	-	-	-	-	4

Table 1-1-2 Correspondence between ROM configuration and ROM cables/ROM probes (2)

ROM configuration	ROM cable		ROM probe				
	RC40AD	RC40D	B110	B113	B118	B120	PL44
271024 DIP x 1	1	-	1	-	-	-	-
271024 DIP x 2	1	1	2	-	-	-	-
271024 PLCC x 1	1	-	-	-	-	-	1
271024 PLCC x 2	1	1	-	-	-	-	2
274096 DIP x 1	1	-	-	1	-	-	-
274096 DIP x 2	1	1	-	2	-	-	-
274096 PLCC x 1	1	-	-	-	-	-	1
274096 PLCC x 2	1	1	-	-	-	-	2
27C4000/16bit x 1	1	-	-	-	1	-	-
27C4000/16bit x 2	1	1	-	-	2	-	-
27C8000/16bit x 1	1	-	-	-	-	1	-
27C8000/16bit x 2	1	1	-	-	-	2	-

Table 1-1-3 Correspondence between ROM configuration and ROM cables/ROM probes (3)

ROM	ROM cable		ROM probe
configuration	RC32AD	RC32D	29F040
29F040 x 1	1	-	1
29F040 x 2	1	1	2
29F040 x 4	1	3	4

Table1-1-4 Correspondence between ROM configuration and ROM cables/ROM probes (4)

ROM	ROM cable		ROM probe		Adapter
configuration	RC40AD	RC40AD	B117	B118	DSOP44RB
28F400/8bit x 1	1	-	1	-	1
28F400/8bit x 2	1	1	2	-	2
28F400/16bit x 1	1	-	-	1	1
28F400/16bit x 2	1	1	-	2	2

Table 1-1-5 Correspondence between ROM configuration and ROM cables/ROM probes (5)

The ROM cable codes have the following meanings:

●	RC28AD	A 28-pin ROM cable with address cable	
●	RC28D	A 28-pin ROM cable	
●	RC32AD	A 32-pin ROM cable with address cable	(also for 27C4000/8bit, 27C8000/8bit)
●	RC32D	A 32-pin ROM cable	(also for 27C4000/8bit, 27C8000/8bit)
●	RC40AD	A 40-pin ROM cable with address cable	(also for 27C4000/16bit, 27C8000/16bit)
●	RC40D	A 40-pin ROM cable	(also for 27C4000/16bit, 27C8000/16bit)

The ROM probe codes have the following meanings:

●	B106	A ROM probe for the 27256
●	B107	A ROM probe for the 27512
●	B108	A ROM probe for the 27010
●	B109	A ROM probe for the 271000
●	B110	A ROM probe for the 271024DIP
●	B111	A ROM probe for the 27020
●	B112	A ROM probe for the 27040
●	B113	A ROM probe for the 274096
●	B117	A ROM probe for the 27C4000/8bits
●	B118	A ROM probe for the 27C4000/16bits
●	B119	A ROM probe for the 27C8000/8bits
●	B120	A ROM probe for the 27C8000/16bits
●	PL44	A ROM probe for the 271024PLCC/274096PLCC
●	29F040	A ROM probe for the 27F040

The conversion adapter codes have the following meanings:

●	DSOP44RB	SOP44pin-to-DIP conversion adapter

See Appendix D, "Supported ROMs", for a list of ROMs for which there are corresponding ROM probes.

1.4 Hardware Configuration

The MDX700 has the following hardware configuration:

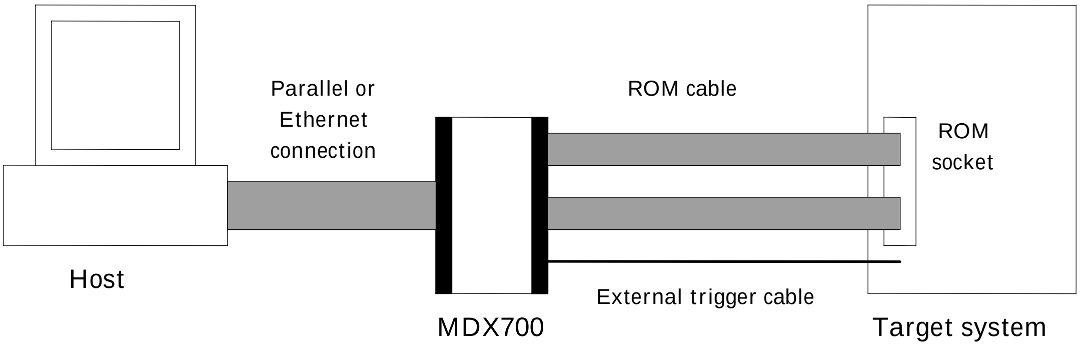


Figure 1-2 Hardware configuration

In the case of a Parallel connection, a parallel interface board is installed in the expansion slot for the host system, and the MDX700 and the board are connected using a parallel interface cable.

In the case of an Ethernet connection, the HUB that connects the MDX700 and the host in a network is connected using a 10BASE-T cable. Also, the host and the MDX700 can be connected directly by using a 10BASE-T cross cable.

The MDX700 and the ROM socket for the target system are connected using a ROM cable and a ROM probe.

The operation of the debugger can be enhanced by connecting the MDX700 and the target system using an external trigger cable. Although the debugger can be operated without using an external trigger cable, the use of an external trigger cable is recommended.

For the target system that has a 3V ROM peripheral circuit, the voltage conversion adapter must be connected between the MDX700 and the target system. The MDX003 is an optional item.

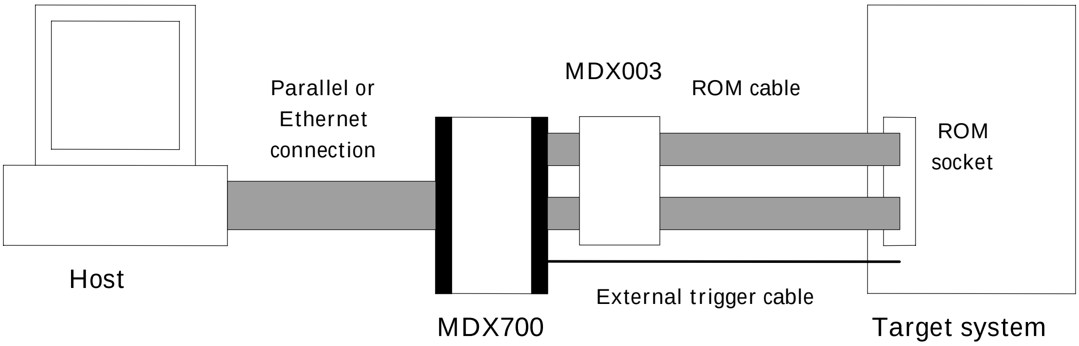


Figure 1-3 Hardware configuration with an MDX003

For details on hardware connection, see Chapter 2, "Connecting the Hardware".

1.5 Software Configuration

The MDX700 has the following software configuration:

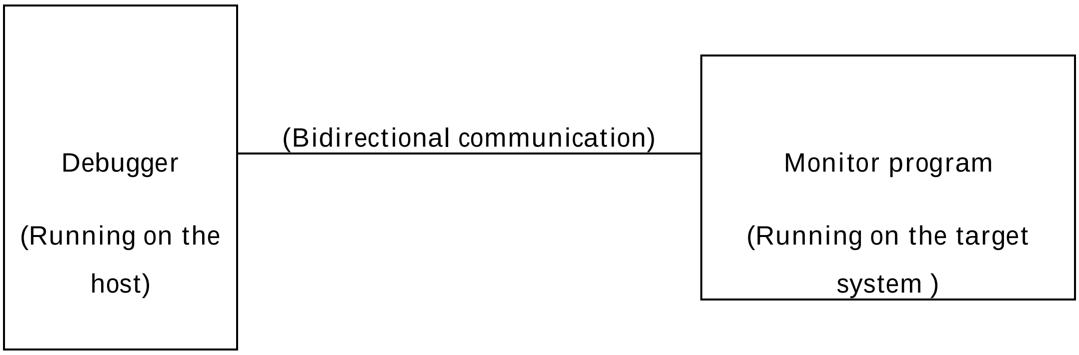

Debugger
(Running on the host)
(Bidirectional communication)
Monitor program
(Running on the target system)

Figure 1-4 Software configuration

The debugger for the operation of the MDX700 requires the operation of a monitor program on the target system in order to implement debugging functions.

The monitor program executes the functional requests that are generated by the debugger on an "as needed" basis, and return the results of the program execution to the debugger. The debugger performs any access to the ROM region while the monitor program performs any access to the RAM region and executes user programs.

Because the monitor program and user programs run on the same system, care should be taken to prevent memory contention between them. A common strategy is to set the operating environment so that the monitor program is allocated in a region that is not required by user programs.

For details on the operating principles, see Appendix H, "Operating Principles".

CHAPTER 2 CONNECTING THE HARDWARE

This chapter describes how to connect the MDX700 to a host system and how to connect the MDX700 to a target system.

Methods for connecting hardware vary with the types of hosts and target systems that are involved. For a particular connection method, see the specific environment in which the connection is to be made.

IMPORTANT: Be sure to turn off the power for the MDX700 before connecting the MDX700 to a device.

2.1 External Features

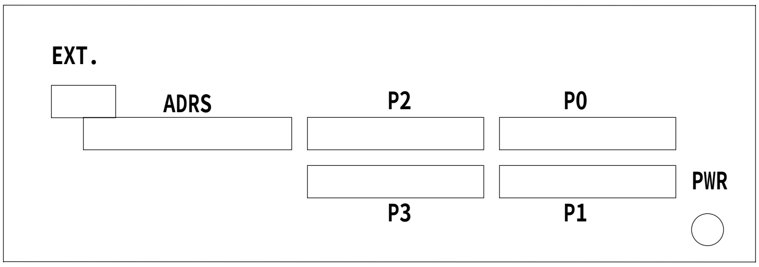


Figure 2-1 MDX700 front panel

ADRS	ROM cable connector
P0	ROM cable connector
P1	ROM cable connector
P2	ROM cable connector
P3	ROM cable connector
EXT.	External trigger cable connector
PWR	MDX003 power cable connector

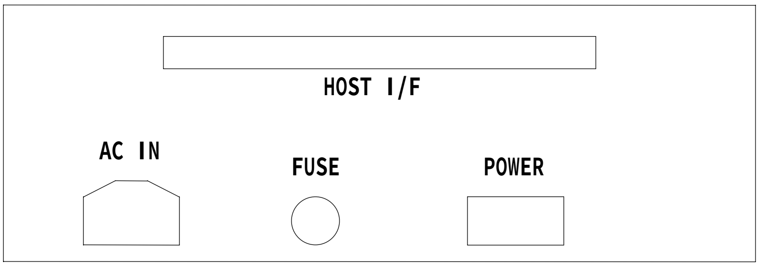


Figure 2-2 MDX700 rear panel Parallel

AC IN	Power cable connector
FUSE	Fuse
POWER	Power switch
HOST I/F	Parallel interface cable connector

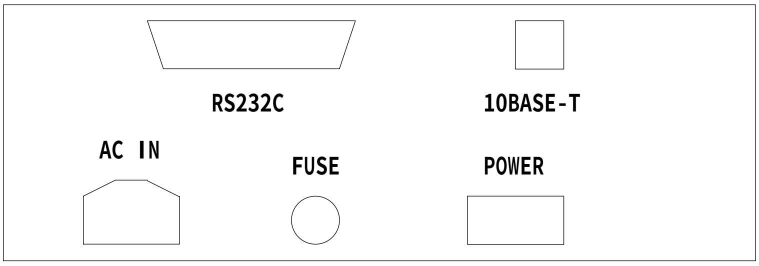


Figure 2-3 MDX700 rear panel Ethernet

AC IN	Power cable connector
FUSE	Fuse
POWER	Power switch
10BASE-T	10BASE-T Ethernet cable connector
RS232C	RS-232C cable connector (for setting an IP address)

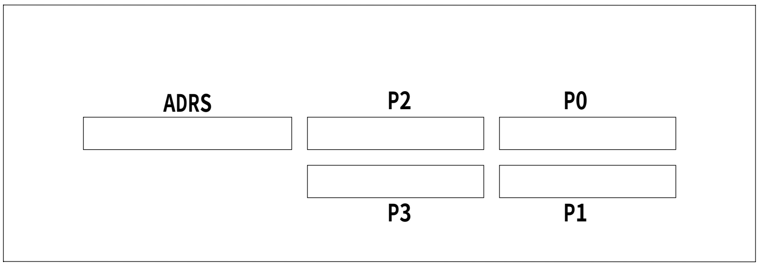


Figure 2-4 MDX003 front panel

ADRS	ROM cable connector
P0	ROM cable connector
P1	ROM cable connector
P2	ROM cable connector
P3	ROM cable connector

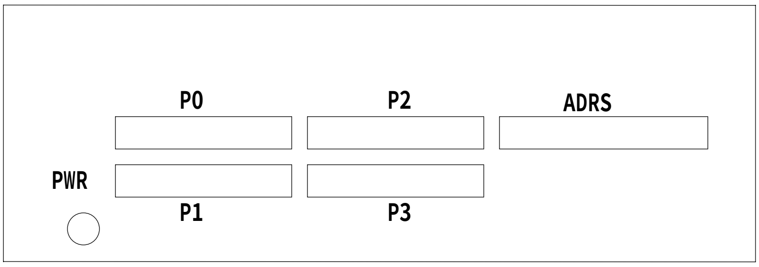


Figure 2-5 MDX003 rear panel

ADRS	MDX700 connector cable
P0	MDX700 connector cable
P1	MDX700 connector cable
P2	MDX700 connector cable
P3	MDX700 connector cable
PWR	MDX700 connector cable

2.2 Connecting the MDX700 to a Host Parallel

The MDX700 is connected to a host system using a parallel interface board and an interface cable as described below.

By factory default, the parallel interface board has settings that permit the use of the I/O address space indicated below. If these I/O addresses are already in use, you need to modify the I/O address that is set on the parallel interface board.*1 *2

PC/AT ISA board

- 0x0100-0x010F
- 0x0110-0x011F

PC-98 C-BUS board

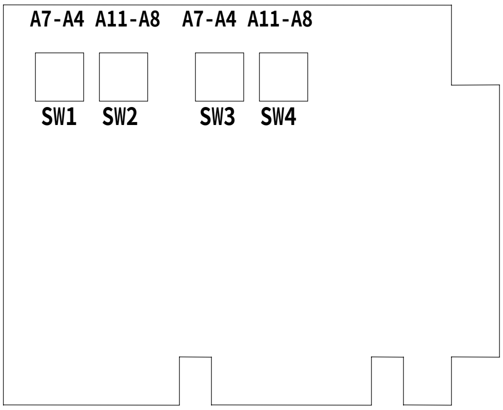
- 0x01D0-0x01DF
- 0x02D0-0x02DF

For connecting two MDX700s to the same host, different I/O addresses should be assigned to their parallel interface boards (see Figures 2-3-15 to 2-3-18).

*1 In Windows 95, you can display all the I/O addresses that are used by Windows 95 by selecting [Control Panel]_[System]_[Device Manager]_[Computer]_[Property]_[I/O Port Address].

*2 The parallel interface board for the MDX700 is not Windows 95 Plug&Play compliant.

I/O addresses on the parallel interface board are set by means of the switches that are provided on the board. You can change I/O addresses on the board by changing the switch settings.



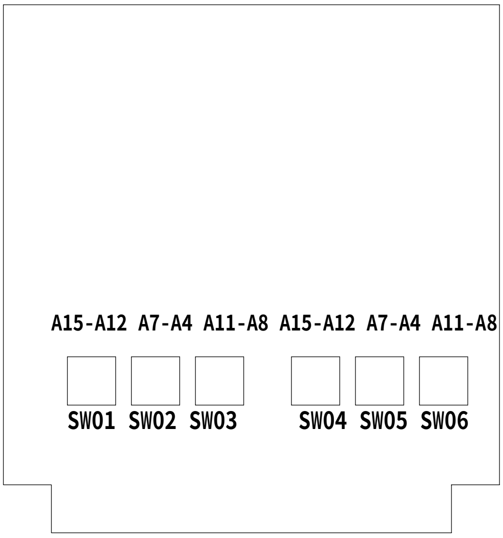
SW1 A7-A4 for the first I/O address (default: 0)

SW2 A11-A8 for the first I/O address (default: 1)

SW3 A7-A4 for the second I/O address (default: 1)

SW4 A11-A8 for the second I/O address (default: 1)

Figure 2-6 PC/AT Parallel interface board (ISA board)



SW01 A15-A12 for the first I/O address (default: 0)

SW02 A11-A8 for the first I/O address (default: 1)

SW03 A7-A4 for the first I/O address (default: D)

SW04 A15-A12 for the second I/O address (default: 0)

SW05 A11-A8 for the second I/O address (default: 2)

SW06 A7-A4 for the second I/O address (default: D)

Figure 2-7 PC-98 Parallel interface board (C-BUS board)

NOTE: PC/AT In the case of an ISA board, I/O address A15-A12 is fixed at 0.

NOTE: When changing an I/O address, be careful to avoid any duplication between first and second I/O addresses.

NOTE: I/O addresses are specified in a configuration file (more on this later).

NOTE: In the current version of MDX700, a second I/O address is not used. The first I/O address should be specified in the configuration file.

After setting the requisite I/O address, install the parallel interface board in the host's expansion slot. See the host manual for board installation procedures.

In the next step, use a parallel interface cable to connect the parallel interface board to the HOST I/F connector for the MDX700.

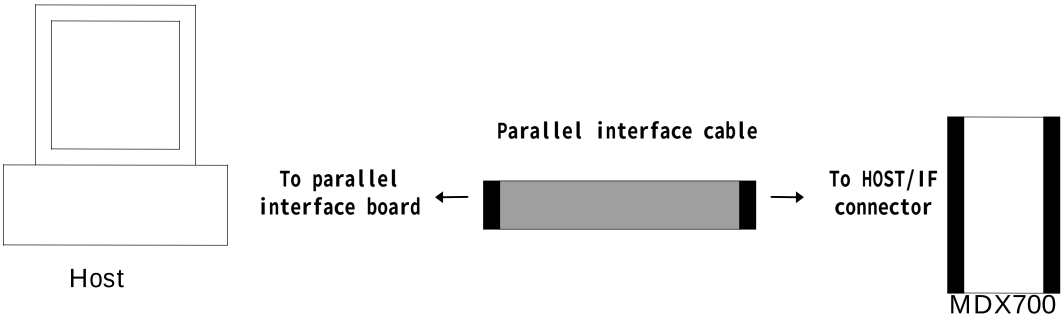


Figure 2-8 Connecting the MDX700 to a host Parallel

2.3 Setting the IP Address for the MDX700 Ethernet

To set the IP address for the MDX700, use an RS-232C cable to connect the PC (PC/AT or PC-98) to the MDX700, and then execute the IP address-setting program IPADDR.EXE on the PC. The procedures to set the address are described below.

First, use the RS-232C straight cable to connect the PC's serial port to the RS-232C connector for the MDX700. In the case of a PC/AT , the COM1 port should be used. In the case of a PC-98 , connect it to the standard RS-232C port.

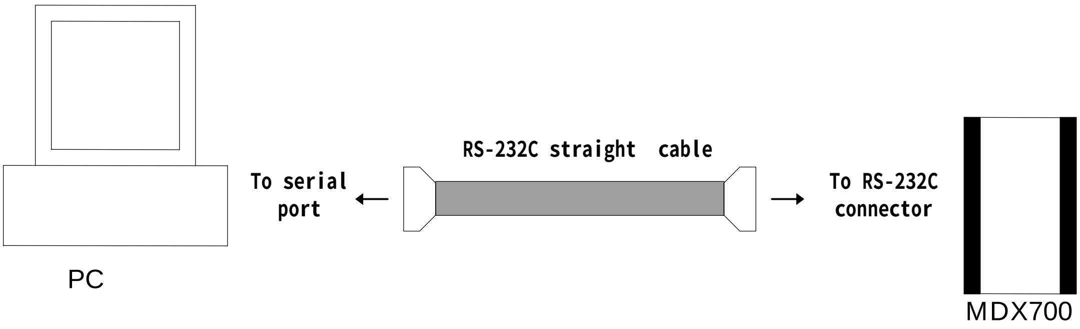


Figure 2-9 Connecting the MDX700 for setting its IP address

In the next step, turn on the power for the PC and the MDX700. In the case of Windows 95, restart your computer in the MS-DOS mode (you can return to the Windows 95 mode by entering the EXIT command).

When MS-DOS has started, insert the diskette containing the IP address-setting program IPADDR.EXE into the diskette drive, and then enter the following command (assuming that the diskette drive is A: and the IP address to be set is 192.10.20.30):[*1]

```
C:\>a:ipaddr 192.10.20.30
```

Use the following command to verify the IP address after it has been set:

```
C:\>a:ipaddr
```

[*1] IPADDR.EXE is an MS-DOS program, which runs on both the IBM-PC and the PC-98.

```
192.10.20.30
```

After setting the IP address for the MDX700, turn off the MDX700 and the PC, and disconnect the RS-232C cable.

If the IP address is not correctly set, the following error message will appear. If this happens, the connection for the serial port and RS-232C cable should be re-checked.

```
err: no response from target-system
received data:
```

After the IP address for the MDX700 has been set, the address should be registered on the host system, using the host name *mdx*.[*1]

IMPORTANT: The specific registration tasks should be performed by the network administrator.

[*1] A host name other than *mdx* can be used. However, by naming the host *mdx*, you can simplify the debugger startup options.

2.4 Connecting the MDX700 to a Host Ethernet

The MDX700 is connected to a host in a network using the connection procedures described below.

Use a 10BASE-T cable to connect the 10BASE-T connector for the MDX700 to the HUB that is connected to the host in a network.

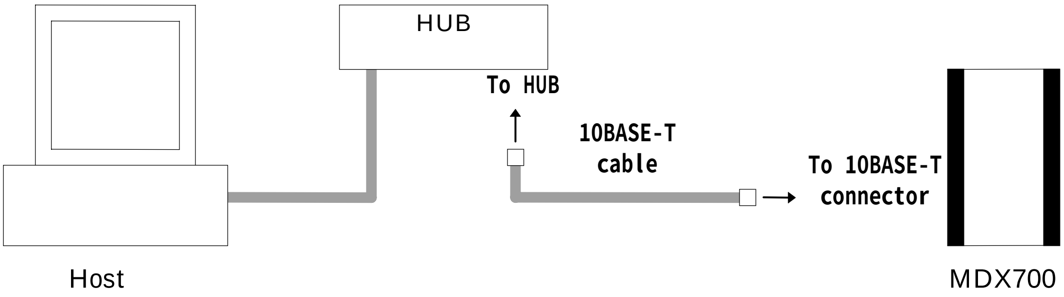


Figure 2-10-1 Connecting the MDX700 to a host (1) Ethernet

When using a 10BASE-T cross cable, directly connect the MDX700 to the host. In this case, however, no additional network devices can be used.

