MDX700
User's Manual

Rev.2.21 1998/11/25

for

Parallel Interface MDX700

Ethernet Interface MDX700

ご注意

- 本マニュアルの一部または全部を無断で複製することはできません。
- 本製品を運用した結果の影響については、いかなる責任も負いません。
- 本製品の仕様および本マニュアルの内容は予告なく変更することがあります。
- MS-DOS、Windows 95、Windows NT は、Microsoft の登録商標です。
- SunOS、Solaris は、Sun Microsystems の登録商標です。
- MULTI は、Green Hills Software の登録商標です。
- SingleStep は、Software Development Systems の登録商標です。
- XRAY は、Mentor Graphics の登録商標です。

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

この度は、MDX700 をご購入いただきまして、誠にありがとうございます。

本マニュアルの内容は、次のとおりです。

第一章 概要

MDX700 の概要、動作環境、製品構成、ハードウェア構成、ソフトウェア構成について記述しています。

第二章 ハードウェアの接続

MDX700 とホストの接続方法、MDX700 とターゲットシステムの接続方法について記述しています。

第三章 デバッガのインストール

デバッガのインストール方法について記述しています。

第四章 デバッガの環境設定

デバッガを使用する前に必要な、環境設定の方法について記述しています。
主に、コンフィグレーション ファイル(mdx.cfg)の変更方法について記述しています。

第五章 デバッガの起動

デバッガの起動方法について記述しています。

第六章 MDXDEB コマンド

簡易デバッガ MDXDEB のコマンドの使い方について記述しています。

第七章 MDXCVT

ファイル変換ツール MDXCVT の操作方法について記述しています。MDXCVT は、S レコード
ファイルまたは IEEE695 ファイルを高速ダウンロードするためのツールです。

付録

仕様、ターゲット システムの制限事項、対応 ROM、対応 CPU、動作原理などの技術情報について記述しています。

本マニュアルでは、次のシンボルを使用します。本文中にこれらのシンボルがある場合、その環境に限定した説明であることをあらわします。

Parallel	MDX700 のインターフェースがパラレル インターフェース仕様
Ethernet	MDX700 のインターフェースがイーサネット インターフェース仕様
PC/AT	ホストが PC/AT、または互換機
PC-98	ホストが PC-98
SPARC	ホストが SPARCstation
68000	CPU が 68000 ファミリ
ARM	CPU が ARM ファミリ (THUMB 含をむ)
THUMB	CPU が THUMB
MIPS	CPU が MIPS ファミリ
PowerPC	CPU が PowerPC ファミリ
SH	CPU が SuperH RISC engine ファミリ (SH-1、SH-2、または SH-3)
SH-1	CPU が SH7030 シリーズまたは SH7020 シリーズ
SH-2	CPU が SH7600 シリーズ
SH-3	CPU が SH7700 シリーズ
V800	CPU が V800 シリーズ(V830、V850、V850E を含む)
V830	CPU が V830 ファミリ
V850	CPU が V850 ファミリ
V850E	CPU が V850E ファミリ

もくじ

第一章 概要

MDX700 の概要、動作環境、製品構成、ハードウェア構成、ソフトウェア構成について記述しています。

MDX700 を初めてお使いになるかたは、この章をお読みください。

1.1 MDX700 の概要

MDX700 は、組み込みシステム向けの開発支援装置です。ROM インサーキット方式を採用しているため、次の特長があります。

- CPU の形状に依存しません。
- デバッガの交換でいろいろな CPU に対応できます。
- 高速 CPU のターゲット システムでも安定して動作します。
- ダウンロードが高速です。(256KB/秒 Parallel)

MDX700 はホスト上のデバッガで操作します。デバッガは、CPU や開発言語の環境に合わせて選択できます。MDX700 に対応しているデバッガは、次のとおりです。

- 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 (製品名XHI68KMD)
- XRAY68K 2.2a on MS-DOS PC-98 (製品名XHI68KMD)
- XRAY68K 3.4 on SunOS/Solaris (製品名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)

これらのデバッガは、ユーザプログラムのダウンロードや実行、ブレークポイントの設定、変数の参照と書き換え、レジスタの参照と書き換えなどの基本的なデバッグ機能を持っています。

MDX700 はターゲット システムの ROM ソケットに接続して使用しますが、RAM や I/O などの ROM 以外の資源に対してもアクセスすることができます。したがって、ユーザ プログラムを RAM 領域にダウンロードしたり、I/O へデータを書き込むこともできます。