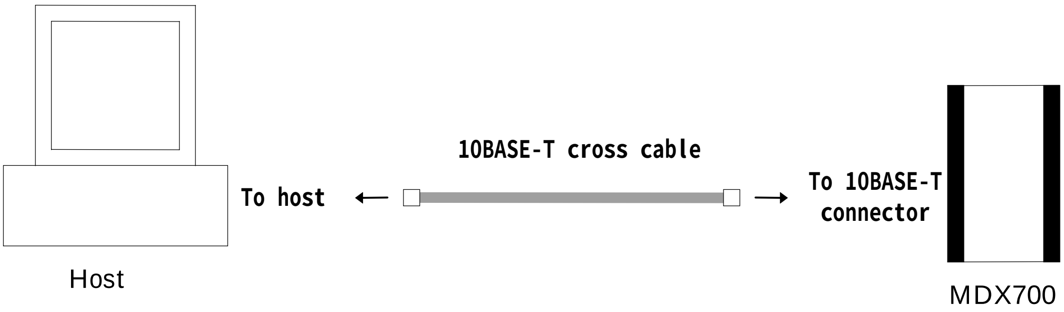


Figure 2-10-2 Connecting the MDX700 to a host (2) Ethernet

2.5 Connecting the MDX700 to an MDX003 (Optional Item)

In situations where the use of an MD003 voltage conversion adapter is required, connect the MDX700 to the MDX003 as follows:

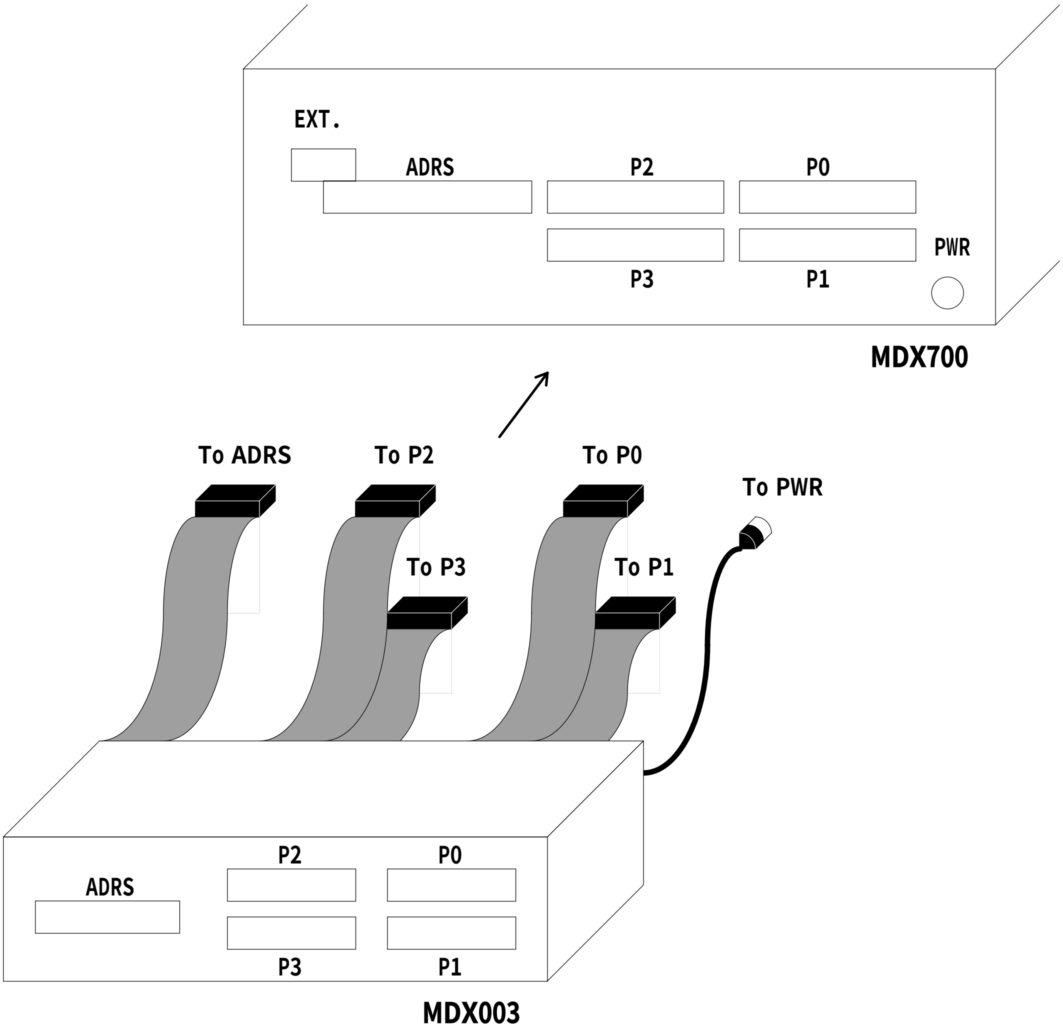


Figure 2-11 Connecting the MDX700 to an MDX003

2.6 Connecting the MDX700 to a Target System

The MDX700 (or an MDX003) is connected to a target system using a ROM cable and a ROM probe according to the following connection procedures:

First, attach ROM probes to all ROM cables.

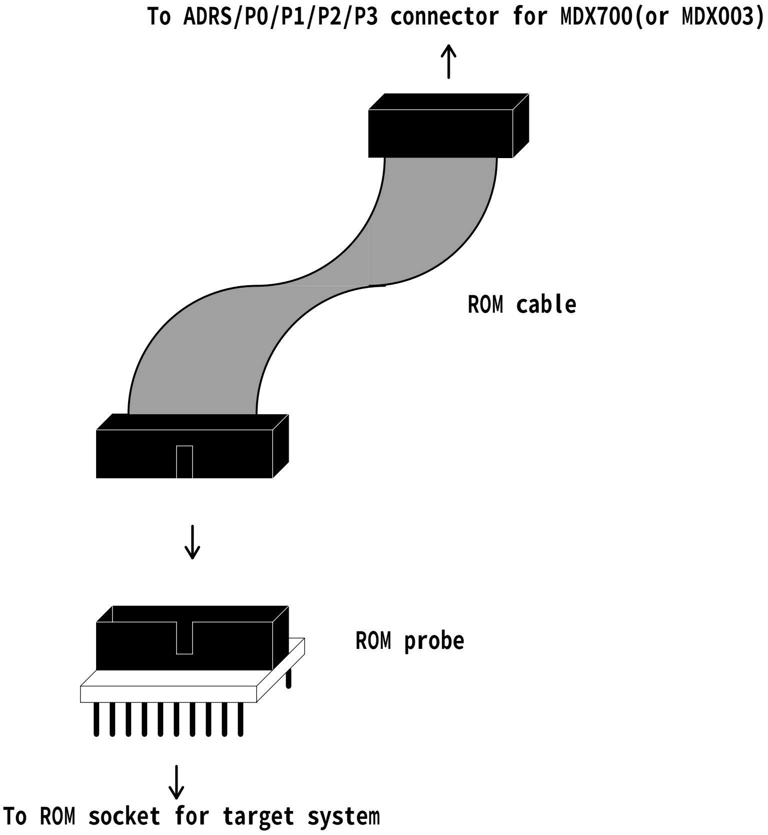

To ADRS/P0/P1/P2/P3 connector for MDX700(or MDX003)
ROM cable
ROM probe
To ROM socket for target system

Figure 2-12 Connecting a ROM cable to a ROM probe

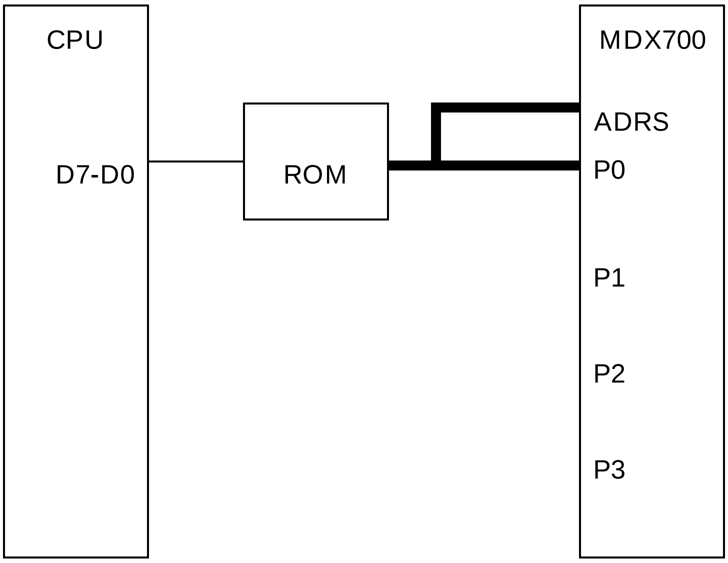
In the next step, connect the ROM probe to the ROM socket for the target system, and connect the other side of ROM cable to the ADRS/P0/P1/P2/P3 connector for the MDX700 (or the MDX003). Select the environment that matches your target system from the connection diagrams (Figures 2-13-1 ... 2-13-18), and perform connection according to the diagram.

NOTE: Before connecting cables, be sure to turn off the MDX700 and the target system.

NOTE: Be careful not to insert the ROM probe in reverse.

NOTE: An MDX003 is omitted in the connection diagrams.

NOTE: In the case of a PowerPC, the CPU data bus bits in the connection diagrams should be read in reverse order. In PowerPC, the MSB is designated as bit 0. In the connection diagramas, however, bit 0 represents the LSB.

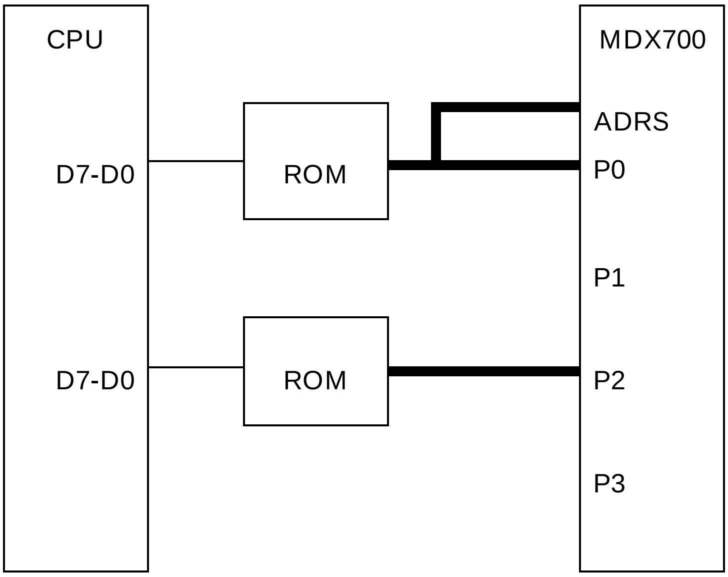


ROM read data bus width: 8 bits

ROM data bus width: 8 bits

One ROM chip

Figure 2-13-1 Connecting the MDX700 to a target system (1)

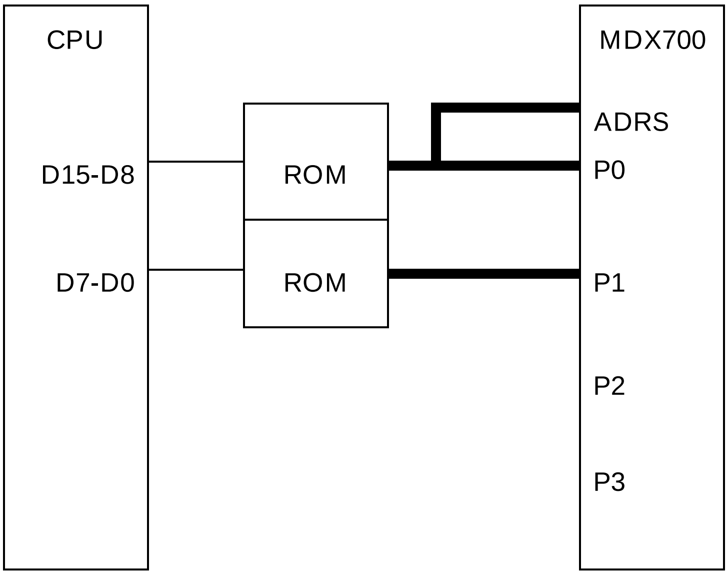


ROM read data bus width: 8 bits

ROM data bus width: 8 bits

Two ROM chips

Figure 2-13-2 Connecting the MDX700 to a target system (2)



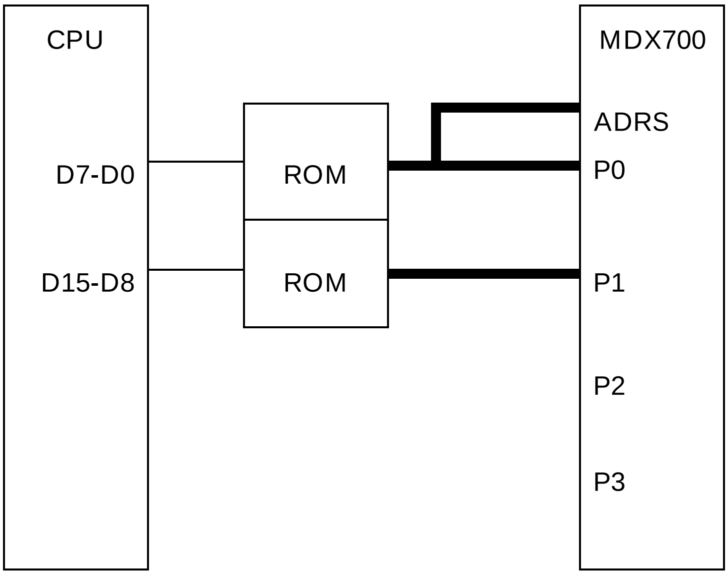
ROM read data bus width: 16 bits

ROM data bus width: 8 bits

Two ROM chips

CPU: big endian

Figure 2-13-3 Connecting the MDX700 to a target system (3)



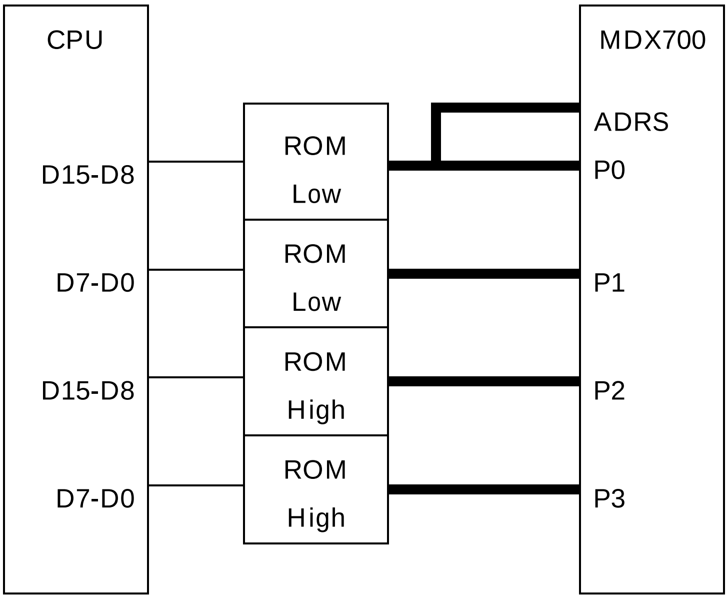
ROM read data bus width: 16 bits

ROM data bus width: 8 bits

Two ROM chips

CPU: little endian

Figure 2-13-4 Connecting the MDX700 to a target system (4)



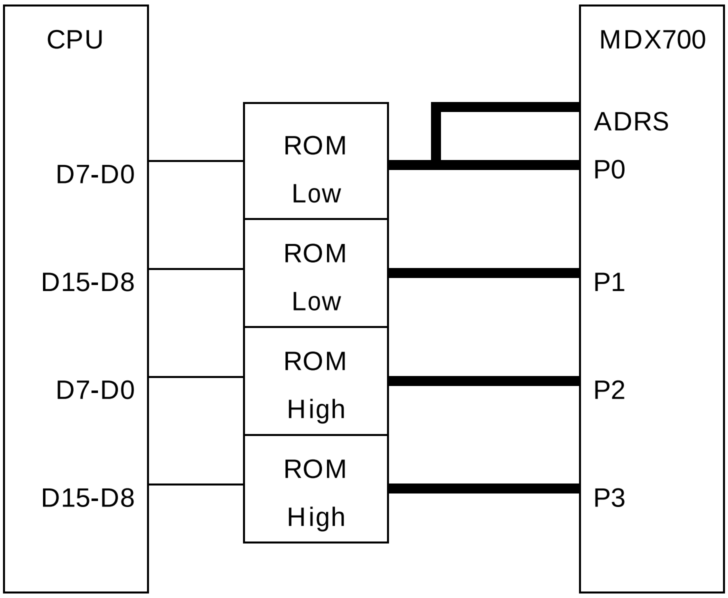
ROM read data bus width: 16 bits

ROM data bus width: 8 bits

Four ROM chips

CPU: big endian

Figure 2-13-5 Connecting the MDX700 to a target system (5)



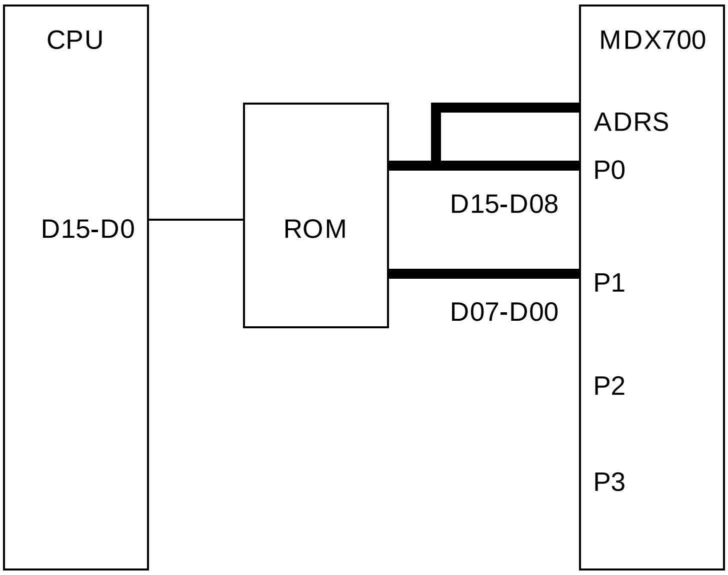
ROM read data bus width: 16 bits

ROM data bus width: 8 bits

Four ROM chips

CPU: little endian

Figure 2-13-6 Connecting the MDX700 to a target system (6)



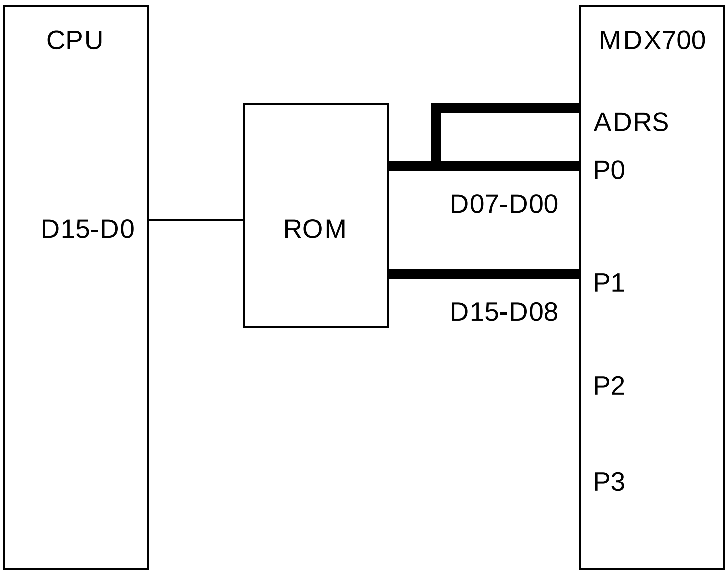
ROM read data bus width: 16 bits

ROM data bus width: 16 bits

One ROM chip

CPU: big endian

Figure 2-13-7 Connecting the MDX700 to a target system (7)



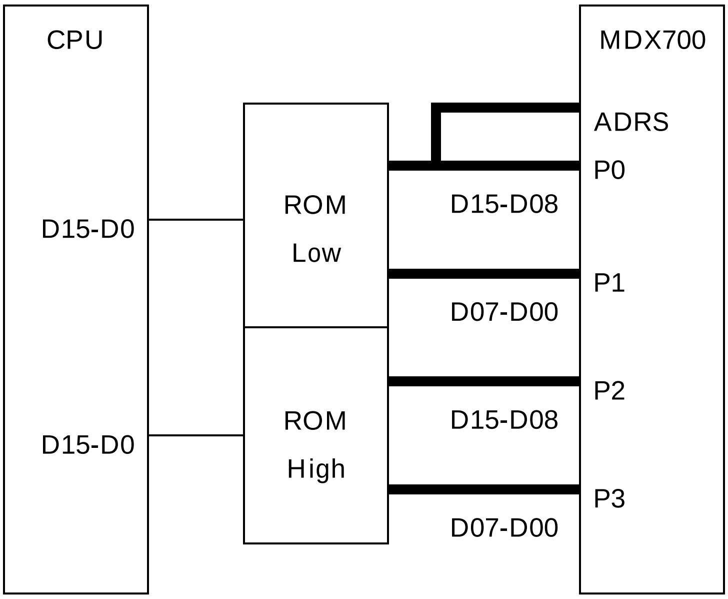
ROM read data bus width: 16 bits

ROM data bus width: 16 bits

One ROM chip

CPU: little endian

Figure 2-13-8 Connecting the MDX700 to a target system (8)



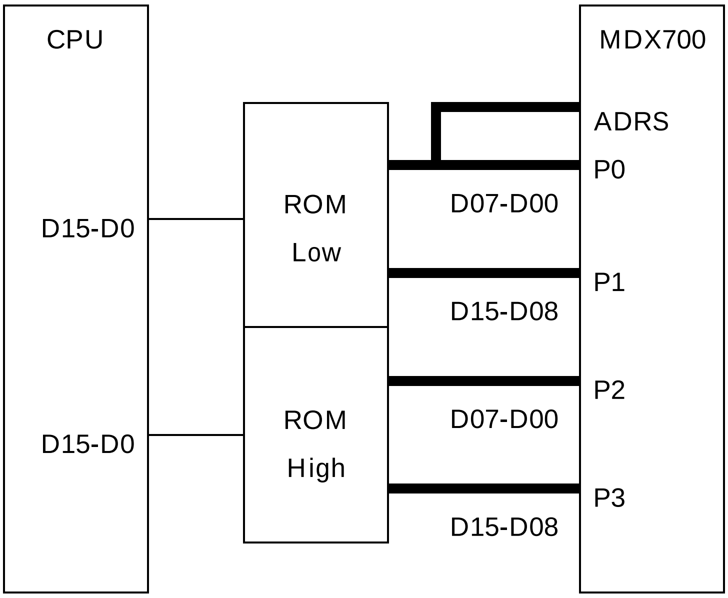
ROM read data bus width: 16 bits

ROM data bus width: 16 bits

Two ROM chips

CPU: big endian

Figure 2-13-9 Connecting the MDX700 to a target system (9)



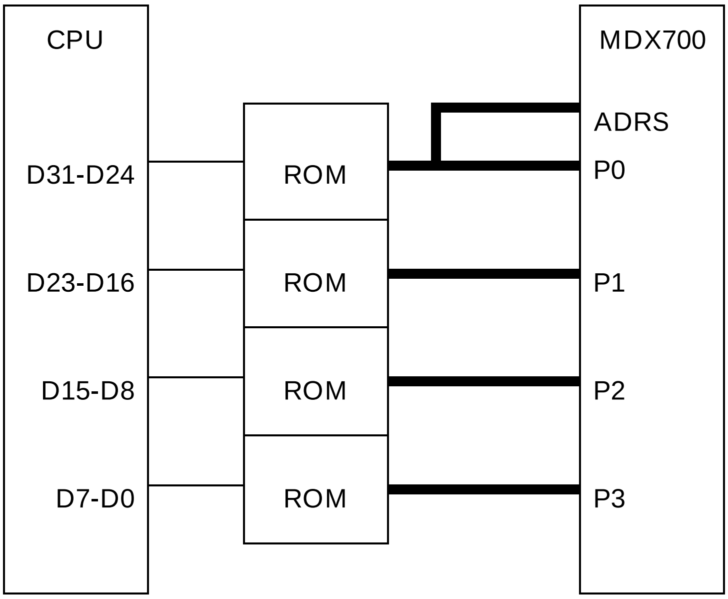
ROM read data bus width: 16 bits

ROM data bus width: 16 bits

Two ROM chips

CPU: little endian

Figure 2-13-10 Connecting the MDX700 to a target system (10)



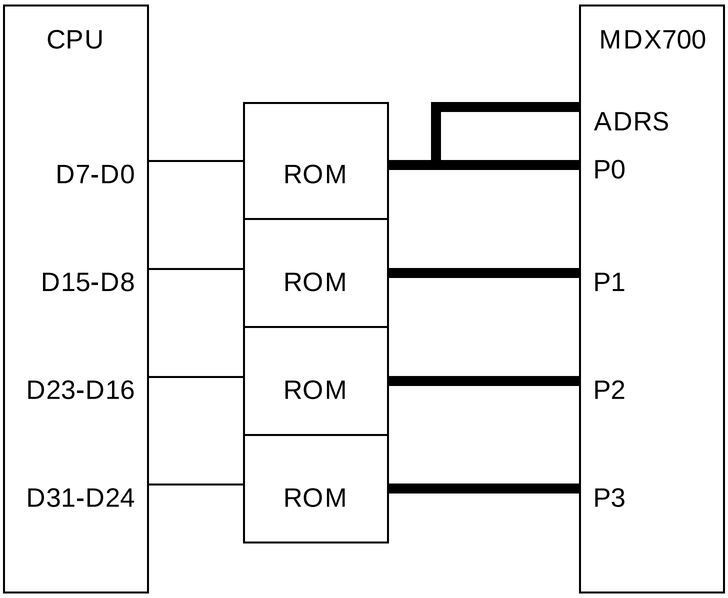
ROM read data bus width: 32 bits

ROM data bus width: 8 bits

Four ROM chips

CPU: big endian

Figure 2-13-11 Connecting the MDX700 to a target system (11)



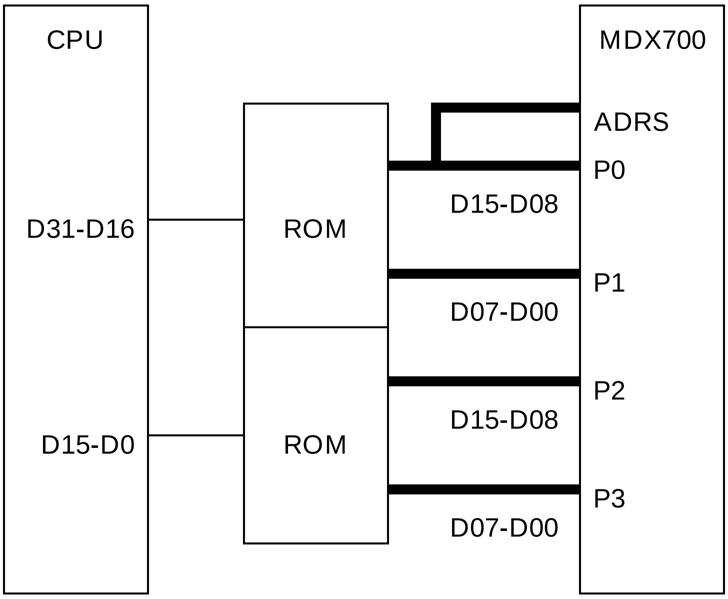
ROM read data bus width: 32 bits

ROM data bus width: 8 bits

Four ROM chips

CPU: little endian

Figure 2-13-12 Connecting the MDX700 to a target system (12)



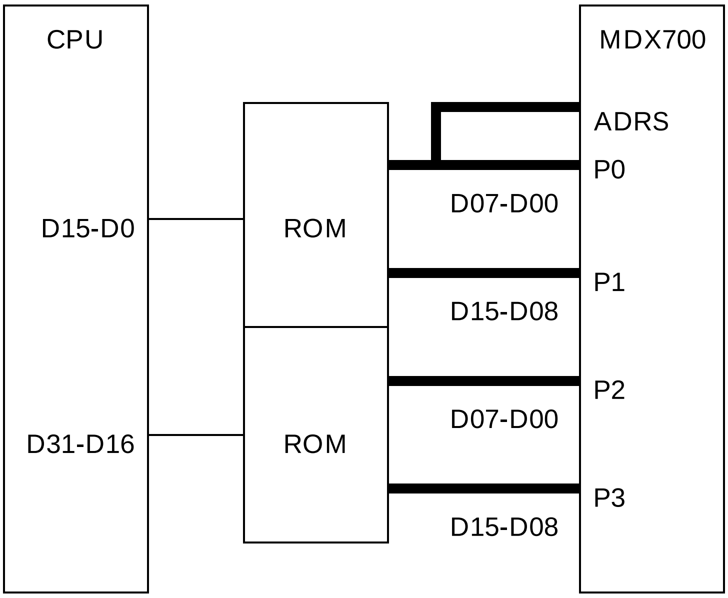
ROM read data bus width: 32 bits

ROM data bus width: 16 bits

Two ROM chips

CPU: big endian

Figure 2-13-13 Connecting the MDX700 to a target system (13)



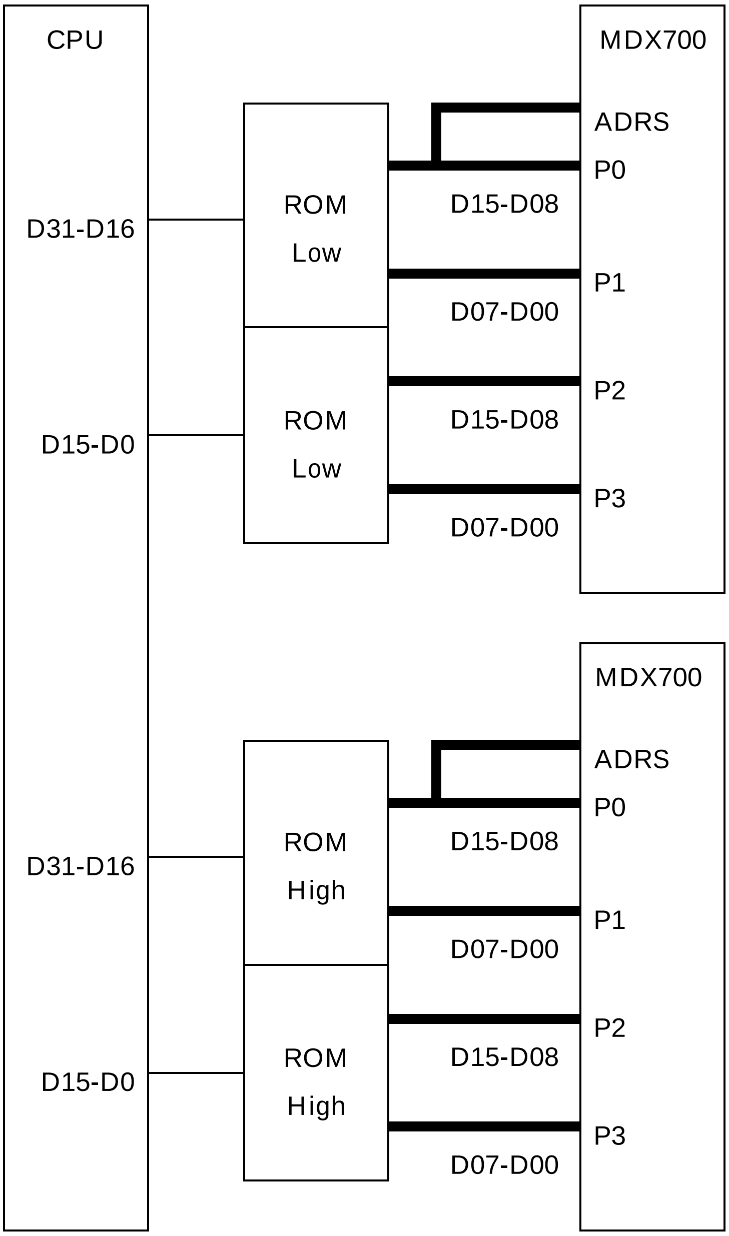
ROM read data bus width: 32 bits

ROM data bus width: 16 bits

Two ROM chips

CPU: little endian

Figure 2-13-14 Connecting the MDX700 to a target system (14)



Parallel

ROM read data bus width: 32 bits

ROM data bus width: 16 bits

Four ROM chips

CPU: big endian

Figure 2-13-15 Connecting the MDX700 to a target system (15)

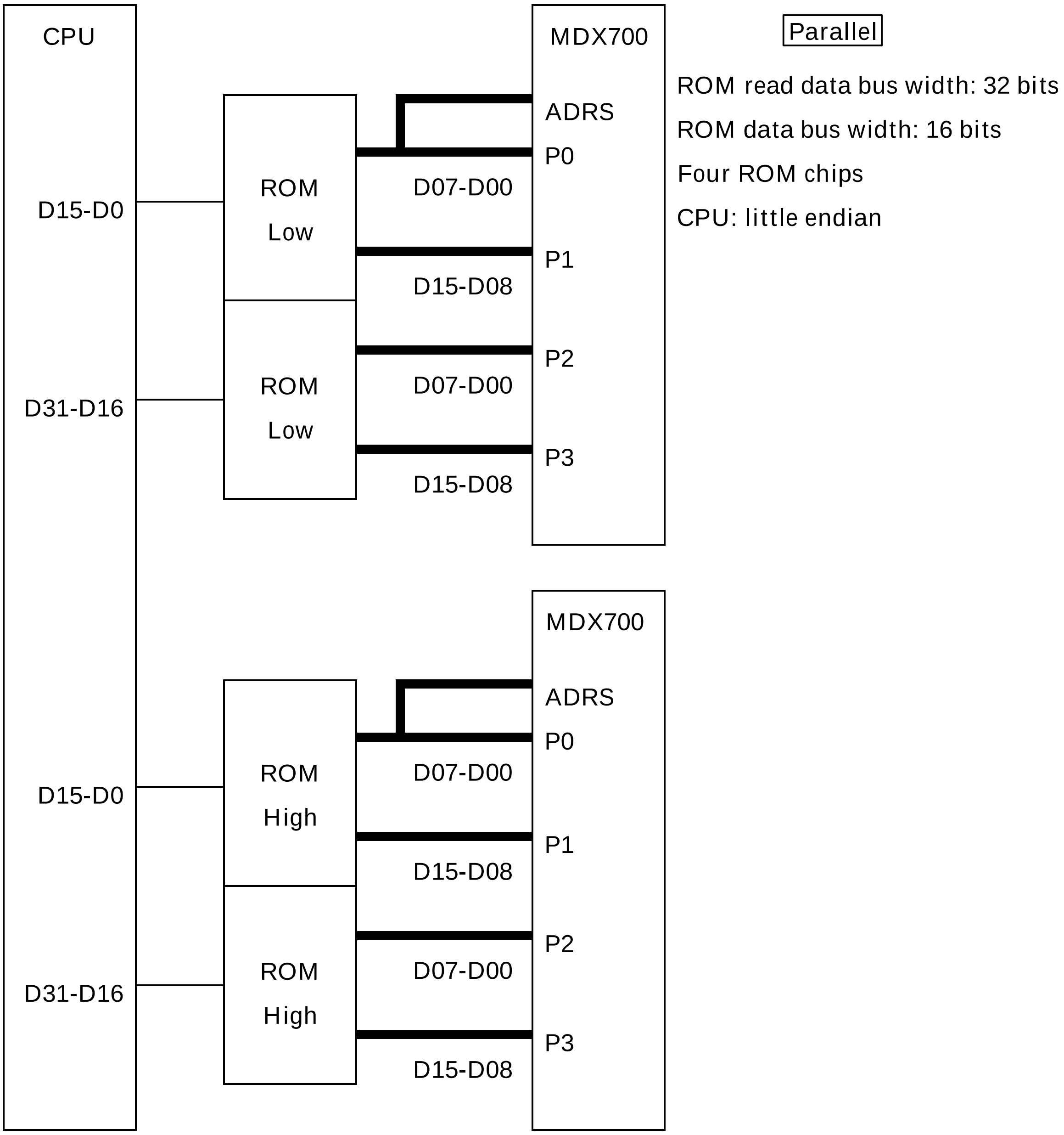
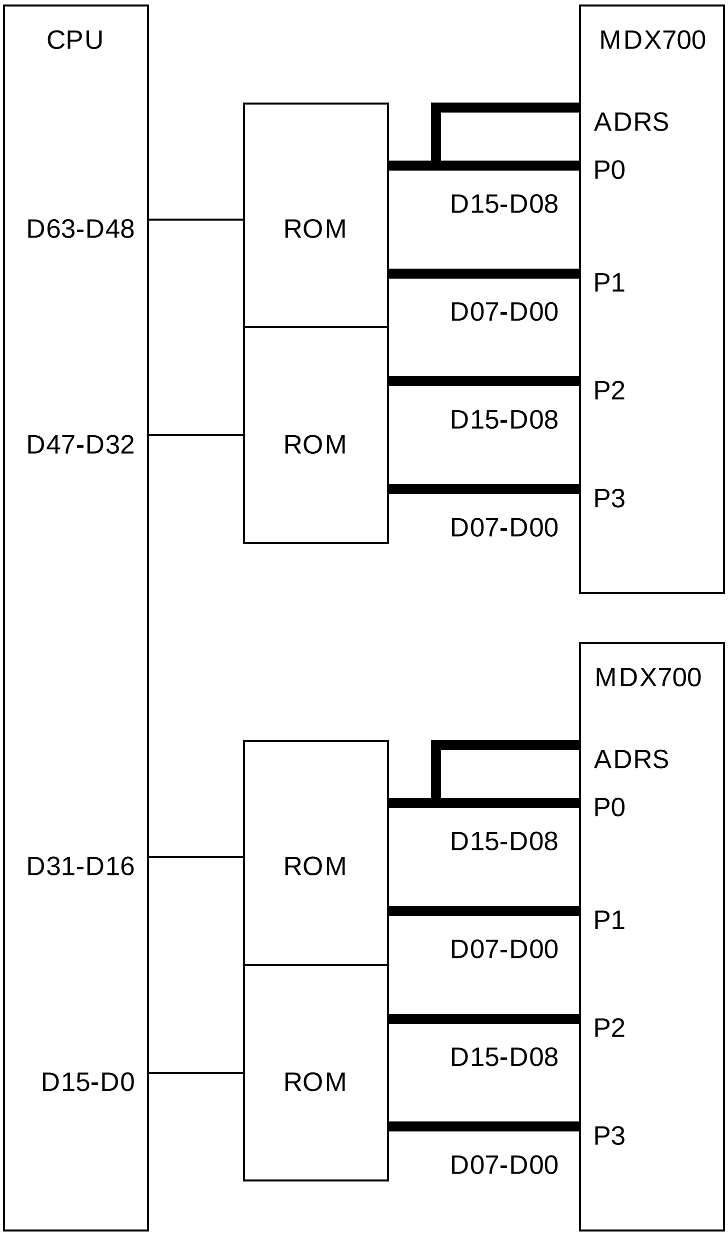


Figure 2-13-16 Connecting the MDX700 to a target system (16)



Parallel

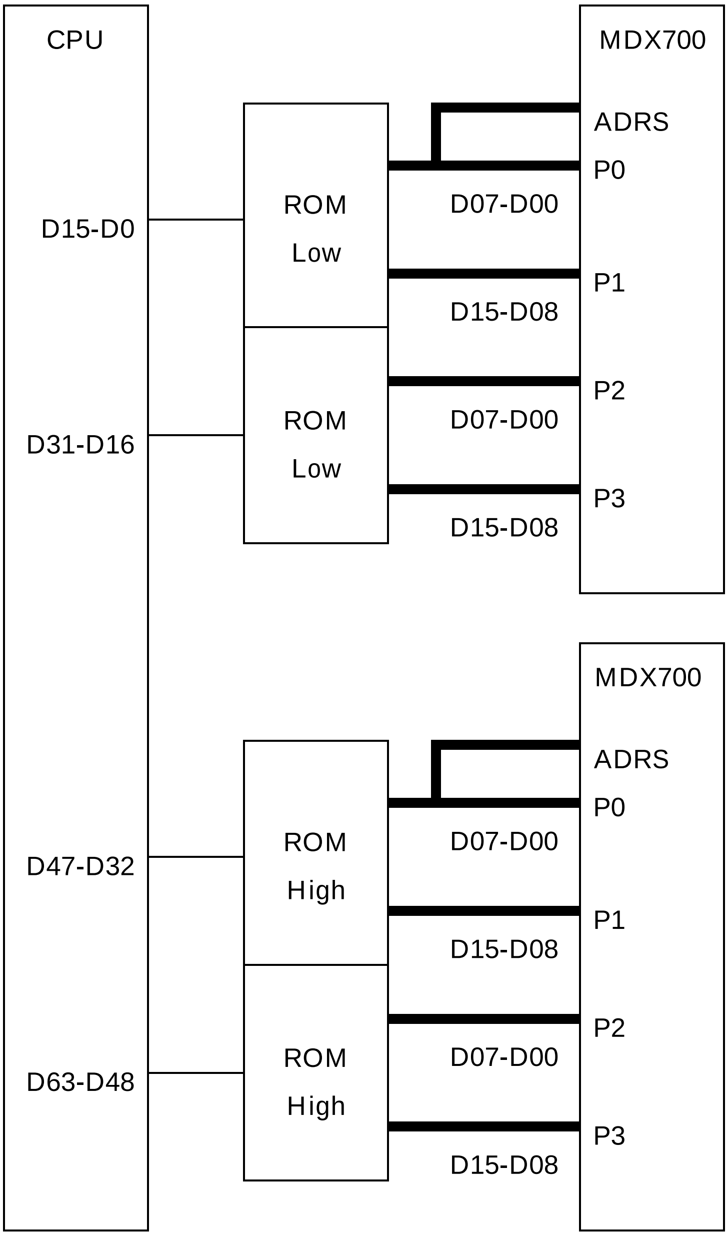
ROM read data bus width: 64 bits

ROM data bus width: 16 bits

Four ROM chips

CPU: big endian

Figure 2-13-17 Connecting the MDX700 to a target system (17)



Parallel

ROM read data bus width: 64 bits

ROM data bus width: 16 bits

Four ROM chips

CPU: little endian

Figure 2-13-18 Connecting the MDX700 to a target system (18)

2.7 Connecting an External Trigger Cable

External trigger cables connect the RESET and NMI signals that are output by the MDX700 to the target system. The EXT connector for the MDX700 should be connected to the RESET/NMI input circuit of the target system as described below.

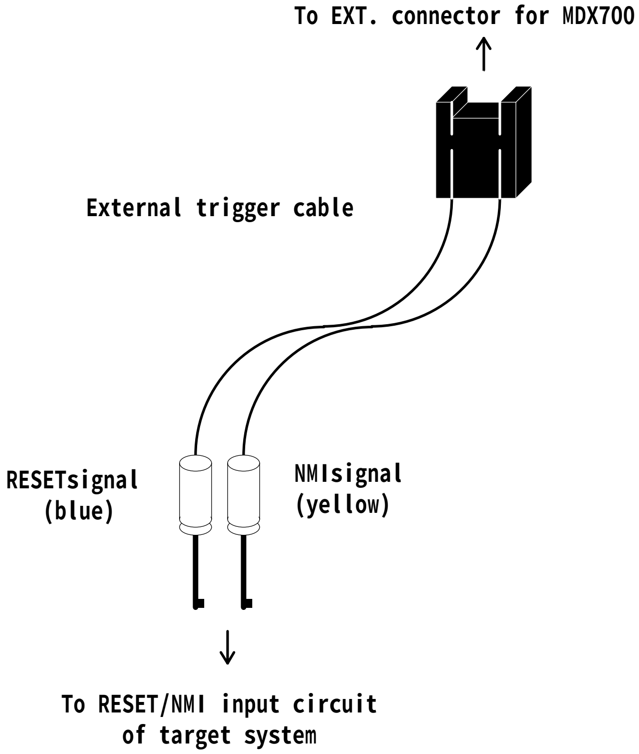


Figure 2-14 Connecting an external trigger cable

Connecting an optional external trigger cable can enhance the ease of the debugger operation.

RESET connected: Makes the debugger immediately available after it is launched.[1]

RESET not connected: The debugger is not immediately available after it is launched. To use the debugger, you need to perform manual operations, such as pressing the RESET switch for the target system (when the power is turned on).

[1] When connecting the RESET signal, set the **RESETVECTOR** in the configuration file (more on this later).

NMI connected: This allows you to apply a break on a user program during program execution by means of a debugger command. [*2]

NMI not connected: In this condition, you cannot apply a break to the user program by means of a debugger command. To apply a break, you need to perform manual operations, such as pressing the RESET switch for the target system.

The RESET and NMI signals are negative logic open collector outputs (equivalent to the 7406). Before these signals can be connected, the target system signal must be in a pulled-up state. The MDX700 assumes that the RESET and NMI signals are connected to the following target system circuit:

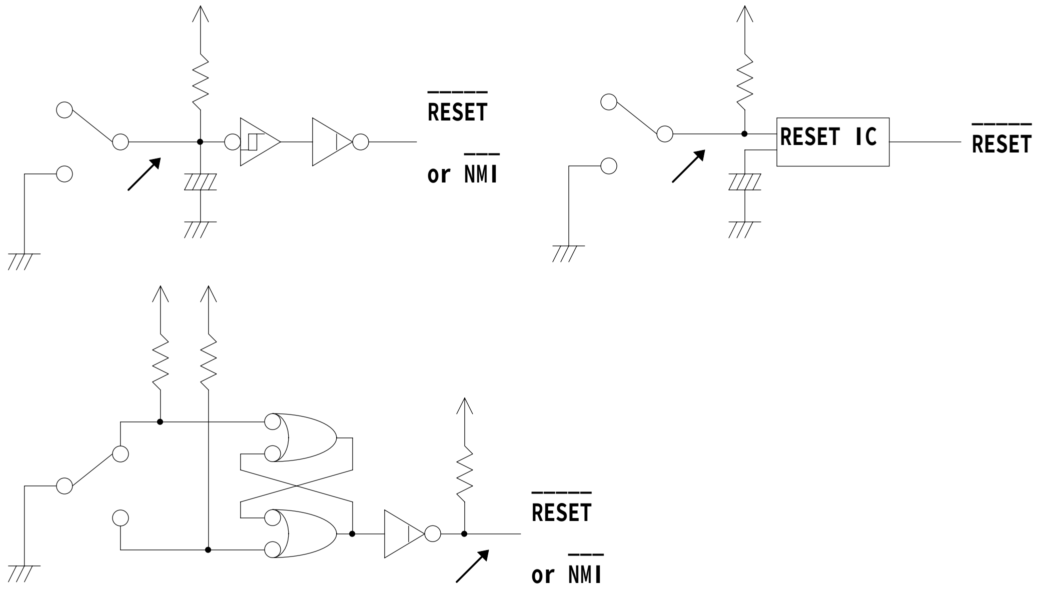

$\overline{\text{RESET}}$
or $\overline{\text{NMI}}$
RESET IC
$\overline{\text{RESET}}$
$\overline{\text{RESET}}$
or $\overline{\text{NMI}}$

Figure 2-15 The target system circuit that can accept the connection of an external trigger cable

[*2] In the case of a $\boxed{\text{PowerPC}}$ and when connecting the NMI signal, set the **ABORTVECTOR** in the configuration file (more on this later).