MDX700 およびデバッガを使用する前には、次の準備作業が必要です。第二章から第四章までを参照して行なってください。

- MDX700 とホストの接続
- MDX700 とターゲット システムの接続
- デバッガのインストール
- デバッガの環境設定
- MDX700 の IP アドレスの設定 Ethernet
- MDX700 の IP アドレスとホスト名の登録 Ethernet

デバッガの起動方法については、第五章を参照してください。もし、デバッガが正常に起動できない場合は、準備作業に誤りがないか、もう一度確認してください。どうしても問題が解決できない場合は、「付録 O トラブル シューティング」を参照してください。

デバッガとして MDXDEB を使用する場合は、第六章を参照してください。また、MDXDEB 以外のデバッガを使用する場合は、デバッガのリリース ノートとマニュアルを参照してください。

デバッガの使用中に問題が発生した場合は、「付録 C 注意事項」、「付録 K ユーザ プログラムとモニタ プログラムの共存」を参照してください。

MDXCVT の使い方については、第七章を参照してください。デバッガに付属する MDXCVT は、ファイル変換ツールです。MDXCVT は、S レコード ファイルまたは IEEE695 ファイルを、高速でダウンロードできる形式のファイルに変換します。

1.2 動作環境

MDX700 およびデバッガ(MDXDEB)を動作させるための環境は、次のとおりです。

Parallel

- ホスト　ISA ボードが接続できる PC/AT、または互換機
  C-BUS ボードが接続できる PC-98
- OS　MS-DOS、Windows 3.1、Windows 95
- メモリ　8M バイト以上
- ハード ディスク　1M バイト以上の空き容量
- ターゲット システム　ROM と RAM が完全に動作している
  ROM に 16K バイト、RAM に 4K バイトの空き容量がある
  RESET 信号、NMI 信号が接続できる方が望ましい

Ethernet

- ホスト　SPARCstation
- OS　SunOS 4.x、Solaris 5.x
- メモリ　8M バイト以上
- ハード ディスク　1M バイト以上の空き容量
- ターゲット システム　ROM と RAM が完全に動作している
  ROM に 16K バイト、RAM に 4K バイトの空き容量がある
  RESET 信号、NMI 信号が接続できる方が望ましい