Conversely, an external trigger cable cannot be connected to the following type of target system circuit:

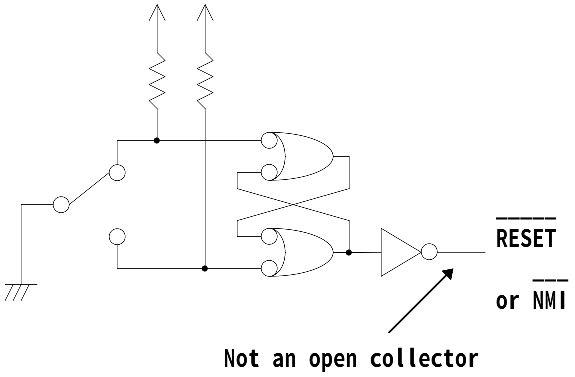


Figure 2-16 The target system circuit that cannot accept the connection of an external trigger cable

2.8 Power-Up Procedures

After connecting the MDX700 to the host or to the target system, connect the power cable for the MDX700.

The different pieces of equipment should be powered up in the following sequence:

1. Host
2. MDX700
3. Target system

Likewise, the equipment should be powered down in the following sequence:

1. Target system
2. MDX700
3. Host

IMPORTANT: Incorrect power-up/down procedures can damage the equipment.

IMPORTANT: Do not connect or disconnect devices when switching the power on.

CHAPTER 3 INSTALLING A DEBUGGER

This chapter explains how to install a debugger.

Different debuggers require different installation procedures. Please consult the section in the manual, along with the release notes, that pertains to the debugger that you are using.

Installing the MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95

1. Both the compiler and the MULTI program should be installed (see the Installation Procedures available from Green Hills Software).

2. Insert the floppy disk for the MDXSERV into the diskette drive.

3. Use Windows Explorer to copy the contents of the diskette to the directory in which the MULTI is already installed. Type the following command when using an MS-DOS compatible window (assuming that the diskette drive is A:, and the MULTI is installed in the C:\GREEN directory):

```
C:\>copy a:*.* c:\green
```

4. Use Windows Explorer to copy the file with the .cfg extension to create an mdx.cfg file. Type the following commands when using an MS-DOS compatible window:

```
C:\>cd c:\green
C:\GREEN>copy mdx_68k.cfg mdx.cfg          68000
C:\GREEN>copy mdx_arm.cfg mdx.cfg          ARM
C:\GREEN>copy mdx_mips.cfg mdx.cfg         MIPS
C:\GREEN>copy mdx_ppc.cfg mdx.cfg          PowerPC
C:\GREEN>copy mdx_sh.cfg mdx.cfg           SH
C:\GREEN>copy mdx_v800.cfg mdx.cfg         V800
```

Installing the MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1

This requires the same procedures as installing the MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95.

Installing the MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris

1. Both the compiler and the MULTI program should be installed (see the Installation Procedures available from Green Hills Software).

2. Insert the floppy disk for the MDXSERV into the diskette drive.

3. Copy the contents of the diskette to the directory in which the MULTI is already installed. (assuming that the diskette drive is /dev/rmt/0, and the MULTI is installed in the /home/greendirectory):

   ```
   % cd /home/geen
   % tar -xvf /dev/rmt/0
   ```

4. Copy the file with the .cfg extension to create an mdx.cfg file. Type the following commands:

   ```
   % cp mdx_68k.cfg mdx.cfg         68000
   % cp mdx_arm.cfg mdx.cfg         ARM
   % cp mdx_mips.cfg mdx.cfg MIPS
   % cp mdx_ppc.cfg mdx.cfg         PowerPC
   % cp mdx_sh.cfg mdx.cfg          SH
   % cp mdx_v800.cfg mdx.cfg V800
   ```

Installing the SingleStep 6.5 68K on Windows 3.1

1. Insert the floppy disk for the SingleStep 68K 6.5(1/2), labeled "AIC", into the diskette drive.

2. Use the File Manager to run the INSTALL.EXE program from the diskette. Follow the directions that are displayed on the screen.

3. Insert the floppy disk for the SingleStep 6.5 for MDX700/68K into the diskette drive.

4. Use the File Manager to copy the contents of the floppy disk to the directory in which the SingleStep is already installed. Type the following command when using an MS-DOS compatible window (assuming that the diskette drive is A:, and the SingleStep is installed in the C:\SDS65 directory):

   ```
   C:\>copy a:*.* c:\sds65\cmd
   ```

5. Run the MDX68KP.EXE[*1] program. Type the following command when using an MS-DOS compatible window (assuming that the SingleStep is installed in the C:\SDS65 directory):

   ```
   C:\>cd sds65\cmd
   C:\SDS65\CMD>mdx68kp
   ```

6. Use the Program Manager to register the MDX68K.EXE program in an icon.

[*1] The MDX68KP.EXE is a self-expanding program that creates program MDX68K.EXE after execution.

Installing the SingleStep 6.5 PowerPC on Windows 3.1

1. Insert the floppy disk for the SingleStep PowerPC 6.5(1/2), labeled "AIC", into the diskette drive.

2. Use the File Manager to run the INSTALL.EXE program from the diskette. Follow the directions that are displayed on the screen.

3. Insert the floppy disk for the SingleStep 6.5 for MDX700/PowerPC into the diskette drive.

4. Use the File Manager to copy the contents of the floppy disk to the directory in which the SingleStep is already installed. Type the following command when using an MS-DOS compatible window (assuming that the diskette drive is A:, and the SingleStep is installed in the C:\SDS65 directory):

```
C:\>copy a:*.* c:\sds65\cmd
```

5. Run the MDXPPCP.EXE[*1] program. Type the following command when using an MS-DOS compatible window (assuming that the SingleStep is installed in the C:\SDS65 directory):

```
C:\>cd sds65\cmd
C:\SDS65\CMD>mdxppcp
```

6. Use the Program Manager to register the MDXPPC.EXE program in an icon.

[*1] The MDXPPCP.EXE is a self-expanding program that creates program MDXPPC.EXE after execution.

Installing the XHI68KMD(XRAY68K 2.2a) on an MS-DOS PC/AT

1. Insert the floppy disk containing the XHI68KMD into the diskette drive.

2. Type the following command (assuming that the diskette drive is A:, and the hard disk is C:):

```
C:\>mkdir c:\xhi68kmd
C:\>copy a:*.* c:\xhi68kmd
```

3. Add the entry c:\xhi68kmd to the PATH environment variable.

```
C:\>path c:\bin;c:\utils;c:\xhi68kmd
```

Installing the XHI68KMD(XRAY68K 2.2a) on an MS-DOS PC-98

1. Insert the floppy disk containing the XHI68KMD into the diskette drive.

2. Type the following command (assuming that the diskette drive is A:, and the hard disk is C:):

```
A:\>mkdir a:\xhi68kmd
A:\>copy c:*.* a:\xhi68kmd
```

3. Add the entry c:\xhi68kmd to the PATH environment variable.

```
A:\>path a:\bin;a:\utils;a:\xhi68kmd
```

4. Set the environment variable DOS16M.

```
A:\>set dos16m=1@2m-5m
```

Installing the XHI68KMD(XRAY68K 3.4) on SunOS/Solaris

1. Insert the cartridge tape for the XHI68KMD into the tape drive.

2. Type the following commands (assuming that the cartridge drive is /dev/rmt/0):

```
% mkdir /home/xhi68kmd
% cd /home/xhi68kmd
% tar -xvf /dev/rmt/0
```

3. If MasterWorks is already installed, type the following commands:

```
% cp master/bin/mdx* /usr/mri/master/bin
% cp master/bin/xhi68kmd /usr/mri/master/bin
% cp master/bin/xsi68kmd /usr/mri/master/bin
% cp master/bin/xsi68kmd.hlp /usr/mri/master/help
```

4. If MasterWorks has not been installed, type the following commands:

```
% setenv PATH .:/usr/ucb:/usr/bin:/usr/openwin/bin:/home/xhi68kmd/master/bin
% setenv XRAYMASTER /home/xhi68kmd/master
% setenv LD_LIBRARY_PATH /usr/openwin/lib:/home/xhi68kmd/master/lib
```

Installing the MDXDEB 3.5 on MS-DOS/Windows 3.1/95

1. Insert the floppy disk containing the MDXDEB program into the diskette drive.

2. Use Windows Explorer to create a directory, and copy the contents of the floppy disk to the directory. Type the following commands when using an MS-DOS compatible window (assuming that the diskette drive is A:, and the program is copied to a directory C:\MDXDEB):

```
C:\>mkdir c:\mdxdeb
C:\>copy a:*.* c:\mdxdeb
```

3. When operating the debugger in an MS-DOS compatible window, add the entry c:\mdxdeb to the PATH environment variable.

```
C:\>path c:\bin;c:\utils;c:\mdxdeb
```

4. When operating the debugger in Windows, use the Explorer to create a shortcut for MDXDEBW.EXE on the desktop by specifying a configuration file in an argument in the command line:

```
c:\mdxdeb\mdxdeb.exe c:\mdxdeb\mdx.cfg
```

Installing the MDXDEB 3.5 on SunOS/Solaris

1. Insert the floppy disk containing the MDXDEB program into the diskette drive.

2. Create a directory. Copy the contents of the floppy disk to the new directory by typing the following commands (assuming that the diskette drive is /dev/rfd0a, and the program is to be copied to a /home/mdxdeb directory):

```
% mkdir /home/mdxdeb
% cd /home/mdxdeb
% tar -xvf /dev/rfd0a
```

3. Add the entry /home/mdxdeb to the PATH environment variable.

```
% setenv PATH .:/usr/ucb:/usr/bin:/usr/openwin/bin:/home/mdxdeb
```

CHAPTER 4 SETTING THE ENVIRONMENT FOR THE DEBUGGER

This chapter explains how to set the environment necessary for using a debugger, with particular attention to configuration file (mdx.cfg) modification procedures.

IMPORTANT: Exercise care when setting the environment. An incorrect environment setting can disable the debugger.

4.1 Setting a Debugger Environment by Means of a Configuration File

You can set a debugger environment by setting up an appropriate configuration file.

Configuration files allow you to specify the requisite operating environment for the MDX700 and target systems. Before using the debugger, you need to create a configuration file for each target system that you intend to use.

The debugger that is supplied includes the following sample configuration file (file name: mdx.cfg):

```
*
*  MDX700 configuration
*
MONITOR         mdxsh.abs       ; monitor program (or mdxshl.abs)
CPU             SH7604          ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             16              ; bus width
ROM             0x00000000      ; ROM start address
ROMSIZE         0x00100000      ; ROM size
WORKROM         0x000FE000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x001FF000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0x00000000      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R15         0x00180000      ; stack pointer
```

Figure 4-1 Sample configuration file (mdx.cfg)[*1]

Normally, you can create the configuration file for a specific target system by modifying the mdx.cfg file.[*2]

Configuration File Syntax

1. Each field is specified in a line. For example, the line "**BUS 16 ;bus width"** assigns the value 16 to the field **BUS**. The entries following the semicolon (;) represent comments.
2. A field is distinguished from a specification value by means of either space or a tab.
3. Hexadecimal numbers are indicated by the prefix 0x.
4. A comment line begins with the letter "*". The pound sign (#) and the semicolon (;) can

[*1] Figure 4-1 is intended as a sample for $\boxed{\text{SH}}$. Actual settings may vary from one CPU to another.

[*2] The configuration file is a text file, which can be modified using the [NotePad] in Windows and [vi] in UNIX.

be substituted for an asterisk (*).

NOTE: When modifying the configuration file, be careful not to delete any lines other than comment lines

NOTE: When modifying the configuration file, be careful not to change fields.

For details on configuration file fields, see Section 4.2, "Configuration File Fields".
Modifying a configuration file requires the following pieces of information:

- The I/O address of the parallel interface board Parallel
- CPU name
- ROM read data bus width
- ROM start address and capacity
- ROM area that can be made available to the monitor program [*3]
- RAM area that can be made available to the monitor program

[*3] An area that can be allocated to the monitor program refers to one which is not used by a user program.

4.2 Configuration File Fields

MONITOR **Name of the monitor program file** --- Specify the file name according to the CPU of the target system.

mdx68k.abs	68K
mdxarm.abs	ARM Big endian
mdxarml.abs	ARM Little endian
mdxmips.abs	MIPS Big endian
mdxmipsl.abs	MIPS Little endian
mdxppc.abs	PowerPC
mdxsh.abs	SH Big endian
mdxshl.abs	SH Little endian
mdxv850.abs	V850 V850E
mdxv800.abs	V800 except V850 or V850E

CPU **CPU name** --- Specify the CPU name of the target system. See Appendix F, "Supported CPUs".

PORT **I/O address of the parallel interface board** Parallel --- Specify the desired I/O address according to the host system being used.[*1]

0x0100	PC/AT
0x01D0	PC-98

If the I/O address of the parallel interface board has been changed irrespective of the host system, specify the new I/O address.

If two MDX700s are connected to a target system, as illustrated in Figures 2-13-15 ... 2-13-18, the two I/O addresses should be specified in a 32-bit format. Specify the I/O address of the MDX700 that is connected to the lower side of the address or the data bus in the low 16 bits; specify the I/O

[*1] In the case of Ethernet, **PORT** is ignored.

address of the upper MDX700 in the high 16 bits. Example: If the upper I/O address is 0x0120 and the lower I/O address is 0x0100, specify **0x01200100.**

BUS

ROM read data bus width --- Specify the width of the data bus (8, 16, 32, or 64) through which the ROM for the target system is read. Normally, the CPU data bus width is specified.

NOTE: In target systems with a ROM bus sizing in effect, do not specify a CPU data bus width.

Example: For a system in which the ROM is read four times using an 8-bit data bus width to generate 32-bit data before the data is fed into the CPU, specify the value 8.

ROM

ROM start address --- Specify the starting address of the ROM for the target system. [2]

If the target system contains multiple ROM chips, specify the starting address of the ROM that is mapped to the lowest address. Example: Specify **0x00100000** if the target system contains a ROM that begins at address 0x00100000 and another ROM that begins at 0x00180000.

NOTE: Not not specify a cache area address. MIPS V800

ROMSIZE

ROM capacity (in bytes) --- Specify the capacity of the target system ROM in bytes.[3]

If the target system contains multiple ROM chips, specify the total capacity of the ROM chips. Example: Specify **0x00100000** if the target system contains two 512Kb ROM chips.

[2] The ROM area to which the MDX700 is not connected should not be included in the calculation.

[3] For ROM capacity, see Appendix D, "Supported ROMs".

If two MDX700s are connected to the target system by using a ROM read data bus width of 32 bits, as illustrated in Figures 2-13-15 ... 2-13-16, specify a value that represents the total capacity of ROM chips plus 0x20000000. Example: If the total capacity of ROM chips is 0x00400000, specify a value **0x20400000.**

WORKROM **Monitor program ROM start address** --- Specify the starting address of the ROM that can be made available to the monitor program.

In situations where the last ROM area is allocated to the monitor program, specify a value that represents ROM final address +1 minus **WORKROMSIZE**. Example: If the ROM final address is 0x03FFFFFF and the **WORKROMSIZE** is 0x00004000, specify a value **0x03FFC000**.

NOTE: Non-ROM area addresses should not be specified.

WORKROMSIZE **Monitor program ROM size (in bytes)** --- Specify the capacity of the ROM, in bytes, that can be allocated to the monitor program. Normally, the sample value that is provided in the configuration file should be used.

WORKRAM **Monitor program RAM start address** --- Specify the starting address of the RAM that is allocated to the monitor program. This address should be one that does not require initialization if possible. [*4]

WORKRAMSIZE **Monitor program RAM size (in bytes)** --- Specify the capacity of the RAM, in bytes, that can be allocated to the monitor program. Normally, the sample value that is provided in the configuration file should be used.

RESETVECTOR **Address of the reset exception vector** --- Specify the address of the reset exception vector that is read after the target system is reset. Normally you can specify the same value as the **ROM** address.

In the case of V800, specify the value that is obtained by subtracting

[*4] When specifying a RM that requires initialization, you must modify the monitor program.

0x0000000F from the final address of ROM.

If the reset exception vector is outside the ROM area, specify a value **0xFFFFFFFF** (in situations where another monitor program is running on the target system).

NOTE: Non-ROM area addresses should not be specified.

ABORTVECTOR **Address of the abort vector break exception vector** ARM PowerPC --- Specify the address of the exception vector that breaks and aborts user programs. [*5]

TIMER **Software timer value** Parallel --- Specify the software timer value to be used by the debugger. Normally, you can use the sample value stored in the configuration file. [*6]

The timer value should be increased only in situations where a "**communication port timeou**t" error occurs when the debugger is started or memory access is made.

REG_xxx **Initial register values (optional)** --- Specify the initial values for the registers that are required by the user program. In the following, the letters "xxx" denote a register name. For register names, see Appendix G, "Register Names". [*7]

Initial values should be specified for the following registers at least. For other registers, the sample initial values that are provided in the configuration file can be used.

- Specify the initial value for the stack pointer at a RAM address.
- Specify the initial value for the VBR register at the starting address of

[*5] In the case of a 68000, a Level 7 interrupt is used for an abort/break. In the case of MIPS SH V800, the NMI is used for an abort/break.

[*6] In the case of Ethernet, any **TIMER** values are ignored.

[*7]Register initialization values cannot be specified for the monitor program.

exception vector. 68000 SH

- In situations where the address of the exception vector is 0x0000_0000, specify the initial value for the MSR register as **0x2000.** For an address 0xFFF0_0000, specify an initial value **0x2040.** PowerPC except 403
- Specify the initial value for the VBR register at the starting address of exception vector. PowerPC 403

4.3 Modifying the Monitor Program

In most cases, you can set the operating environment for the debugger simply by modifying the configuration file. However, depending on the requirements imposed by the target system, you may also need to modify the monitor program.

A target system may require a change in the monitor program under the following conditions:

- Access to RAM[*1] requires the initialization of the DRAM controller.
- Access to RAM requires the initialization of internal registers in the CPU.

In the case of these target systems, use the following procedures to add an initialization code for RAM access to the monitor program:

1. Use the Editor to open the source file for the monitor program. [*2]
2. Use the assembly language to enter the initialization code at the USER_INIT label, which is located at the end of the source file for the monitor program.
3. Close the Editor.
4. Assemble the source file for the monitor program to create an executable file. [*3]

NOTE: The address to which the monitor program is loaded changes dynamically with the value of the **WORKROM** parameter specified in the configuration file. Therefore, any initialization code, as the monitor program, should be coded in a relocatable code (position-independent code).

[*1] Strictly speaking, this refers to the RAM area specified by the **WORKRAM** and **WORKRAMSIZE** parameters.

[*2] For the names of monitor program source files, see the debugger release notes.

[*3] Different debuggers require different assembing procedures. For assembling procedures, see the debugger release notes or comments in the monitor program source files.

4.4 Adding an Initialization Code by Modifying the Configuration File

Initialization code can be added by modifying either the source file for the monitor program or the configuration file.

In revising the monitor program, an initialization code can be coded in the assembly language. The method that involves the modification of the configuration file, however, can allow the coding of an initialization code only in machine language written in hexadecimal.

The method of revising the configuration file, while requiring complex procedures, offers the advantage of not requiring a repeat modification of the monitor program when the monitor program is upgraded.

The following describes the procedures for adding an initialization code through the modification of the configuration file:

1. With the Editor, open a source file (new file) for the assembler.
2. Type your initialization code in assembly language.
3. At the end of the initialization code, type a return instruction from the subroutine. [*1]
4. Close the Editor.
5. Specify the option to generate a list file, and assemble the source file.
6. Add the machine language code that is output to the list file to the configuration file.

For entering an initialization code in the configuration file in machine language, use an **INIT_CODE** item as shown below. The size of a machine language entry is the smallest instruction size that is supported by the CPU. Machine language entries should be coded starting from the highest bit, irrespective of the endian of the CPU.

> NOTE: The initialization code that is specified in the configuration file takes precedence over the initialization code supplied in the monitor program source file.

[*1] In the case of the 68000, the ***jmp (a6)*** instruction is used instead of the ***rts*** instruction.

68000

```
INIT_CODE       0x41F9          ; lea     $1F800,A0
INIT_CODE       0x0001          ;
INIT_CODE       0xF800          ;
INIT_CODE       0x103C          ; move.b  #0,D0
INIT_CODE       0x0000          ;
INIT_CODE       0x1080          ; move.b  d0,(A0)
INIT_CODE       0x4ED6          ; jmp     (A6)
```

ARM

```
INIT_CODE      0xe59f0008       ; ldr     r0, .+16
INIT_CODE      0xe3a01003       ;  mov     r1, 3
INIT_CODE      0xe5c01000       ;  strb    r1, [r0]
INIT_CODE      0xe1a0f00e       ;  mov     pc, lr
INIT_CODE      0xffffa404       ;  .data.w 0xFFFFA404
```

MIPS

```
INIT_CODE       0x3c06ab00      ; lui     $6, 0xAB00
INIT_CODE       0x34c600a2      ; ori     $6, $6, 0x00A2
INIT_CODE       0x34070006      ; ori     $7, $0, 0x0006
INIT_CODE       0xa4c70000      ; sh      $7, 0($6)
INIT_CODE       0x03e00008      ; jr      $ra             # return
INIT_CODE       0x00000000      ; nop     # in delay slot
```

PowerPC

```
INIT_CODE       0x3d4000ff      ; lis     r10, 0x00FF
INIT_CODE       0x39600000      ; li      r11, 0x0000
INIT_CODE       0x994a4000      ; stb     r10, 0x4000(r10)
INIT_CODE       0x4e800020      ; blr
```

SH

```
INIT_CODE       0xD002          ; mov     #0x0480007C,r0
INIT_CODE       0xE100          ; mov     #0x0,r1
INIT_CODE       0x2012          ; mov.l   r1, @r0
INIT_CODE       0x000B          ; rts
INIT_CODE       0x0009          ; nop
INIT_CODE       0x0009          ; nop
INIT_CODE       0x0480
INIT_CODE       0x007C
```

V800

```
INIT_CODE       0x5640          ; movhi 0x0028, zero, r10
INIT_CODE       0x0028          ;
INIT_CODE       0x5E80          ; ori 0x0090, zero, r11
INIT_CODE       0x0090          ;
INIT_CODE       0x5F4A          ; st.b r11, 0x0006[r10]
INIT_CODE       0x0006          ;
INIT_CODE       0x007F          ; jmp [lp]
```

CHAPTER 5 LAUNCHING THE DEBUGGER

This chapter explains how to launch a debugger.

Different debuggers may require different startup methods. See the section in this manual that pertains to the debugger that you are using. The relevant debugger manual and release notes should also be consulted.

If the RESET signal is connected to the target system, refer to the applicable debugger startup method.

If the RESET signal is not connected to the target system, launch the debugger by performing the following procedures: [*1]

1. Start the dubugger.
2. A timeout error is displayed.
3. Press the reset switch on the target system (or turn on the power).
4. Re-initialize the debugger.

Starting the MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95

1. Start the MULTI.
2. Enter the following command to remotely connect the server program MDXSERV. [*2]

```
remote mdxserv
```

- When re-initializing the debugger, enter the *remote* command again.
- The configuration file mdx.cfg that exists in the same directory as the MDXSERV is retrieved. If it is desired to specify the configuration file explicitly, enter the *remote* command as shown below:

```
remote mdxserv c:\green\mdx.cfg
```

[*1] This procedure is required only after the power is turned on.

[*2] When the system is in the builder mode, specify *mdxserv* as a server name, and then press the REMOTE button.

Starting the MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1

This requires the same startup procedures as "Starting the MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95".

Starting the MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris

1. Start the MULTI.
2. Enter the following command to remotely connect the server program MDXSERV.[*1]

```
remote mdxserv
```

- When re-initializing the debugger, enter the *remote* command again.
- The configuration file mdx.cfg that exists in the same directory as the MDXSERV is retrieved. If it is desired to specify the configuration file explicitly, enter the *remote* command as shown below:

```
remote mdxserv /home/green/mdx.cfg
```

- When specifying a host name other than *mdx*, use the -h option as follows:

```
remote mdxserv -h mdx1
```

[*1] When the system is in the builder mode, specify *mdxserv* as a server name, and then press the REMOTE button.

Starting the SingleStep 6.5 68K on Windows 3.1

Start the MDX68K.EXE program. If an associated icon is already registered using the Program Manager, double-click on the icon.

- When re-initializing the debugger, close the debugger and restart it.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the environment variable MDX_CFG, and then execute the MDX68K.EXE program.

```
C:\>set MDX_CFG=c:\sds65\cmd\mdx.cfg
```

Starting the SingleStep 6.5 PowerPC on Windows 3.1

1. Start the MDXPPC.EXE program. If an associated icon is already registered using the Program Manager, double-click on the icon.

- When re-initializing the debugger, close the debugger and restart it.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the environment variable MDX_CFG, and then execute the MDXPPC.EXE program.

```
C:\>set MDX_CFG=c:\sds65\cmd\mdx.cfg
```

Starting the XHI68KMD(XRAY68K 2.2a) on an MS-DOS PC/AT

1. Start the XHI68KMD.EXE program. Enter the following command:

```
C:\>xhi68kmd
```

- When re-initializing the debugger, close the debugger and restart it.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the environment variable MDX_CFG, and then execute the XHI68KMD.EXE program.

```
C:\>set MDX_CFG=c:\sds65\cmd\mdx.cfg
```

Starting the XHI68KMD(XRAY68K 2.2a) on an MS-DOS PC-98

This requires the same startup procedure as "Starting the XHI68KMD(XRAY68K 2.2a) on an MS-DOS PC/AT".

Starting the XHI68KMD(XRAY68K 3.4) on SunOS/Solaris

1. Start the xhi68kmd program. Enter the following command:

```
% xhi68kmd
```

- When re-initializing the debugger, close the debugger and restart it.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the environment variable MDX_CFG, and then execute the xhi68kmd program.

```
% setenv MDX_CFG /home/green/mdx.cfg
```

Starting the MDXDEB 3.5 on MS-DOS

1. Run the MDXDEB.EXE program. Enter the following command.

```
C:\>mdxdeb
```

- When re-initializing the debugger, enter the I command.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the name of the file after *mdxdeb*.

```
C:\>mdxdeb c:\mdxdeb\mdx.cfg
```

Starting the MDXDEB 3.5 on Windows 3.1/95

1. Double-click on the shortcut for the MDXDEBW.EXE program.

- When re-initializing the debugger, enter the I command.
- The configuration file specified in the command line argument is retrieved.

Starting the MDXDEB 3.5 on SunOS/Solaris

1. Run the *mdxdeb* program. Enter the following command:

```
% mdxdeb
```

- When re-initializing the debugger, enter the I command.
- The configuration file *mdx.cfg* that exists in the directory specified by the environment variable PATH is retrieved. For explicitly specifying the configuration file, specify the name of the file after the *mdxdeb* command.

```
% mdxdeb /home/mdxdeb/mdx.cfg
```

CHAPTER 6 MDXDEB COMMANDS

This chapter explains how to use the commands for the quick debugger MDXDEB.

Debuggers other than the MDXDEB support some of the MDXDEB commands as extended commands. Details may be found in the relevant debugger release notes.

A *addr* MIPS

Changes memory contents in the assembly language.

addr: Starting memory change address (hexadecimal)

Example:

```
> A A0000000
A0000000 add r1,r2,r3
A0000004 or r4,r5,r6
A0000008  (terminated with a carriage return only)
```

B

Displays a breakpoint.

Example:

```
> B
0 0000800C         (The first digit represents the breakpoint number.)
1 0001E8D4         (The succeding digits represent the breakpoint address.)
```

B *addr*

Sets breakpoints. A maximum of 64 breakpoints can be set on instruction.

addr: Breakpoint address (hexadecimal)

Example:

```
> B 1E8D4
```

B/C {*num*|*}

Clears breakpoints.

num: Breakpoint number (decimal)

*****: All breakpoints

Example:

```
> B/C 10           (Clears breakpoint #10.)
> B/C *            (Clears all breakpoints.)
```

C

Displays the configuration of the MDX700.

D[/B/W/L] ***addr1*****[,*****addr2*****]**

Displays memory contents. If ***addr1*** is omitted, displays the contents of memory continuously. If ***addr2*** is omitted, diplays 64 bytes of memory at a time.

/B: 8 bits **/W:** 16 bits **/L:** 32 bits

addr1: Starting memory display address (hexadecimal)

addr2: Ending memory display address (hexadecimal)

Example:

```
> D/B 1000
> D/L 2000,20FF
> D
```

E[/B/W/L] ***addr=data***

Modifies the contents of memory. If ***=data*** is omitted, modifies memory contents interactively.

/B: 8 bits **/W:** 16 bits **/L:** 32 bits

addr: Starting memory change address (hexadecimal)

data: Memory change data (hexadecimal)

Example:

```
> E/B 1000=55
> E/W 3000=11,22,33          (data can be delimited with commas.)
> E/L 2000
00002000 00000000 11223344
00002004 00000000 55667788
00002008 00000000 .          (Terminates with a period.)
```

F[/B/W/L] ***addr1,addr2,data***

Fills memory.

/B: 8 bits **/W:** 16 bits **/L:** 32 bits

addr1: Memory fill starting address (hexadecimal)

addr2: Memory fill ending address (hexadecimal)

data: Fill data (hexadecimal)

Example:

```
> F/B 0,3FF,FF
> F/L 1000,1FFF,0
```

G [*addr*]

Executes a user program. If ***addr*** is omitted, executes the user program from the current PC. If the program execution fails to halt at a breakpoint, press any key. The user program can be halted by issuing the NMI signal. [*1]

addr: User program starting address (hexadecimal)

Example:

```
> G
> G 1000
```

H

Displays a help message.

I [*config*]

Outputs the RESET signal and re-initializes the MDX700. If ***config*** is omitted, re-initializes the MDX700 by using the current configuration file.

config: The configuration file with which the MDX700 is to be initialized.

Example:

```
> I
> I c:\target1\mdx.cfg
```

K

Tests the MDX700's emulation memory. If no errors are found, displays an OK upon completion of the test. If an error is found, displays the address at which the error occurred, the affected data, and the expectation data.

L *file*[,*offset*]

Downloads MDX binary files, S-record files, Intel hexa files, or COFF files onto memory. The file format is automatically recognized. If a file name extension is omitted, the .mdx extension is supplied.

file: The file to be downloaded

offset: Offset address (ignored in the case of MDX binary files)

Example:

```
> L prog1.abs
> L prog1.abs,2000
> L prog2.mdx
```

[*1] **If the NMI signal is not connected to the target system, the user program cannot be halted by means of key operations. To halt the user program, use the ABORT switch on the system.**

M ***addr1,addr2,addr3***

Block-transfers memory contents.

addr1: Source memory starting address (hexadecimal)

addr2: Source memory ending address (hexadecimal)

addr3: Destination memory address (hexadecimal)

Example:

```
> M 1000,10FF,2000
```

P[/B/W/L] ***addr=data*** V800

Modifies the contents of an I/O port. If ***=data*** is omitted, displays the contents of the I/O port.

/B: 8 bits **/W:** 16 bits **/L:** 32 bits

addr: I/O port address (hexadecimal)

data: I/O port-modifying data (hexadecimal)

Example:

```
> P/W 1000=55
> P/L 2000
```

Q Closes the MDXDEB.

R Displays the contents of a register.

R/F MIPS PowerPC

Displays the contents of the FPU register.

R ***reg=data*** Modifies the contents of a register.

reg: Register name (see Appendix G, "Register Names").

data: Register modification data (hexadecimal)

Example:

```
> R PC=2000
```

S [*num*] Executes a user program in steps. If *num* is omitted, executes only one step.

num: Number of steps to be executed (decimal)

Example:

```
> S
> S 10
```

V Displays the versions of the debugger and the monitor program.

CHAPTER 7 MDXCVT

This chapter describes how to operate the file conversion tool MDXCVT, which allows you to rapidly download S-record files and IEEE695 files.

In Windows, this program should be run in an MS-DOS compatible window.

Starting the MDXCVT

```
mdxcvt [-c config|-s|-i|-v|-o offset] infile [outfile]
```

infile	Input file name
outfile	Output file name (default: input file name with the extension *.mdx*)
-c config	Configuration file name
-s	Reads an S-record file.
-i	Reads an IEEE695 file (default).
-v	Displays information of the configuration file.
-o offset	Adds an offset address to the output file.

The MDXCVT converts either an S-record file or an IEEE695 file, and generates an MDX binary file. Subsequently, the MDX binary file can be downloaded rapidly using the L command of the MDXDEB.

NOTE: Objects that are located outside the ROM area cannot be converted. If a non-ROM area object is found in the input file, the following warning message will be displayed:

```
WARNING: Not in ROM range: xxxx
```

NOTE: Objects in the ROM area that overlap with the **WORKROM area cannot be converted. If an object that overlaps with the WORKROM area** is found in the input file, the following warning message will be displayed:

```
ERROR: In WORKROM range: xxxx
```

NOTE: If any of the following fields in the configuration file, which are referenced by the MDXCVT program, are modified, the MDXCVT program should be rerun.

```
ROM
ROMSIZE
WORKROM
WORKROMSIZE
```

APPENDIX A SPECIFICATIONS

Equipment dimensions	75mm(H)x235mm(W)x175mm(D)
Weight	2.3Kg
Power supply	AC100V 50Hz/60Hz
Power consumption	20W Parallel 30W Ethernet
ROM cable	300mm
External trigger cable	300mm
Operating temp. range	0℃～35℃
Storage temp. range	-10℃～55℃
Ambient humidity range	30%～85%
Supported ROMs	See Appendix D, "Supported ROMs".
Supported number	One, two, or four 8-bit ROM chips
of ROM chips	One or two 16-bit ROM chips
Emulation memory size	512K bytes (or 2MB)[*1]
Access time	75n sec. from CS
Interface	Dedicated parallel Parallel
	Ethernet 10BASE-T Ethernet
Supported CPUs	See Appendix F, "Supported CPUs".
Downloading speed	256K bytes/second Parallel
	1.2M bytes/minute Ethernet
Constraints	See Appendix B, "Target System Constraints".

Supported debuggers

- MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 (68K/ARM/MIPS/PowerPC/SH/V800)
- MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1 (68K/ARM/MIPS/PowerPC/SH/V800)
- MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris (68K/ARM/MIPS/PowerPC/SH/V800)
- SingleStep 6.5 68K on Windows 3.1
- SingleStep 6.5 PowerPC on Windows 3.1
- XRAY68K 2.2a on MS-DOS PC/AT (product name: XHI68KMD)
- XRAY68K 2.2a on MS-DOS PC-98 (product name:XHI68KMD)
- XRAY68K 3.4 on SunOS/Solaris (product name:XHI68KMD)

[*1] With a ROM read data bus width of 8 bits, only half the emulation memory size is available for use.

- MDXDEB 3.5 on MS-DOS/Windows 3.1/95 (68K/ARM/MIPS/PowerPC/SH/V800)
- MDXDEB 3.5 on SunOS/Solaris (68K/ARM/MIPS/PowerPC/SH/V800)

APPENDIX B TARGET SYSTEM CONSTRAINTS

1. ROM and RAM are operating smoothly.

2. ROM has free space of 16K bytes, and RAM, 4K bytes.

3. The RESET and NMI signals should be interconnected.

4. A ROM socket is installed.

5. ROM is not banked.

6. ROM is not cached. ROM is not page-accessed.

7. ROM is byte-accessible (access can be restricted to specified addresses only).

8. When a ROM address signal changes, either the CS or the OE is inactivated (see the following page).

9. The ROM CS or OE contains a signal that represents the decoding of a higher address (see the following page).

10. In situations where multiple ROM chips are installed, all ROM address signals are identical.

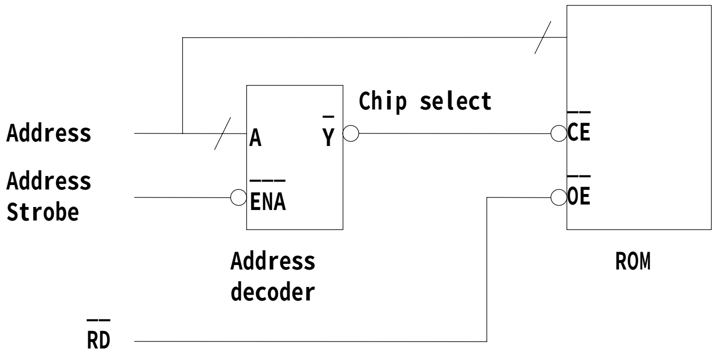
11. Programs can be executed on the RAM.

12. 68000 Neither ROM nor RAM is partitioned by function code signals (FC0-FC2).

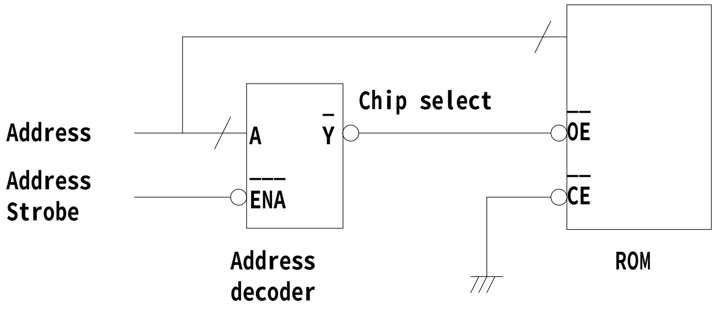
13. PowerPC The CPU runs only in a big endian.

14. The system endian should not be changed.

Target system in which the MDX700 runs

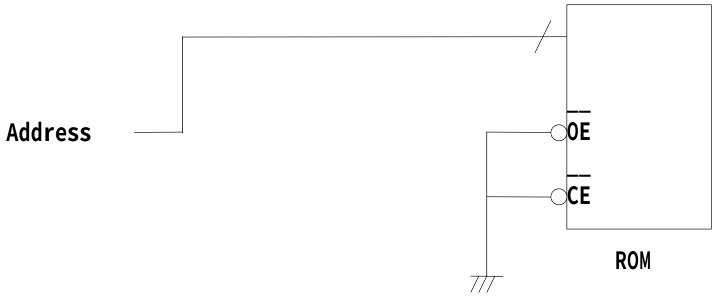


Target system in which the MDX700 runs conditionally



- In the case of a ROM read data bus width of 16 bits, only two 8-bit ROM chips or one 16-bit ROM chip can be used.
- In the case of a ROM read data bus width of 8 bits, only one ROM chip can be used.

Target system in which the MDX700 does not run



APPENDIX C NOTES

1. Downloading a user program into the memory area specified with **WORKROM** and **WORKRAM** in the configuration file disables the debugger.

2. If the NMI signal is not connected to the target system, do not break and abort a user program by operating the debugger. [*1]

3. 68000 SH Initialize the VBR register in the configuration file. Changing VBR register values when a user program is being executed disables the debugger.

4. PowerPC (403 only) Initialize the EVPR register in the configuration file. Changing EVPR register values when a user program is being executed disables the debugger.

5. MIPS User programs should be run in a memory space that does not use the TLB in the kernel mode (0x80000000-0xBFFFFFFF).

6. MIPS The R27 register, which is reserved by the monitor program, should not be used by user programs.

7. MIPS The BEV bit (normally set to 1) should not be changed during the execution of a user program.

8. V800 except V830 The following instructions cannot be executed stepwise:

   HALT
   LDSR
   RETI
   TRAP

9. ARM MIPS PowerPC SH-3 V800 In an exception handler, breakpoints can be set only on the instructions after any of the following registers are saved in a stack:

[*1] *Halt* and other commands in the case of the MULTI debugger

R14/SPSR	ARM
EPC/ErrorEPC	MIPS
SRR0/SRR1	PowerPC
SPC/SSR	SH-3
EIPC/EIPSW/FEPC/FEPSW	V800

10. MIPS In an exception handler, breakpoints can be set only on the instructions for which the status register EXL has been cleared (multiple interrupts are enabled).

11. SH-3 Breakpoints cannot be set on the TRAPA instruction. Program execution can be halted at the breakpoint, but program execution cannot be resumed from the breakpoint. To continue the execution of a program, the any breakpoint that is set on the TRAPA instruction should be released.

APPENDIX D SUPPORTED ROMs

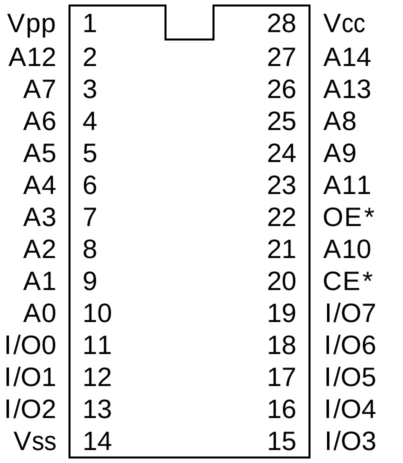
ROM probe	ROM size	ROM code	Manufacturer
B106 27256	32K x 8bit (0x8000byte)	HN27C256AG HN27C256HG μPD27C256AD TC57256AD TC57H256D M5L27256K M5M27C256K MBM27C256A-nnCZ 27256 27C256	Hitachi Hitachi NEC Toshiba Toshiba Mitsubishi Mitsubishi Fujitsu intel intel
B107 27512	64K x 8bit (0x10000byte)	HN27512G HN27C512AG μPD27C512D TM27512AD TC57512AD M5L5L27512K MBM27C512-nnCZ 27512	Hitachi Hitachi NEC Toshiba Toshiba Mitsubishi Fujitsu intel
B108 27010	128K x 8bit (0x20000byte)	HN27C101AG μPD27C1001AD TC571000D TC571000AD TC57H1000AD M5M27C101K MBM27C1001-nnZ 27010 27C010	Hitachi NEC Toshiba Toshiba Toshiba Mitsubishi Fujitsu intel intel

ROM probe	ROM size	ROM code	Manufacturer
B109 271000	128K x 8bit (0x20000byte)	HN27C301AG µPD27C1000AD TC571001D TC571001AD TC57H1001AD M5M27C100K MBM27C1000-nnZ	Hitachi NEC Toshiba Toshiba Toshiba Mitsubishi Fujitsu
B110 271024 DIP	64K x 16bit (0x20000byte)	HN27C1024HG µPD27C1024D µPD27C1024AD TC57H1024D TC57H1024AD MBM27C1024-nnZ 27210 27C210	Hitachi NEC NEC Toshiba Toshiba Fujitsu Intel Intel
B111 27020	256K x 8bit (0x40000byte)	µPD27C2001D M5M27C201K	NEC Mitsubishi
B112 27040	512K x 8bit (0x80000byte)	HN27C4001G µPD27C4001DZ TC574000D TC574000DI M5M27C401K MBM27C4001-nnZ 27040	Hitachi NEC Toshiba Toshiba Mitsubishi Fujitsu Intel

ROM probe	ROM size	ROM code	Manufacturer
B113 274096 DIP	256K x 16bit	HN27C4096G	Hitachi
	(0x80000byte)	HN27C4096HG	Hitachi
		HN27C4096AG	Hitachi
		HN27C4096AHG	Hitachi
		TC574096D	Toshiba
		MBM27C4096-nnZ	Fujitsu
		27C240	Intel
PL44 271024 PLCC	64K x 16bit	HN27C1024HCC	Hitachi
	(0x20000byte)	MBM27C1024-nnTV	Fujitsu
PL44 274096 PLCC	256K x 16bit	HN27C4096CC	Hitachi
	(0x80000byte)	HN27C4096HCC	Hitachi
		HN27C4096ACC	Hitachi
See PL44 Jumper Table.		HN27C4096AHCC	Hitachi
B117 27C4000 8bit	512K x 8bit (0x80000byte)	HN27C4000G	Hitachi
B118 27C4000 16bit	256K x 16bit (0x80000byte)	HN27C4000G	Hitachi
B119 27C8000 8bit	1024K x 8bit (0x100000byte)	μPD27C8000	NEC
B120 27C8000 16bit	512K x 16bit (0x100000byte)	μPD27C8000	NEC
29F040 DIP	512K x 8bit (0x80000byte)	Am29F040	AMD
B117 27C4000 8bit with DSOP44RB	512K x 8bit (0x80000byte)	PA28F400	Intel
B118 27C4000 16bit with DSOP44RB	256K x 16bit (0x80000byte)	PA28F400	Intel

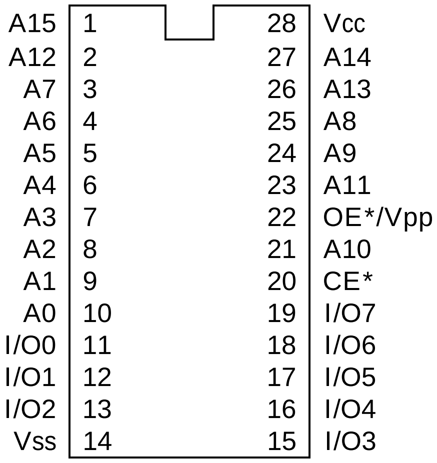
PL44 jumper	JP1	JP2	JP3	JP4	JP5	JP6
27C1024	1-2ON	2-3ON	2-3ON	2-3ON	2-3ON	2-3ON
27C4096	1-2ON	1-2ON	2-3ON	1-2ON	1-2ON	2-3ON

APPENDIX E ROM PIN ASSIGNMENTS



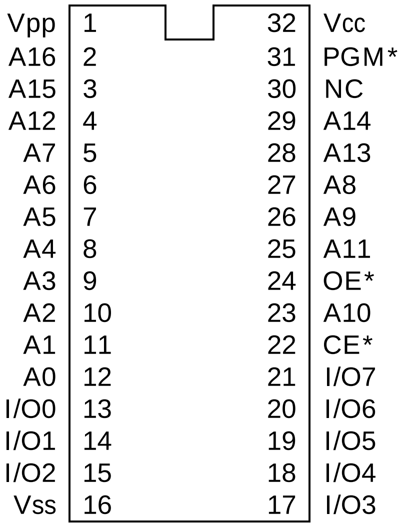
Vpp	1	28	Vcc
A12	2	27	A14
A7	3	26	A13
A6	4	25	A8
A5	5	24	A9
A4	6	23	A11
A3	7	22	OE*
A2	8	21	A10
A1	9	20	CE*
A0	10	19	I/O7
I/O0	11	18	I/O6
I/O1	12	17	I/O5
I/O2	13	16	I/O4
Vss	14	15	I/O3