MDXDEB 以外のデバッガでは、動作条件が異なります。詳しくは、デバッガのリリースノートを参照してください。

コンパイラの動作環境については、コンパイラのマニュアルを参照してください。

ターゲット システムの動作環境については、「付録 B ターゲット システムの制限事項」を参照してください。

1.3 製品構成

MDX700 の製品構成は、次のとおりです。万一、欠品がございましたら、弊社までご連絡ください。なお、電圧変換アダプタ MDX003 は、オプションです。

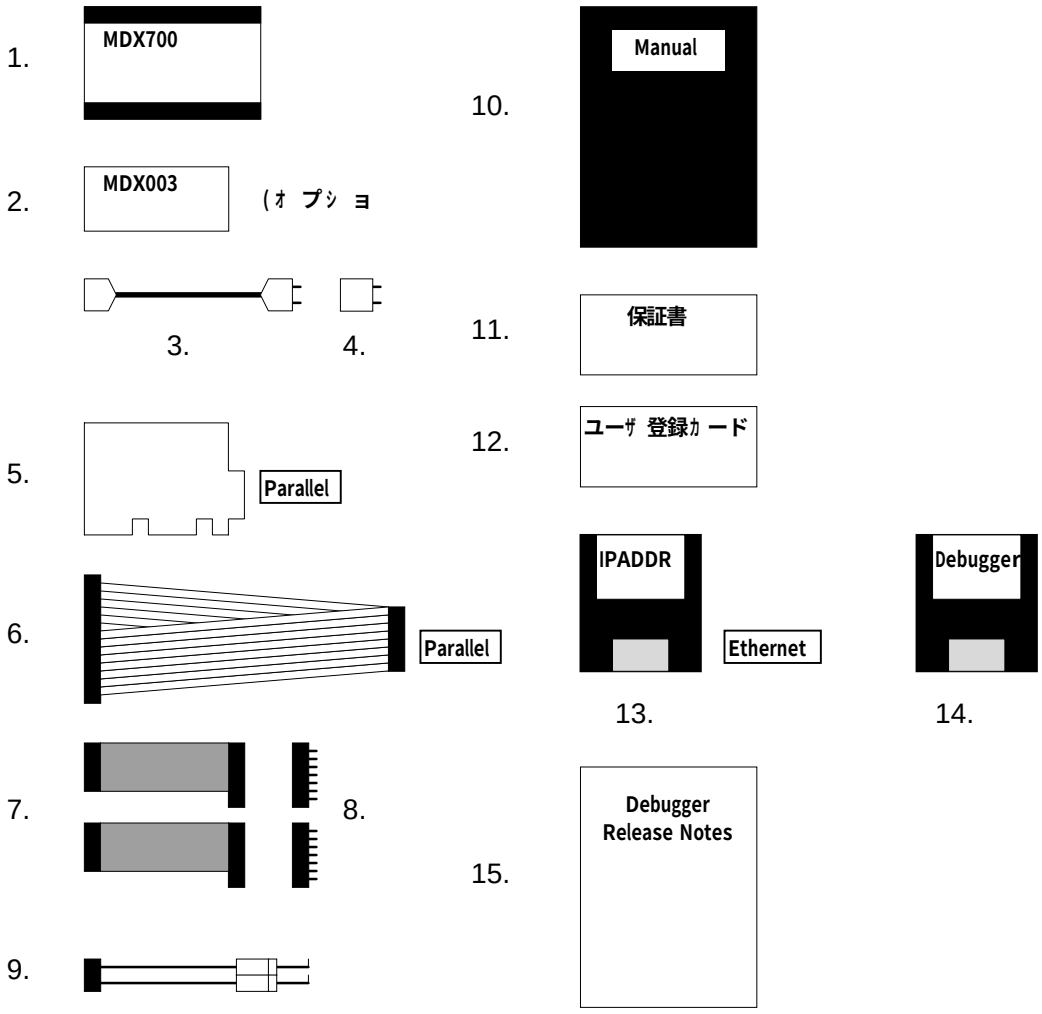


図 1-1 製品構成

1. MDX700 本体
2. MDX003（オプション）
3. 電源ケーブル
4. 3P-2P 変換ソケット
5. パラレル インターフェース ボード（ISA ボード、または C-BUS ボード） Parallel
6. パラレル インターフェース ケーブル Parallel
7. ROM ケーブル（次ページ以降参照）
8. ROM プローブ（次ページ以降参照）
9. 外部トリガ ケーブル
10. ユーザーズ マニュアル
11. 保証書
12. ユーザー登録カード
13. IP アドレス設定プログラム用フロッピー ディスク Ethernet
14. デバッガ用フロッピー ディスク
15. デバッガ用リリース ノート

【注意】Parallel または Ethernet は、それぞれのインターフェース仕様の MDX700 だけに含まれている製品です。

【注意】ROM ケーブルと ROM プローブの製品構成については、次ページ以降を参照してください。

【重要】ユーザー登録カードは、必要事項をご記入の上、弊社までご返送ください。

MDX700 の製品構成の中で、ROM ケーブルと ROM プローブは、ROM の構成によって異なります。ROM の構成と、ROM ケーブル/ROM プローブの対応表は、次のとおりです。表内の数字は、ROM ケーブルまたは、ROM プローブの個数をあらわします。

ROM の構成	ROM ケーブル		ROM プローブ	
	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

表 1-1-1 ROM の構成と ROM ケーブル/ROM プローブ対応表 1

ROM の構成	ROM ケーブル		ROM プローブ					
	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

表 1-1-2 ROM の構成と ROM ケーブル/ROM プローブ対応表 2

ROM の構成	ROM ケーブル		ROM プローブ				
	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	-

表 1-1-3 ROM の構成と ROM ケーブル/ROM プローブ対応表 3

ROM の構成	ROM ケーブル		ROM プローブ
	RC32AD	RC32D	29F040
29F040 x 1	1	-	1
29F040 x 2	1	1	2
29F040 x 4	1	3	4

表 1-1-4 ROM の構成と ROM ケーブル/ROM プローブ対応表 4

ROM の構成	ROM ケーブル		ROM プローブ		変換アダプタ
	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

表 1-1-5 ROM の構成と ROM ケーブル/ROM プローブ対応表 5

ROM ケーブルの型番の意味は、次のとおりです。

- RC28AD　　28pin アドレス付き ROM ケーブル
- RC28D　　28pinROM ケーブル
- RC32AD　　32pin アドレス付き ROM ケーブル　　(兼 27C4000/8bit、27C8000/8bit)
- RC32D　　32pinROM ケーブル　　(兼 27C4000/8bit、27C8000/8bit)
- RC40AD　　40pin アドレス付き ROM ケーブル　　(兼 27C4000/16bit、27C8000/16bit)
- RC40D　　40pinROM ケーブル　　(兼 27C4000/16bit、27C8000/16bit)