B106 27256



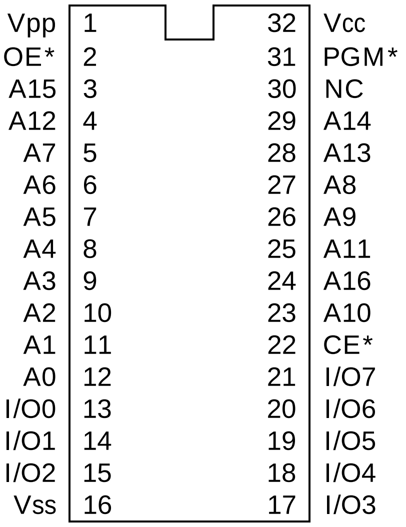
A15	1	28	Vcc
A12	2	27	A14
A7	3	26	A13
A6	4	25	A8
A5	5	24	A9
A4	6	23	A11
A3	7	22	OE*/Vpp
A2	8	21	A10
A1	9	20	CE*
A0	10	19	I/O7
I/O0	11	18	I/O6
I/O1	12	17	I/O5
I/O2	13	16	I/O4
Vss	14	15	I/O3

B107 27512



Vpp	1	32	Vcc
A16	2	31	PGM*
A15	3	30	NC
A12	4	29	A14
A7	5	28	A13
A6	6	27	A8
A5	7	26	A9
A4	8	25	A11
A3	9	24	OE*
A2	10	23	A10
A1	11	22	CE*
A0	12	21	I/O7
I/O0	13	20	I/O6
I/O1	14	19	I/O5
I/O2	15	18	I/O4
Vss	16	17	I/O3

B108 27010



Vpp	1	32	Vcc
OE*	2	31	PGM*
A15	3	30	NC
A12	4	29	A14
A7	5	28	A13
A6	6	27	A8
A5	7	26	A9
A4	8	25	A11
A3	9	24	A16
A2	10	23	A10
A1	11	22	CE*
A0	12	21	I/O7
I/O0	13	20	I/O6
I/O1	14	19	I/O5
I/O2	15	18	I/O4
Vss	16	17	I/O3

B109 271000

APPENDIX E. ROM PIN ASSIGNMENTS

Vpp	1	32	Vcc
A16	2	31	PGM*
A15	3	30	A17
A12	4	29	A14
A7	5	28	A13
A6	6	27	A8
A5	7	26	A9
A4	8	25	A11
A3	9	24	OE*
A2	10	23	A10
A1	11	22	CE*
A0	12	21	I/O7
I/O0	13	20	I/O6
I/O1	14	19	I/O5
I/O2	15	18	I/O4
Vss	16	17	I/O3

B111 27020

Vpp	1	32	Vcc
A16	2	31	A18
A15	3	30	A17
A12	4	29	A14
A7	5	28	A13
A6	6	27	A8
A5	7	26	A9
A4	8	25	A11
A3	9	24	OE*
A2	10	23	A10
A1	11	22	CE*
A0	12	21	I/O7
I/O0	13	20	I/O6
I/O1	14	19	I/O5
I/O2	15	18	I/O4
Vss	16	17	I/O3

B112 27040

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

B110 271024

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

B113 274096

APPENDIX E. ROM PIN ASSIGNMENTS

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

B117/B118
27C4000

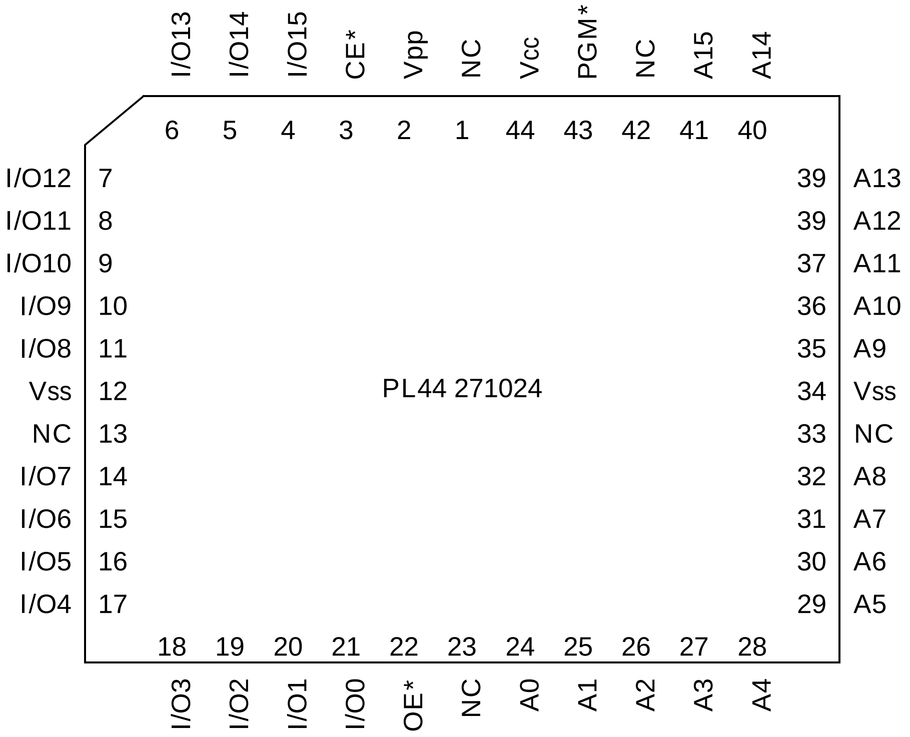
A18	1	42	NC
A17	2	41	A8
A7	3	40	A9
A6	4	39	A10
A5	5	38	A11
A4	6	37	A12
A3	7	36	A13
A2	8	35	A14
A1	9	34	A15
A0	10	33	A16
CE*	11	32	BYTE*/Vpp
Vss	12	31	Vss
OE*	13	30	I/O15/A-1
I/O0	14	29	I/O7
I/O8	15	28	I/O14
I/O1	16	27	I/O6
I/O9	17	26	I/O13
I/O2	18	25	I/O5
I/O10	19	24	I/O12
I/O3	20	23	I/O4
I/O11	21	22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

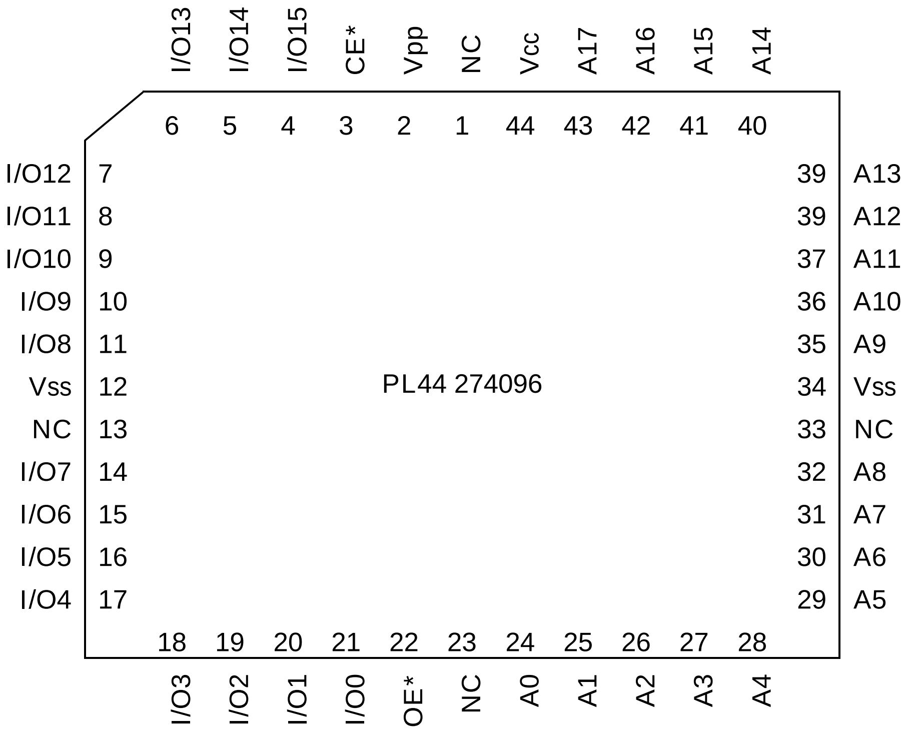
B119/B120
27C8000

A18	1	32	Vcc
A16	2	31	WE*
A15	3	30	A17
A12	4	29	A14
A7	5	28	A13
A6	6	27	A8
A5	7	26	A9
A4	8	25	A11
A3	9	24	OE*
A2	10	23	A10
A1	11	22	CE*
A0	12	21	I/O7
I/O0	13	20	I/O6
I/O1	14	19	I/O5
I/O2	15	18	I/O4
Vss	16	17	I/O3

29F040

APPENDIX E. ROM PIN ASSIGNMENTS





APPENDIX F SUPPORTED CPUs

- The entries in the table below are case-sensitive, where _LE denotes a little endian, _64 a 64-bit register, and _LE64 a little endian with a 64-bit register.
- If your CPU is not on the list, choose the CPU which has the same core CPU instead.

68000	MIPS	MIPS	PowerPC	SH	V800
68000	R3051	R3051_LE	PPC403	SH7032	V805
68010	R3052	R3052_LE	PPC603	SH7034	V810
68020	R3081	R3081_LE	PPC603e	SH7020	V820
68EC020	R3600	R3600_LE	PPC604	SH7021	V821
68030	R3800	R3800_LE	PPC604e	SH7604	V830
68EC030	R3900	R3900_LE	PPC821	SH7702	V831
68040	R4000	R4000_LE	PPC860	SH7707	V851
68EC040	R4100	R4100_LE		SH7708	V852
68LC040	R4200	R4200_LE		SH7707_LE	V853
CPU32	R4300	R4300_LE		SH7708_LE	V850E
CPU32+	R4400	R4400_LE			
68008	R4600	R4600_LE	ARM		
68EC000	R4650	R4650_LE	ARM7		
68HC000	R4700	R4700_LE	ARM7_LE		
68HC001	R5000	R5000_LE	ARM7T		
68301	R4000_64	R4000_LE64	ARM7T_LE		
68302	R4100_64	R4100_LE64	(T: Thumb)		
68303	R4200_64	R4200_LE64			
68305	R4300_64	R4300_LE64			
68306	R4400_64	R4400_LE64			
68330	R4600_64	R4600_LE64			
68331	R4650_64	R4650_LE64			
68332	R4700_64	R4700_LE64			
68F333	R5000_64	R5000_LE64			
68340					
68341					
68349					
68360					

APPENDIX G REGISTER NAMES

- **_HI denotes the high** 32 bits of a 64-bit register.
- **_LO denotes the low** 32 bits of a 64-bit register.
- **CPn denotes the n-th register of co-processor 0 registers.** MIPS

68000

D0 D1 D2 D3 D4 D5 D6 D7 A0 A1 A2 A3 A4 A5 A6

PC SR USP SSP ISP MSP VBR SFC DFC

CACR CAAR CRP_HI CRP_LO SRP_HI SRP_LO URP SRP TC

TT0 TT1 AC0 AC1 DTT0 DTT1 ITT0 ITT1 DACR0 DACR1 IACR0 IACR1 MMUSR ACUSR

FP0 FP1 FP2 FP3 FP4 FP5 FP6 FP7 FPCR FPSR FPIAR

SH

R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

SR GBR VBR MACH MACL PR PC SSR SPC

R0_BANK R1_BANK R2_BANK R3_BANK R4_BANK R5_BANK R6_BANK R7_BANK

V800

R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

R16 R17 R18 R19 R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31

PC EIPC EIPSW FEPC FEPSW ECR PSW PIR TKCW CHCW ADTRE

DPC DPSW HCCW ADTRE0 ADTRE1 ADTRD0 ADTRD1 ADTRD2 ADTRD3 DCW

DTPC DTPSW DBPC DBPSW CTBP DIR

MIPS

R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

R16 R17 R18 R19 R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31

PC HI LO

F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14 F15 F16

F17 F18 F19 F20 F21 F22 F23 F24 F25 F26 F27 F28 F29 F30 F31 FCR0 FCR31

CP0 CP1 CP2 CP3 CP4 CP5 CP6 CP7 CP8 CP9 CP10 CP11 CP12 CP13 CP14 CP15 CP16

CP17 CP18 CP19 CP20 CP21 CP22 CP23 CP24 CP25 CP26 CP27 CP28 CP29 CP30 CP31

MIPS 64bit

R0_HI R0_LO R1_HI R1_LO R2_HI R2_LO R3_HI R3_LO R4_HI R4_LO R5_HI R5_LO

R6_HI R6_LO R7_HI R7_LO R8_HI R8_LO R9_HI R9_LO R10_HI R10_LO R11_HI R11_LO

R12_HI R12_LO R13_HI R13_LO R14_HI R14_LO R15_HI R15_LO R16_HI R16_LO

R17_HI R17_LO R18_HI R18_LO R19_HI R19_LO R20_HI R20_LO R21_HI R21_LO

R22_HI R22_LO R23_HI R23_LO R24_HI R24_LO R25_HI R25_LO R26_HI R26_LO

R27_HI R27_LO R28_HI R28_LO R29_HI R29_LO R30_HI R30_LO R31_HI R31_LO

PC_HI PC_LO HI_HI HI_LO LO_HI LO_LO

F0_HI F0_LO F1_HI F1_LO F2_HI F2_LO F3_HI F3_LO F4_HI F4_LO F5_HI F5_LO

F6_HI F6_LO F7_HI F7_LO F8_HI F8_LO F9_HI F9_LO F10_HI F10_LO F11_HI F11_LO

F12_HI F12_LO F13_HI F13_LO F14_HI F14_LO F15_HI F15_LO F16_HI F16_LO

F17_HI F17_LO F18_HI F18_LO F19_HI F19_LO F20_HI F20_LO F21_HI F21_LO

F22_HI F22_LO F23_HI F23_LO F24_HI F24_LO F25_HI F25_LO F26_HI F26_LO

F27_HI F27_LO F28_HI F28_LO F29_HI F29_LO F30_HI F30_LO F31_HI F31_LO

FCR0_HI FCR0_LO FCR31_HI FCR31_LO

CP0_HI CP0_LO CP1_HI CP1_LO CP2_HI CP2_LO CP3_HI CP3_LO CP4_HI CP4_LO

CP5_HI CP5_LO CP6_HI CP6_LO CP7_HI CP7_LO CP8_HI CP8_LO CP9_HI CP9_LO

CP10_HI CP10_LO CP11_HI CP11_LO CP12_HI CP12_LO CP13_HI CP13_LO CP14_HI CP14_LO

CP15_HI CP15_LO CP16_HI CP16_LO CP17_HI CP17_LO CP18_HI CP18_LO CP19_HI CP19_LO

CP20_HI CP20_LO CP21_HI CP21_LO CP22_HI CP22_LO CP23_HI CP23_LO CP24_HI CP24_LO

CP25_HI CP25_LO CP26_HI CP26_LO CP27_HI CP27_LO CP28_HI CP28_LO CP29_HI CP29_LO

CP30_HI CP30_LO CP31_HI CP31_LO

ARM

R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

R8_FIQ R9_FIQ R10_FIQ R11_FIQ R12_FIQ R13_FIQ R14_FIQ SPSR_FIQ

R13_IRQ R14_IRQ SPSR_IRQ

R13_SVC R14_SVC SPSR_SVC

R13_ABT R14_ABT SPSR_ABT

R13_UND R14_UND SPSR_UND

PC CPSR

PowerPC

GPR0 GPR1 GPR2 GPR3 GPR4 GPR5 GPR6 GPR7 GPR8 GPR9 GPR10 GPR11 GPR12 GPR13
GPR14 GPR15 GPR16 GPR17 GPR18 GPR19 GPR20 GPR21 GPR22 GPR23 GPR24 GPR25
GPR26 GPR27 GPR28 GPR29 GPR30 GPR31
FPR0_HI FPR0_LO FPR1_HI FPR1_LO FPR2_HI FPR2_LO FPR3_HI FPR3_LO
FPR4_HI FPR4_LO FPR5_HI FPR5_LO FPR6_HI FPR6_LO FPR7_HI FPR7_LO
FPR8_HI FPR8_LO FPR9_HI FPR9_LO FPR10_HI FPR10_LO FPR11_HI FPR11_LO
FPR12_HI FPR12_LO FPR13_HI FPR13_LO FPR14_HI FPR14_LO FPR15_HI FPR15_LO
FPR16_HI FPR16_LO FPR17_HI FPR17_LO FPR18_HI FPR18_LO FPR19_HI FPR19_LO
FPR20_HI FPR20_LO FPR21_HI FPR21_LO FPR22_HI FPR22_LO FPR23_HI FPR23_LO
FPR24_HI FPR24_LO FPR25_HI FPR25_LO FPR26_HI FPR26_LO FPR27_HI FPR27_LO
FPR28_HI FPR28_LO FPR29_HI FPR29_LO FPR30_HI FPR30_LO FPR31_HI FPR31_LO
PC CR FPSCR XER LR CTR TBL TBU MSR HID0 PVR HID1
IBAT0U IBAT0L IBAT1U IBAT1L IBAT2U IBAT2L IBAT3U IBAT3L
DBAT0U DBAT0L DBAT1U DBAT1L DBAT2U DBAT2L DBAT3U DBAT3L
SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8 SR9 SR10 SR11 SR12 SR13 SR14 SR15
SDR1 DMISS DCMP HASH1 HASH2 IMISS ICMP RPA PMC1 PMC2 MMCR0 SDA SIA DAR
SPRG0 SPRG1 SPRG2 SPRG3 DSISR SRR0 SRR1 DEC EAR IABR DABR PIR

EIE EID NRI CMPA CMPB CMPC CMPD ICR DER COUNTA COUNTB CMPE CMPF CMPG CMPH
LCTRL1 LCTRL2 ICTRL BAR DPDR DPIR IMMR
IC_CST IC_ADR IC_DAT DC_CST DC_ADR DC_DAT
MI_CTR MI_AP MI_EPN MI_TWC MI_RPN MI_DBCAM MI_DBRAM0 MI_DBRAM1 MD_CTR
M_CASID MD_AP MD_EPN M_TWB MD_TWC MD_RPN M_TW MD_DBCAM MD_DBRAM0 MD_DBRAM1

BEAR BESR BR0 BR1 BR2 BR3 BR4 BR5 BR6 BR7 DMACC0 DMACC1 DMACC2 DMACC3
DMACR0 DMACR1 DMACR2 DMACR3 DMACT0 DMACT1 DMACT2 DMACT3
DMADA0 DMADA1 DMADA2 DMADA3 DMASA0 DMASA1 DMASA2 DMASA3
DMASR EXIER EXISR IOCR CDBCR DAC1 DAC2 DBCR DBSR DCCR DEAR ESR EVPR
IAC1 IAC2 ICCR ICDBDR PBL1 PBL2 PBU1 PBU2 PIT SRR2 SRR3
TBHI TBLO TCR TSR

APPENDIX H OPERATING PRINCIPLES

Internal operations that occur when the debugger is started

1. The configuration file is read.
2. The monitor program and the contents of the configuration file are downloaded to the address specified by **WORKROM** in the configuration file. [*1]
3. The reset vector that resets the monitor program is downloaded to the address specified by the **RESETVECTOR** in the configuration file.
4. The RESET signal is output. If the RESET signal is connected to the target system, the monitor program is reset and started, and the debugger begins communicating with the monitor program.
5. An exception-handling address is fetched from the monitor program. [*2]
6. An exception vector is generated from the exception-handling address.
7. The exception vector is downloaded onto the target system.

The above internal operations permit the monitor program to be moved simply through a modification of the **WORKROM** in the configuration file. Additionally, this technique avoids the need to code an exception vector in the source file for the monitor program.

Functions of the monitor program

The monitor program receives functional requests from the debugger, executes the requests, returns the results to the debugger, and repeats these actions. The monitor program executes the following functional requests:

- Reading from RAM
- Writing to RAM
- Executing a user program
- Stepwise execution of a user program 68000
- Calculating the next PC ARM MIPS PowerPC SH V800

[*1] Enables the monitor program to reference the contents of the configuration file.

[*2] Because the address to which the monitor program is downloaded varies with the value of the **WORKROM** parameter, the debugger dynamically acquires the address of the exception vector.

The monitor program transmits these functional requests to the monitor program as needed.

Breakpoint function

The breakpoint function is implemented by using the CPU's illegal instruction exception. The instruction at the address at which a breakpoint is set is replaced by an illegal instruction before the user program is executed; it is automatically restored to the original instruction after the user program has been breaked.

The following illegal instructions are used in breakpoints:

● **0x4AFC**	68000	(ILLEGAL instruction)
● **0xE7FFE7FF**	ARM	
● **0xDEDE**	THUMB	
● **0x0000000D**	MIPS	(BREAK instruction)
● **0x00000000**	PowerPC	
● **0xF00F**	SH	
● **0x6C6C**	V830	(BRKPNT instruction)
● **0xFFFEFFFF**	V850	
● **0xF840**	V850E	
● **0x5858**	V800 except V830,V850 or V850E	

Stepwise execution function

In the case of a 68000 , the stepwise execution function is implemented through the use of a CPU trace exception.

In the case of a ARM MIPS PowerPC SH V800 CPU, the stepwise execution function is implemented through the use of an illegal instruction as with breakpoints. Specifically, the value of the PC is calculated after an instruction is executed, a breakpoint (illegal instruction) is set at the calculated position, and then the user program is executed.

Abort/break function

The abort/break function is implemented through the use of the following exceptions:

- Level 7 interrupt 68000
- An exception specified by the **ABORTVECTOR** in the configuration file ARM PowerPC
- NMI exception MIPS SH V800

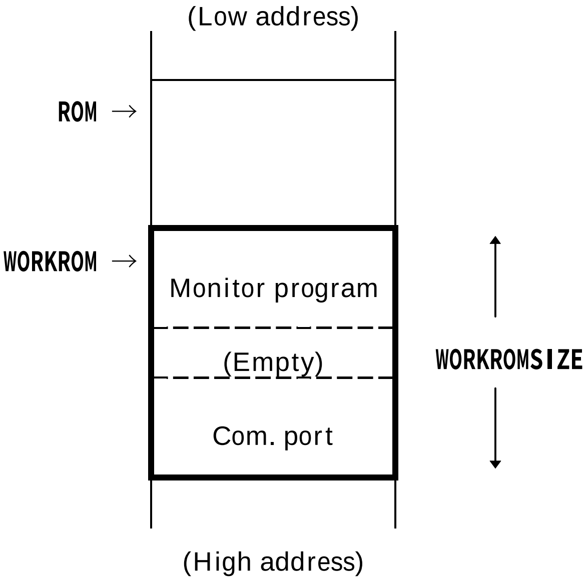
If the debugger performs an operation to halt the user program that is being executed [*3], the debugger outputs an NMI signal from the external trigger cable. This causes the user program to halt. [*4]

[*3] *Halt* commands in the case of the MULTI debugger for example.

[*4] If the NMI signal is not connected to the target system, the debugger cannot apply an abort/break.

APPENDIX I COMMUNICATION PORT AREA

The monitor program is downloaded onto the area specified by the **WORKROM** and **WORKROMSIZE** parameters in the configuration file, from the beginning of the area. The end portion of this area is used as a communication port that supports communications between the monitor program and the host system.



The size of a communication port can vary according to the way the **BUS** parameter is specified in the configuration file:

BUS 8	512 bytes
BUS 16	1024 bytes
BUS 32	2048 bytes
BUS 64	4096 bytes

Therefore, the start address of a communication port can be calculated as follows:

Com. port start address = (WORKROM + WORKROMSIZE) - 512 * (BUS / 8)

NOTE: The starting address of a communication port must be boundaryu-aligned with the size of the communication port.

NOTE: When modifying the **WORKROM** or **WORKROMSIZE** parameter, make sure that the monitor program area does not overlap with the communication port area.

NOTE: When modifying the **WORKROM** or **WORKROMSIZE** parameter, make sure that the starting address of the communication port is boundary-aligned.

APPENDIX J MONITOR PROGRAM EXCEPTION VECTORS

- The uppercase characters enclosed in parentheses denote values that are specified in the configuration file.
- The uppercase characters not enclosed in parentheses denote labels that are included in the monitor program source file.
- The labels delimited by slashes are switched by the CPU.
- The entry **(nop)** means that the *nop* continues on the next instruction.

68000

```
(RESETVECTOR)       dc.l     (WORKRAM) + (WORKRAMSIZE) / 2
(RESETVECTOR)+4     dc.l     (WORKROM)
(REG_VBR)+0x08      dc.l     BUS_ERROR
(REG_VBR)+0x0C      dc.l     ADDR_ERROR
(REG_VBR)+0x10      dc.l     ENTRY_68000/ENTRY_68010/ENTRY_68020/ENTRY_68030/
                             ENTRY_68EC030/ENTRY_68040/ENTRY_68EC040/ENTRY_68LC040
(REG_VBR)+0x24      ditto
(REG_VBR)+0x7C      ditto
```

MIPS R3xxx

```
(RESETVECTOR)       j       (WORKROM)       During resetting only
(RESETVECTOR)       j       ENTRY_R3000
(RESETVECTOR)+4     nop
0xBFC00100          j       ENTRY_R3000
                   (nop)
0xBFC00170          j       ENTRY_R3000
                   (nop)
0xBFC00180          j       ENTRY_R3000
                   (nop)
0xBFC001F0          j       ENTRY_R3000
                   (nop)
0xBFC00200          j       ENTRY_R3000
                   (nop)
0xBFC00270          j       ENTRY_R3000
```

```
0xBFC00274          nop
```

MIPS R4xxx, R5xxx

```
(RESETVECTOR)       j       (WORKROM)       During resetting only
(RESETVECTOR)       j       ENTRY_32BIT/ENTRY_64BIT
(RESETVECTOR)+4     nop
0xBFC00200          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC00270          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC00280          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC002F0          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC00300          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC00370          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC00380          j       ENTRY_32BIT/ENTRY_64BIT
                   (nop)
0xBFC003F0          j       ENTRY_32BIT/ENTRY_64BIT
0xBFC003F4          nop
```

ARM

```
(RESETVECTOR)       ldr     pc, .+0x20
(ABORTVECTOR)       ldr     pc, .+0x20
(RESETVECTOR+0x20)  .data.l ENTRY_ARM7
(ABORTVECTOR+0x20)  .data.l ENTRY_ARM7
```

PowerPC (not 403) MSR[IP]=0

```
(RESETVECTOR)       b       (WORKROM)
(ABORTVECTOR)       b       ENTRY_PPC
0x00000700          b       ENTRY_PPC
```

PowerPC (not 403) MSR[IP]=1

```
(RESETVECTOR)       b       (WORKROM)
(ABORTVECTOR)       b       ENTRY_PPC
0xFFF00700          b       ENTRY_PPC
```

PowerPC (403)

```
(RESETVECTOR)       b       (WORKROM)
(ABORTVECTOR)       b       ENTRY_PPC
(REG_EVPR)+0x700    b       ENTRY_PPC
```

SH-1 SH-2

```
(RESETVECTOR)       .data.l (WORKROM)
(RESETVECTOR)+0x04  .data.l (WORKRAM) + (WORKRAMSIZE) / 2
(RESETVECTOR)+0x08  .data.l (WORKROM)
(RESETVECTOR)+0x0C  .data.l (WORKRAM) + (WORKRAMSIZE) / 2
(REG_VBR)+0x10      .data.l ENTRY_SH
(REG_VBR)+0x2C      .data.l ENTRY_SH
```

SH-3

```
(RESETVECTOR)        mov.l   @(+8,pc),r0
(RESETVECTOR)+0x02   jmp     @r0
(RESETVECTOR)+0x04   nop
(RESETVECTOR)+0x06   nop
(RESETVECTOR)+0x08   data.l  (WORKROM)
(REG_VBR)+0x100      mov.l   r0,@-sp
(REG_VBR)+0x102      mov.l   @(+6,pc),r0
(REG_VBR)+0x104      jmp     @r0
(REG_VBR)+0x106      mov.l   @sp+,r0
(REG_VBR)+0x108      data.l  ENTRY_SH3
                     (nop)
(REG_VBR)+0x1F0      mov.l   r0,@-sp
(REG_VBR)+0x1F2      mov.l   @(+6,pc),r0
(REG_VBR)+0x1F4      jmp     @r0
(REG_VBR)+0x1F6      mov.l   @sp+,r0
(REG_VBR)+0x1F8      data.l  ENTRY_SH3
(REG_VBR)+0x1FA      nop
(REG_VBR)+0x1FC      nop
(REG_VBR)+0x1FE      nop
(REG_VBR)+0x600      mov.l   r0,@-sp
(REG_VBR)+0x602      mov.l   @(+6,pc),r0
(REG_VBR)+0x604      jmp     @r0
(REG_VBR)+0x606      mov.l   @sp+,r0
(REG_VBR)+0x608      data.l  ENTRY_SH3
                     (nop)
(REG_VBR)+0x6F0      mov.l   r0,@-sp
(REG_VBR)+0x6F2      mov.l   @(+6,pc),r0
(REG_VBR)+0x6F4      jmp     @r0
(REG_VBR)+0x6F6      mov.l   @sp+,r0
(REG_VBR)+0x6F8      data.l  ENTRY_SH3
(REG_VBR)+0x6FA      nop
(REG_VBR)+0x6FC      nop
(REG_VBR)+0x6FE      nop
```

V800 except V830, V850 or V850E

```
(RESETVECTOR)       movhi   hi(WORKROM), r0, r1
(RESETVECTOR)+0x04  ori     lo(WORKROM), r1, r1
(RESETVECTOR)+0x08  jmp     [r1]
 BASE = (RESETVECTOR) & 0xFFFFFF00
(BASE)+0x90         st.w    r1, -8[sp]
(BASE)+0x94         movhi   hi(ENTRY_V800), r0, r1
(BASE)+0x98         ori     lo(ENTRY_V800), r1, r1
(BASE)+0x9C         jmp     [r1]
(BASE)+0xD0         st.w    r1, -8[sp]
(BASE)+0xD4         movhi   hi(ENTRY_V800), r0, r1
(BASE)+0xD8         ori     lo(ENTRY_V800), r1, r1
(BASE)+0xDC         jmp     [r1]
```

V830

```
(RESETVECTOR)       movhi   hi(WORKROM), r0, r1
(RESETVECTOR)+0x04  ori     lo(WORKROM), r1, r1
(RESETVECTOR)+0x08  jmp     [r1]
 BASE = (RESETVECTOR) & 0xFFFFFF00
(BASE)+0xD0         st.w    r1, -8[sp]
(BASE)+0xD4         movhi   hi(ENTRY_V800), r0, r1
(BASE)+0xD8         ori     lo(ENTRY_V800), r1, r1
(BASE)+0xDC         jmp     [r1]
(BASE)+0xE0         st.w    r1, -8[sp]
(BASE)+0xE4         movhi   hi(ENTRY_V800), r0, r1
(BASE)+0xE8         ori     lo(ENTRY_V800), r1, r1
(BASE)+0xEC         jmp     [r1]
```

V850 V850E

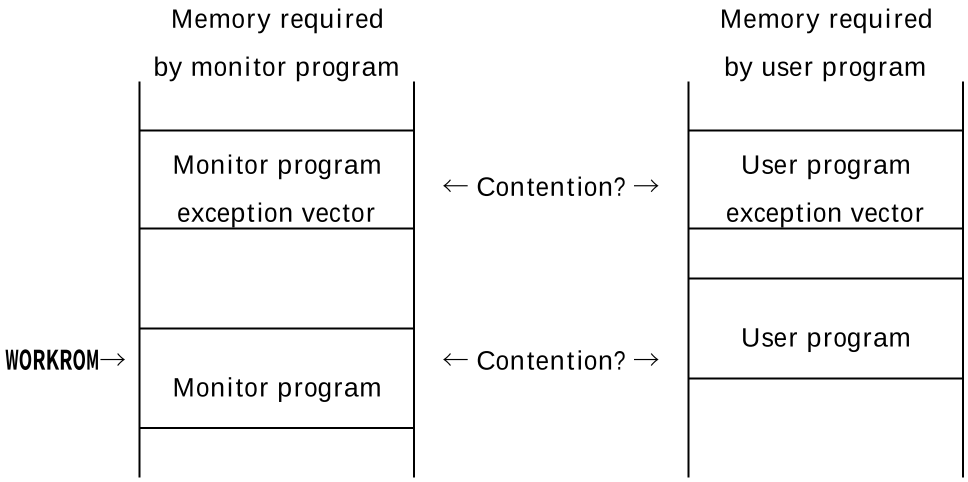
```
(RESETVECTOR)       movhi   hi(WORKROM), r0, r1
(RESETVECTOR)+0x04  ori     lo(WORKROM), r1, r1
(RESETVECTOR)+0x08  jmp     [r1]
0x00000010          st.w    r1, -8[sp]
0x00000014          movhi   hi(ENTRY_V800), r0, r1
0x00000018          ori     lo(ENTRY_V800), r1, r1
0x0000001C          jmp     [r1]
0x00000060          st.w    r1, -8[sp]
0x00000064          movhi   hi(ENTRY_V800), r0, r1
0x00000068          ori     lo(ENTRY_V800), r1, r1
0x0000006C          jmp     [r1]
```

APPENDIX K RUNNING USER PROGRAM WITH THE MONITOR PROGRAM

When creating a user program, the following conditions should be noted with care so that there will be no memory contention between that program and the monitor program. A debugger cannot be used when there is a memory contention.

- The user program does not use the area specified by parameters **WORKROM/WORKROMSIZE** and **WORKRAM/WORKRAMSIZE** in the configuration file.

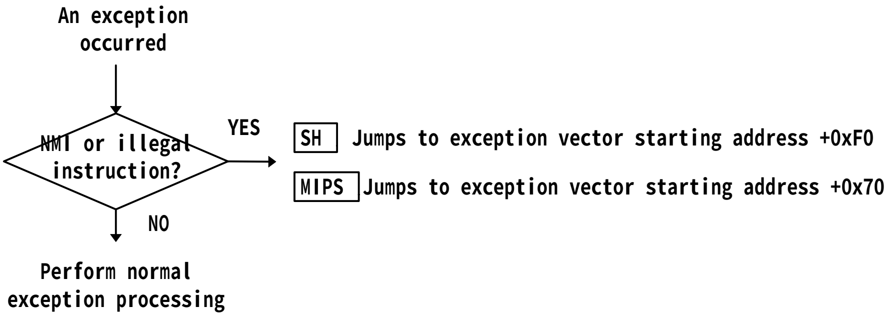
- The user program does not use the exception vector area required by the monitor program. See Appendix J, "Monitor Program Exception Vectors", for a description of exception vectors that are required by the monitor program.


Memory required by monitor program
Memory required by user program
Monitor program exception vector
← Contention? →
User program exception vector
User program
WORKROM→
← Contention? →
Monitor program

If there is a contention between user program and monitor program, the load address of the user program should be moved to a location that does not conflict with the monitor program.

In situations where the contention is only between exception vectors, the contention can be resolved by using debugger commands. Specifically, the exception vector for the monitor program can be reloaded by executing the I command after the user program has been downloaded.

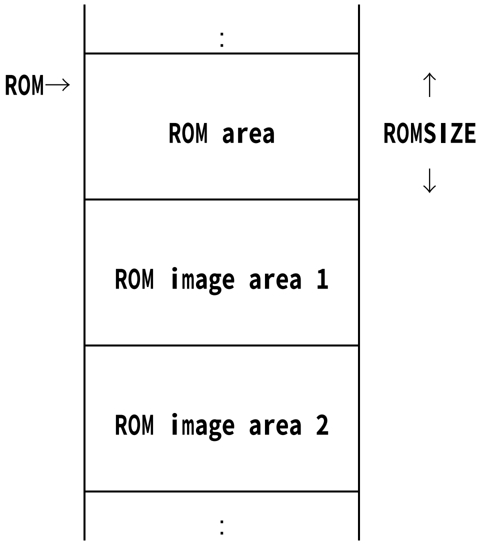
In the case of an MIPS SH-3, multiple exceptions use a single exception vector, thus resulting in a contention for the exception vector. When using these CPUs, the following procedure should be added at the beginning of the NMI and the exception vector that is used by an illegal instruction:


An exception occurred
NMI or illegal instruction?
YES
SH Jumps to exception vector starting address +0xF0
MIPS Jumps to exception vector starting address +0x70
NO
Perform normal exception processing

The above procedure causes control to jump to the monitor program only when an illegal instruction due to a breakpoint and an NMI due to an abort/break have occurred. To accommodate the addition of these procedures, the monitor program exception vector is composed of two branch instructions. See Appendix J, "Monitor Program Exception Vectors" for further details on this topic.

APPENDIX L USING THE ROM IMAGE AREA

Target systems in which addresses are not fully decoded contain a ROM image area that can be accessed in the same way as a ROM area.



In the normal environment setting, the ROM image area behaves the same way as RAM, i.e., data can be read from it, but cannot be written to it.

To make the ROM image area writable, as the ROM area, you need to add an item **ROM_IMAGE** to the configuration file. The start address of the ROM image area should be specified in the **ROM_IMAGE** entry as follows:

```
ROM             0x00000000      ; ROM start address
ROMSIZE         0x00100000      ; ROM size
ROM_IMAGE       0x00100000      ; ROM image start address
```

NOTE: **ROM_IMAGE** can be used only once in the configuration file.

NOTE: The ROM image area will have the same size as the **ROMSIZE.**

APPENDIX M USING THE CACHE ROM AREA

The procedures for using the cache ROM area are as follows:

1. In the case of SH MIPS V800, specify the cache ROM area as a ROM image area. See Appendix L, "Using a ROM Image Area", for further details.

2. Modify the monitor program by opening its source file and by adding a code that flushes all cache to the following label sections:

   USER_UPDATE
   USER_RUN

3. Assemble the modified source file. See the debugger release notes for assembly procedures.

USER_UPDATE refers to a subroutine that is called whenever memory contents have been modified. Likewise, **USER_RUN** refers to a subroutine that is called before a user program is executed. By flushing the cache in these subroutines, it is possible to cause the CPU to recognize the cache even in situations where the ROM area has been rewritten.

APPENDIX N ERROR MESSAGES

MDXERR: bad host name: xxxx Ethernet

The specified host name is invalid. Make sure that the host name is properly registered.

MDXERR: connection refused: Ethernet

The attempted connection was refused. Check to see that the IP address of the MDX700 is the same as the IP address of the specified host name.

MDXERR: bad communication port Parallel

The parallel interface board is inaccessbile. Make sure that the power for the MDX700 is on. Also make sure that the switch settings for the parallel interface board are in agreement with the PORT value in the configuration file.

MDXERR: file not found: xxxx

The specified file was not found. Make sure that the file and directory names are specified correctly.

MDXERR: bad configuration file: xxxx

The contents of the configuration file are invalid. Make sure that the configuration file was modified correctly. If a (boundary) error is displayed, set **WORKROM** and **WORKROMSIZE** parameters so that the start address of the communication port is flush against the requested boundary. See Appendix I, "Communication Port Area", for details.

MDXERR: bad monitor file:

The monitor program cannot be read correctly. Make sure that the program is in the S-record file format.

MDXERR: bad MDX binary file: xx xx xx ...

The specified file is not in the MDX binary file format. Make sure that the specified file is one that has been converted using the MDXCVT.

MDXERR: communication port timeout:

The debugger is unable to communicate with the monitor program. Make sure that the MDX700 is properly connected to the target system. See Appendix O, "Troubleshooting", for details.

MDXERR: communication port timeout: (but monitor was started)

The debugger is unable to communicate with the monitor program (the monitor program was launched successfully). Make sure that RAM exists at a **WORKRAM** address. If the monitor program has been modified, make sure that the revised code does not contain errors.

APPENDIX O TROUBLESHOOTING

The error message "**MDXERR: communication port timeout:"** indicates that the debugger is unable to communicate with the monitor program. When this message appears, check the following items:

- The power for the MDX700 and the target system is on.
- The MDX700 is properly connected to the target system, and the ROM probe is not inserted in reverse.
- The configuration file has the proper settings.

The following procedures can be employed to identify the cause of the error:

Step 1

1. If the RESET signal is connected, disconnect it.
2. If the monitor program has been modified, reset it to the factory default.

Step 2

3. Start the MDXDEB (ignore any error indications).
4. Using the E command, rewrite the memory in the ROM area.
5. Using the D command, verify the memory contents that have been rewritten.

A rewritable ROM area contains the following correct settings. If the ROM area fails to be rewritten, these settings should be re-checked.

- Connecting the MDX700 to the host
- Setting the switches for the parallel interface board Parallel
- The **PORT** in the configuration file Parallel

Step 3

6. Manually reset the target system.
7. Use the V command to display version information.

If the V command can be executed flawlessly, the following items are correctly set. If an error occurs, these items should be re-checked:

- Connecting the MDX700 to the target system
- **BUS** in the configuration file
- **ROM** in the configuration file
- **ROMSIZE** in the configuration file
- **WORKROM** in the configuration file
- **WORKROMSIZE** in the configuration file
- **RESETVECTOR** in the configuration file

Step 4

8. Use the I command to re-initialize the system.

If the I command can be executed flawlessly, the following items are correctly set. If an error occurs, these items should be re-checked:

- **WORKRAM** in the configuration file
- **WORKRAMSIZE** in the configuration file
- **TIMER** in the configuration file

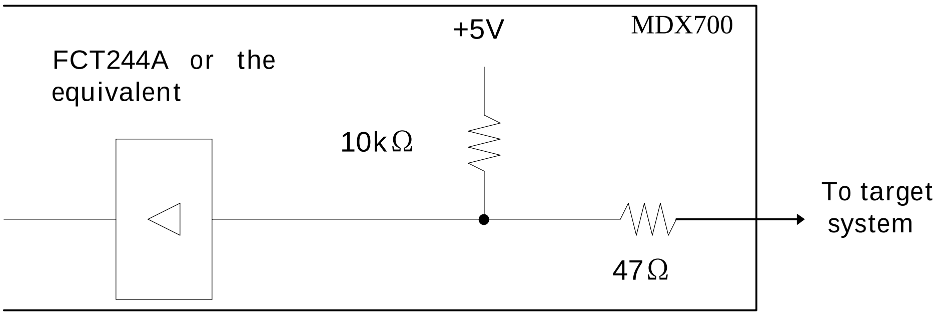
Step 5

9. When using the RESET signal, connect the MDX700 to the target system, and repeat the above steps beginning with Step 2. If an error occurs, check to make sure that the target system is connected to the proper device. Alternatively, increase the **TIMER** value in the configuration file.

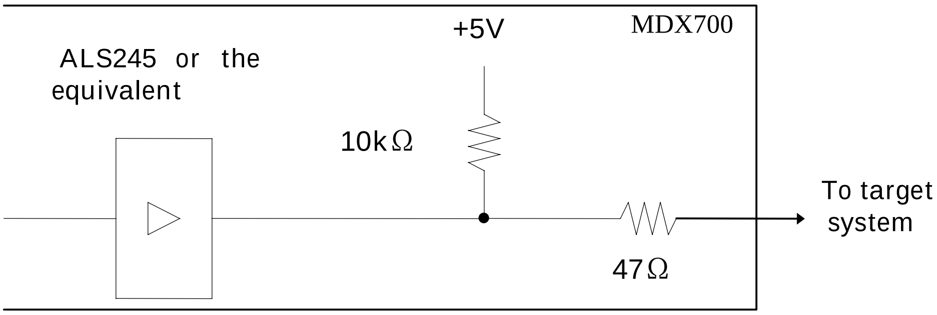
10. If the monitor program has been modified, specify the revised program, and repeat the above steps beginning with Step 2. If an error occurs, make sure that the program was not modified incorrectly.

APPENDIX P PROBING THE TARGET SYSTEM

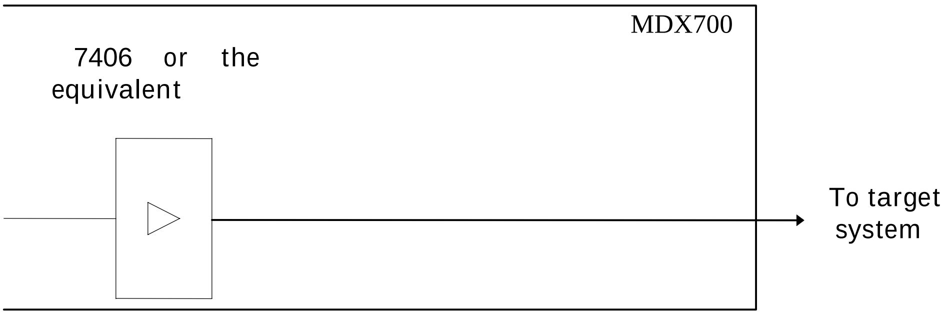
ADDRESS, CS, OE



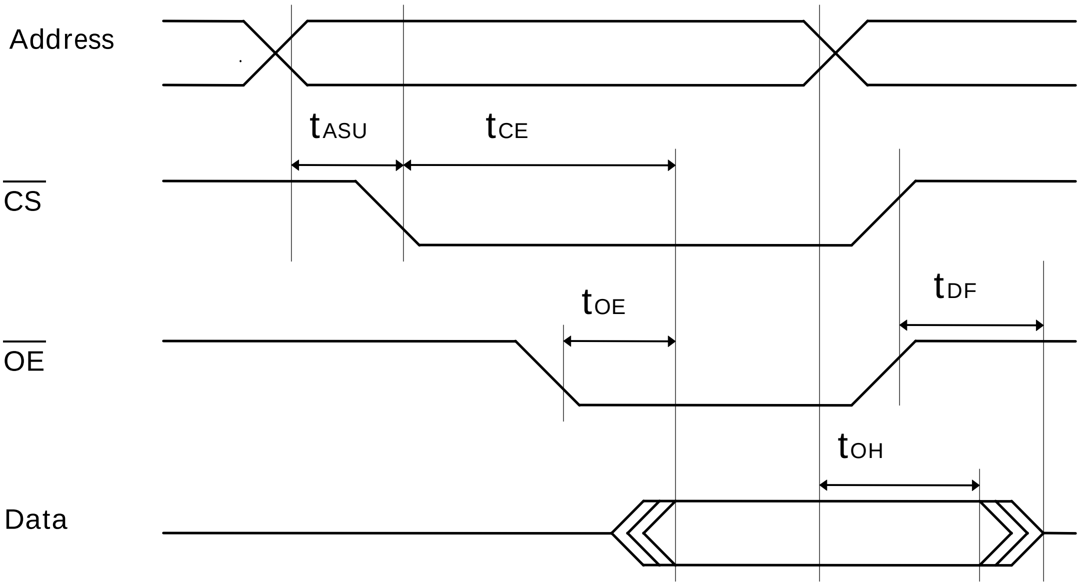
Data



RESET, NMI



APPENDIX Q DATA ACCESS TIMING



	min	max	
t_{ASU}	0		CS assert to address valid
t_{CE}		75n sec	CS access time
t_{OE}		50n sec	OE access time
t_{OH}	10n sec		output hold time
t_{DF}		40n sec	output floating time

APPENDIX R ROM CABLE CONNECTOR PIN ASSIGNMENTS

The table below shows connector-side pin assingments for connection to a ROM probe. The code inside the parentheses are connector codes.

RC28AD/RC28D

(Hirose HIF6A-40PA-1.27DSA)

DATA00	A01	B01	ROMCODE0
GND	A02	B02	ROMCODE1
DATA01	A03	B03	ROMCODE2
GND	A04	B04	ROMCODE3
DATA02	A05	B05	ADDR00
GND	A06	B06	ADDR01
DATA03	A07	B07	ADDR02
GND	A08	B08	ADDR03
DATA04	A09	B09	ADDR04
GND	A10	B10	ADDR05
DATA05	A11	B11	ADDR06
GND	A12	B12	ADDR07
DATA06	A13	B13	ADDR08
GND	A14	B14	ADDR09
DATA07	A15	B15	ADDR10
GND	A16	B16	ADDR11
CE*	A17	B17	ADDR12
GND	A18	B18	ADDR13
OE*	A19	B19	ADDR14
GND	A20	B20	ADDR15

RC32AD/RC32D

(Hirose HIF6A-52PA-1.27DSA)

DATA00	A01	B01	ROMCODE0
GND	A02	B02	ROMCODE1
DATA01	A03	B03	ROMCODE2
GND	A04	B04	ROMCODE3
DATA02	A05	B05	ADDR00
GND	A06	B06	ADDR01
DATA03	A07	B07	ADDR02
GND	A08	B08	ADDR03
DATA04	A09	B09	ADDR04
GND	A10	B10	ADDR05
DATA05	A11	B11	ADDR06
GND	A12	B12	ADDR07
DATA06	A13	B13	ADDR08
GND	A14	B14	ADDR09
DATA07	A15	B15	ADDR10
GND	A16	B16	ADDR11
CE*	A17	B17	ADDR12
GND	A18	B18	ADDR13
OE*	A19	B19	ADDR14
GND	A20	B20	ADDR15
OPEN	A21	B21	ADDR16
OPEN	A22	B22	ADDR17
OPEN	A23	B23	ADDR18
OPEN	A24	B24	ADDR19
OPEN	A25	B25	ADDR20
OPEN	A26	B26	ADDR21

APPENDIX R. ROM CABLE CONNECTOR PIN ASSIGNMENTS

RC40AD/RC40D

(Hirose HIF6A-68PA-1.27DSA)

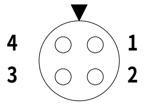
ROMCODE0	A01	B01	DATA00
ROMCODE1	A02	B02	GND
ROMCODE2	A03	B03	DATA01
ROMCODE3	A04	B04	GND
ADDR00	A05	B05	DATA02
ADDR01	A06	B06	GND
ADDR02	A07	B07	DATA03
ADDR03	A08	B08	GND
ADDR04	A09	B09	DATA04
ADDR05	A10	B10	GND
ADDR06	A11	B11	DATA05
ADDR07	A12	B12	GND
ADDR08	A13	B13	DATA06
ADDR09	A14	B14	GND
ADDR10	A15	B15	DATA07
ADDR11	A16	B16	GND
ADDR12	A17	B17	CEL*
ADDR13	A18	B18	GND
ADDR14	A19	B19	OEL*
ADDR15	A20	B20	GND
ADDR16	A21	B21	DATA08
ADDR17	A22	B22	GND
ADDR18	A23	B23	DATA09
ADDR19	A24	B24	GND
ADDR20	A25	B25	DATA10
ADDR21	A26	B26	GND
OPEN	A27	B27	DATA11
OPEN	A28	B28	GND
DATA15	A29	B29	DATA12
GND	A30	B30	GND
CEU*	A31	B31	DATA13
GND	A32	B32	GND
OEU*	A33	B33	DATA14
GND	A34	B34	GND

0:GND 1:Open

ROMCODE 3 2 1 0	ROMprobe
0 0 0 0	32K x 8bit
0 0 0 1	64K x 8bit
0 0 1 0	128K x 8bit
0 0 1 1	256K x 8bit
0 1 0 0	512K x 8bit
0 1 0 1	1024K x 8bit
0 1 1 0	2048K x 8bit
0 1 1 1	4096K x 8bit
1 0 0 0	
1 0 0 1	64K x 16bit
1 0 1 0	128K x 16bit
1 0 1 1	256K x 16bit
1 1 0 0	512K x 16bit
1 1 0 1	1024K x 16bit
1 1 1 0	2048K x 16bit
1 1 1 1	4096K x 16bit

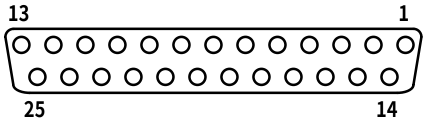
APPENDIX S PWR CONNECTOR PIN ASSIGNMENTS

(Hirose HR10A-7R-4S)



Pin number	Signal name	Description
1	+5V	
2	GND	
3	Not connected	
4	Not connected	

APPENDIX T RS-232C CONNECTOR PIN ASSIGNMENTS Ethernet



Pin number	Signal name	Description
2	RxD	Receive data
3	TxD	Send data
4	CTS	Clear to send
5	RTS	Request to send
6	Shorted with pin 20	
7	GND	Grounding
20	Shorted with pin 6	
Other	Not connected	

APPENDIX U EXAMPLES OF CONFIGURATION FILES

```
*
*  MDX700 configuration for AVME-140 (AVAL DATA 68040 VME board)
*
MONITOR         mdx68k.abs      ; monitor program
CPU             68040           ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             32              ; bus width
ROM             0xFF400000      ; ROM start address
ROMSIZE         0x00040000      ; ROM size
WORKROM         0xFF43E000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0xFFF9F000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0xFF400000      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_PC          0x40008000
REG_SR          0x2700
REG_ISP         0x40005000
REG_MSP         0x40006000
REG_USP         0x40007000
REG_VBR         0x40000000
REG_DTT0        0x4000C000
REG_DTT1        0x00FFC040
REG_ITT0        0x00000000
REG_ITT1        0x00FFC000
```

```
*
*  MDX700 configuration for DVE-R4000/20 (DENSAN R4400 VME board)
*
MONITOR         mdxmips.abs     ; monitor program
CPU             R4400           ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             32              ; bus width
ROM             0xBFC00000      ; ROM start address
ROMSIZE         0x00040000      ; ROM size
WORKROM         0xBFC3C000      ; work ROM start address
WORKROMSIZE     0x00004000      ; work ROM size
WORKRAM         0xA000F000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0xBFC00000      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_PC          0xBFC08000      ; program counter
REG_R29         0xA0008000      ; stack pointer
REG_CP12        0x20410000      ; Status.CU1=1 Status.BEV=1 Status.DE=1
REG_CP16        0x00008003      ; Config.BE=1 Config.K0=3
*
*  USER_INIT code
*
INIT_CODE       0x3c01bfbf      ; lui     $1, 0xBFBF
INIT_CODE       0x3c023800      ; lui     $2, 0x3800      # CSR0 = 0x3800C000
INIT_CODE       0x3442c000      ; ori     $2, $2, 0xC000
INIT_CODE       0xac220000      ; sw      $2, 0x0000($1)
INIT_CODE       0x3c020000      ; lui     $2, 0x0000      # CSR3 = 0x00000103
INIT_CODE       0x34420103      ; ori     $2, $2, 0x0103
INIT_CODE       0xac22000c      ; sw      $2, 0x000C($1)
INIT_CODE       0x3c020010      ; lui     $2, 0x0010      # CSR19 = 0x001020FF
INIT_CODE       0x344220ff      ; ori     $2, $2, 0x20FF
INIT_CODE       0xac22004c      ; sw      $2, 0x004C($1)
INIT_CODE       0x3c027766      ; lui     $2, 0x7766      # CSR24 = 0x77665550
INIT_CODE      0x34425550     ; ori    $2, $2, 0x5550
INIT_CODE      0xac220060     ; sw     $2, 0x0060($1)
INIT_CODE      0x3c020555     ; lui    $2, 0x0555      # CSR25 = 0x05557432
INIT_CODE      0x34427432     ; ori    $2, $2, 0x7432
INIT_CODE      0xac220064     ; sw     $2, 0x0064($1)
INIT_CODE      0x3c027654     ; lui    $2, 0x7654      # CSR26 = 0x76543210
INIT_CODE      0x34423210     ; ori    $2, $2, 0x3210
INIT_CODE      0xac220068     ; sw     $2, 0x0068($1)
INIT_CODE      0x3c027654     ; lui    $2, 0x7654      # CSR27 = 0x76543210
INIT_CODE      0x34423210     ; ori    $2, $2, 0x3210
INIT_CODE      0xac22006c     ; sw     $2, 0x006C($1)
INIT_CODE      0x3c028000     ; lui    $2, 0x8000      # IFR0 = 0x80000000
INIT_CODE      0x34420000     ; ori    $2, $2, 0x0000
INIT_CODE      0xac220070     ; sw     $2, 0x0070($1)
INIT_CODE      0x3c020000     ; lui    $2, 0x0000      # IFR3 = 0x00000000
INIT_CODE      0x34420000     ; ori    $2, $2, 0x0000
INIT_CODE      0xac22007c     ; sw     $2, 0x007C($1)
INIT_CODE      0x03e00008     ; jr     $ra             # return
INIT_CODE      0x00000000     ; nop    # in delay slot
```

```
*
*  MDX700 configuration for Sony R3051 board
*
MONITOR         mdxmipsl.abs    ; monitor program
CPU             R3051_LE        ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             8               ; bus width
ROM             0xBFC00000      ; ROM start address
ROMSIZE         0x00080000      ; ROM size
WORKROM         0xBFC7c000      ; work ROM start address
WORKROMSIZE     0x00004000      ; work ROM size
WORKRAM         0xA01bf000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0xbfc00000      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R29         0xA01F8000      ; stack pointer
REG_CP12        0x20410000      ; Status.CU1=1 Status.BEV=1 Status.DE=1
REG_CP16        0x00008003      ; Config.BE=1 Config.K0=3
REG_CP10        0x00005654      ; Portsize
REG_CP2         0x6EFF4B00      ; Busctrl
*
*  Initialize code
*
INIT_CODE       0x24025654      ; li $2, 0x00005654
INIT_CODE       0x40825000      ; mtc0 $2, C0_PORTSIZE
INIT_CODE       0x3c026FFF      ; li $2, 0x6FFFCB00
INIT_CODE       0x3442CB00      ; mtc0 $2, C0_BUSCTRL
INIT_CODE       0x40821000
INIT_CODE       0x03e00008      ; jr $ra
INIT_CODE       0x00000000      ; nop
```

```
*
*  MDX700 configuration for MVME1603 (Motorola PowerPC 603 VME board)
*
MONITOR         mdxppc.abs      ; monitor program
CPU             PPC603          ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             8               ; bus width
ROM             0xFFF80000      ; ROM start address
ROMSIZE         0x00080000      ; ROM size
WORKROM         0xFFFFE000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x00780000      ; work RAM start address
WORKRAMSIZE     0x00010000      ; work RAM size
RESETVECTOR     0xFFFFFFFF      ; RESET vector addres (ffffffff = not used)
ABORTVECTOR     0x00000100      ; ABORT vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_PVR         0x00030302      ; processor version register
REG_MSR         0x00002000      ; machine status register
REG_GPR1        0x00018000      ; stack pointer
```

```
*
*  MDX700 configuration for DVE-SH7700 (DENSAN SH-3 VME board)
*
MONITOR         mdxsh.abs       ; monitor program
CPU             SH7708          ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             32              ; bus width
ROM             0xA0000000      ; ROM start address
ROM_IMAGE       0x80000000
ROMSIZE         0x00040000      ; ROM size
WORKROM         0xA003E000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x047FF000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0xA0000000      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R15         0x04040000      ; stack pointer
REG_PC          0x04000000
REG_SR          0x400000F0
REG_VBR         0xA0000000
*
*  USER_INIT code
*
INIT_CODE       0xD002          ; mov     #0x0480007C,r0
INIT_CODE       0xE100          ; mov     #0x0,r1
INIT_CODE       0x2012          ; mov.l   r1, @r0
INIT_CODE       0x000B          ; rts
INIT_CODE       0x0009          ; nop
INIT_CODE       0x0009          ; nop
INIT_CODE       0x0480
INIT_CODE       0x007C
```

```
*
*  MDX700 configuration for DVE-V830/20 (DENSAN V830 VME board)
*
MONITOR         mdxv800.abs     ; monitor program
CPU             V830            ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             32              ; bus width
ROM             0x7FFC0000      ; ROM start address
ROM_IMAGE       0xFFFC0000      ; ROM image start address
ROMSIZE         0x00040000      ; ROM size
WORKROM         0x7FFFD000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x00010000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0x7FFFFFF0      ; RESET vector addres (ffffffff = not used)
TIMER           200000          ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R3          0x00008000      ; stack pointer
REG_PC          0x00002000      ; program counter
*
*  USER_INIT code
*
INIT_CODE       0xfc00          ; out.w   r0, 0x007C[r0]
INIT_CODE       0x007c
INIT_CODE       0x181f          ; jmp     [lp]                  # return
```

```
*
*  MDX700 configuration for (NEC V830 sound middleware evaluation board)
*
MONITOR         mdxv800.abs     ; monitor program
CPU             V830            ; CPU type
PORT            0x010003D0      ; Host interface I/O address
BUS             32              ; bus width
ROM             0x7FC00000      ; ROM start address
ROMSIZE         0x20400000      ; ROM size
WORKROM         0x7FFFD000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x103FF000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0x7FFFFFF0      ; RESET vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R3          0x10008000      ; stack pointer
REG_PC          0x10000000      ; program counter
```

```
*
*  MDX700 configuration for RT-V831 (NEC V831 evaluation board)
*
*  Note: connect D00-D07 ROM cable to P0 connector, and D08-D15 to P1.
*  Note: connect blue cable to TP_RESET, and yellow cable to TP_NMI.
*  Note: increase TIMER value if timeout error detected.
*
MONITOR         mdxv800.abs     ; monitor program
CPU             V831            ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             16              ; bus width
ROM             0x4FFE0000      ; ROM start address
ROM_IMAGE       0xFFFE0000
ROMSIZE         0x00020000      ; ROM size
WORKROM         0x4FFE8000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0x000FF000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0x4FFFFFF0      ; RESET vector addres (ffffffff = not used)
TIMER           400000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_R3          0x00010000      ; stack pointer
REG_PC          0x00008000      ; program counter
*
*  USER_INIT code
*
INIT_CODE       0xbd40          ; movhi   0xC000, zero, r10
INIT_CODE       0xc000
INIT_CODE       0xb160          ; ori     0x0082, zero, r11
INIT_CODE       0x0082
INIT_CODE       0xf56a          ; out.h   r11, 0x0020[r10]
INIT_CODE       0x0020
INIT_CODE       0xb160          ; ori     0x800F, zero, r11
INIT_CODE       0x800f
INIT_CODE       0xf56a          ; out.h   r11, 0x0022[r10]
INIT_CODE       0x0022
INIT_CODE       0xb160          ; ori     0x0010, zero, r11     # CS4 I/O
INIT_CODE       0x0010
INIT_CODE       0xf56a          ; out.h   r11, 0x0010[r10]
INIT_CODE       0x0010
INIT_CODE       0xbd40          ; movhi   0x0400, zero, r10     # clear LED
INIT_CODE       0x0400
INIT_CODE       0x417f          ; mov     -1, r11
INIT_CODE       0xfd6a          ; out.w   r11, 0x0000[r10]
INIT_CODE       0x0000
INIT_CODE       0x181f          ; jmp     [lp]                  # return
```

```
*
*  MDX700 configuration for YP3C-1 (Beyond the river PowerPC 403GC board)
*
*  Note: connect U4 ROM to P0 connector, U7 ROM to P1 connector.
*
MONITOR         mdxppc.abs      ; monitor program
CPU             PPC403          ; CPU type
PORT            0x0100          ; Host interface I/O address
BUS             16              ; bus width
ROM             0xFFFC0000      ; ROM start address
ROM_IMAGE       0xFFF00000
ROMSIZE         0x00040000      ; ROM size
WORKROM         0xFFFFD000      ; work ROM start address
WORKROMSIZE     0x00002000      ; work ROM size
WORKRAM         0xFFD7F000      ; work RAM start address
WORKRAMSIZE     0x00001000      ; work RAM size
RESETVECTOR     0xFFFFFFFC      ; RESET vector addres (ffffffff = not used)
ABORTVECTOR     0xFFFFFFFF      ; ABORT vector addres (ffffffff = not used)
TIMER           80000           ; RESET & communication port timeout
*
*  Register Initialize
*
REG_PC          0xFFD10000      ; program counter
REG_PVR         0x00200200      ; processor version register
REG_MSR         0x00000000      ; machine status register
REG_EVPR        0xFFD00000      ; exception vector prefix register
REG_GPR1        0xFFD78000      ; stack pointer
REG_BR0         0xFF18BFFE      ; (default)
REG_BR1         0xFD1904E0      ; wait 4
REG_BR2         0x0018BFFE      ; wait 64
REG_BR3         0x1018BFFE      ; wait 64
*
*  USER_INIT code
*
INIT_CODE       0x3c60ff18      ; lis     r3, 0xFF18
INIT_CODE       0x6063bffe      ; ori     r3, r3, 0xBFFE
INIT_CODE       0x7c602386      ; mtbr0   r3
INIT_CODE       0x3c60fd19      ; lis     r3, 0xFD19
INIT_CODE       0x606304e0      ; ori     r3, r3, 0x04E0
INIT_CODE       0x7c612386      ; mtbr1   r3
INIT_CODE       0x3c600018      ; lis     r3, 0x0018
INIT_CODE       0x6063bffe      ; ori     r3, r3, 0xBFFE
INIT_CODE       0x7c622386      ; mtbr2   r3
INIT_CODE       0x3c601018      ; lis     r3, 0x1018
INIT_CODE       0x6063bffe      ; ori     r3, r3, 0xBFFE
INIT_CODE       0x7c632386      ; mtbr3   r3
INIT_CODE       0x4e800020      ; blr
```

```
*
*  MDX700 configuration for SHARP ARM790 Evaluation Board
*
*  Note: connect blue cable to M51957 2pin, and yellow cable to .
*  Note: connect yellow cable to RA1 9pin(INT0). And INT0 should be pull-up.
*
MONITOR         mdxarml.abs      ; monitor program
CPU             ARM7_LE          ; CPU type
PORT            0x0100           ; Host interface I/O address
BUS             8                ; bus width
ROM             0x00000000       ; ROM start address
ROMSIZE         0x00080000       ; ROM size
WORKROM         0x0007C000       ; work ROM start address
WORKROMSIZE     0x00004000       ; work ROM size
WORKRAM         0x0027F000       ; work RAM start address
WORKRAMSIZE     0x00001000       ; work RAM size
RESETVECTOR     0x00000000       ; RESET vector addres (ffffffff = not used)
ABORTVECTOR     0x00000018       ; ABORT vector addres (ffffffff = not used)
TIMER           300000           ; RESET & communication port timeout
*
*  Register Initialize
*
*REG_CPSR        0x00000013       ; status reg in supervisor mode
REG_CPSR        0x00000010       ; status reg in user mode
REG_PC          0x00008000       ; program counter
REG_R13         0x60000800       ; stack pointer
REG_R13_FIQ     0x60000800       ; stack pointer
REG_R13_SVC     0x60000800       ; stack pointer
REG_R13_ABT     0x60000800       ; stack pointer
REG_R13_IRQ     0x60000800       ; stack pointer
REG_R13_UND     0x60000800       ; stack pointer
*
*  USER_INIT code
*
INIT_CODE       0xe1a0000e         ; mov     r0, lr
INIT_CODE       0xeb000000         ; bl      .+8
INIT_CODE       0x00000050      ; .data.w INITBL -.
INIT_CODE       0xe1a0300e      ; mov     r3, lr
INIT_CODE       0xe5933000      ; ldr     r3, [r3]
INIT_CODE       0xe083300e      ; add     r3, r3, lr      # r3 = INITBL
INIT_CODE       0xe1a0e000      ; mov     lr, r0
INIT_CODE       0xe5930000      ; ldr     r0, [r3]    [loop:]
INIT_CODE       0xe2833004      ; add     r3, r3, 4
INIT_CODE       0xe3500000      ; cmps    r0, 0
INIT_CODE       0x01a0f00e      ; moveq   pc, lr
INIT_CODE       0xe5931000      ; ldr     r1, [r3]
INIT_CODE       0xe2833004      ; add     r3, r3, 4
INIT_CODE       0xe5932000      ; ldr     r2, [r3]
INIT_CODE       0xe2833004      ; add     r3, r3, 4
INIT_CODE       0xe3520008      ; cmps    r2, 8
INIT_CODE       0x05c01000      ; streqb  r1, [r0]
INIT_CODE       0xe3520010      ; cmps    r2, 16
INIT_CODE       0x01c010b0      ; streqh  r1, [r0]
INIT_CODE       0xe3520020      ; cmps    r2, 32
INIT_CODE       0x05801000      ; streq   r1, [r0]
INIT_CODE       0xeafffff0      ; b       loop
                                ; INITBL:
INIT_CODE       0xFFFFA404      ; LSCR = 0x03
INIT_CODE       0x00000003
INIT_CODE       0x00000008
INIT_CODE       0xFFFFA104      ; BCR1 = 0x0000A300
INIT_CODE       0x0000A300
INIT_CODE       0x00000020
INIT_CODE       0xFFFFA040      ; SDR0 = 0x00007802
INIT_CODE       0x00007802
INIT_CODE       0x00000020
INIT_CODE       0xFFFFA000      ; START0 = 0x00200000
INIT_CODE       0x00200000
INIT_CODE       0x00000020
INIT_CODE       0xFFFFA020      ; STOP0 = 0x00280000
INIT_CODE       0x00280000
INIT_CODE       0x00000020
INIT_CODE        0xFFFFA800        ; ICR0 = 0x00000000
INIT_CODE        0x00000000
INIT_CODE        0x00000020
INIT_CODE        0xFFFFA80C        ; IRQER = 0x00000001
INIT_CODE        0x00000001
INIT_CODE        0x00000020
INIT_CODE        0xFFFFA808        ; ICLR = 0x000003FF
INIT_CODE        0x000003FF
INIT_CODE        0x00000020
INIT_CODE        0x00000000        ; # End of Table
```