ROM プローブの型番の意味は、次のとおりです。

- B106 27256 用 ROM プローブ
- B107 27512 用 ROM プローブ
- B108 27010 用 ROM プローブ
- B109 271000 用 ROM プローブ
- B110 271024DIP 用 ROM プローブ
- B111 27020 用 ROM プローブ
- B112 27040 用 ROM プローブ
- B113 274096 用 ROM プローブ
- B117 27C4000/8bit 用 ROM プローブ
- B118 27C4000/16bit 用 ROM プローブ
- B119 27C8000/8bit 用 ROM プローブ
- B120 27C8000/16bit 用 ROM プローブ
- PL44 271024PLCC/274096PLCC 用 ROM プローブ
- 29F040　　27F040 用 ROM プローブ

変換アダプタの型番の意味は、次のとおりです。

- DSOP44RB　　SOP44pin to DIP 変換アダプタ

ROM プローブが対応している ROM の一覧については、「付録 D 対応 ROM」を参照してください。

1.4 ハードウェアの構成

MDX700 のハードウェア構成は、次のとおりです。

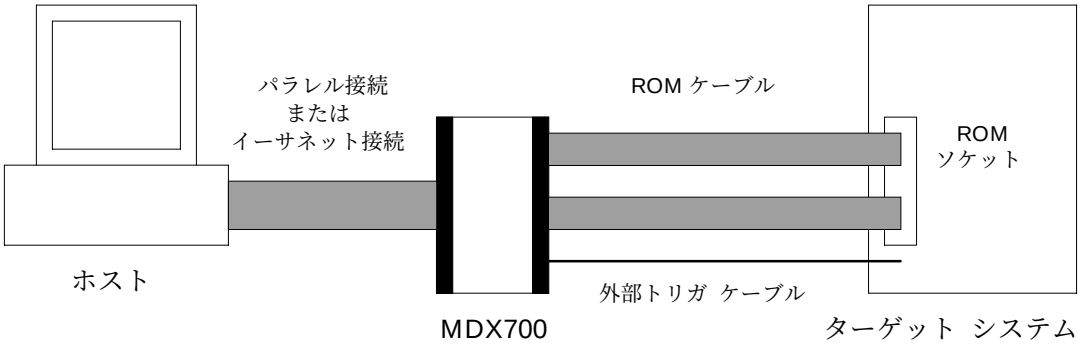


図 1-2 ハードウェア構成

Parallel の場合、ホストの拡張スロットにパラレル インターフェース ボードを実装し、MDX700 とボードをパラレル インターフェース ケーブルで接続します。

Ethernet の場合、MDX700 とホストとネットワーク接続している HUB を、10BASE-T ケーブルで接続します。10BASE-T クロス ケーブルを使用すると、ホストと MDX700 を直接接続することもできます。

MDX700 とターゲット システムの ROM ソケットとは、ROM ケーブルと ROM プローブを使って接続します。

さらに、MDX700 とターゲット システムを外部トリガ ケーブルで接続することによって、デバッガの操作性を向上させることができます。外部トリガ ケーブルなしでもデバッガは動作できますが、できるだけ接続することをおすすめいたします。

ターゲット システムの ROM 周辺回路が 3V 仕様の場合は、MDX700 とターゲット システムの間に電圧変換アダプタ MDX003 を接続します。MDX003 は、オプションです。

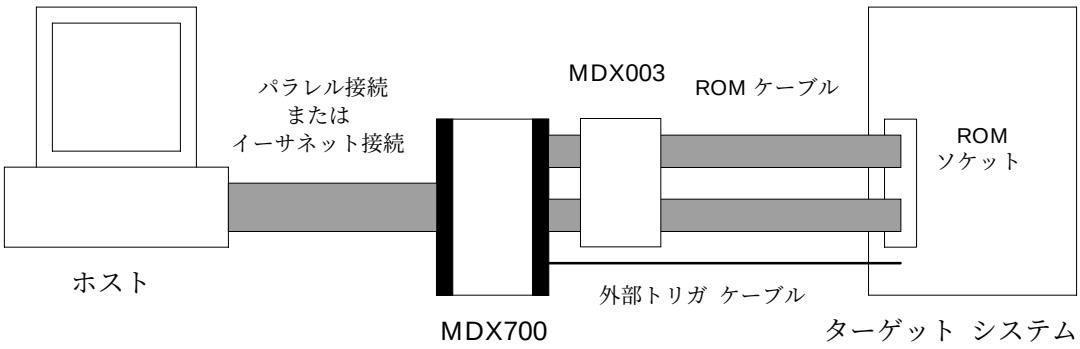


図 1-3 MDX003 を含むハードウェア構成

詳しいハードウェアの接続方法については、「第二章 ハードウェアの接続」を参照してください。

1.5 ソフトウェアの構成

MDX700 のソフトウェア構成は、次のとおりです。

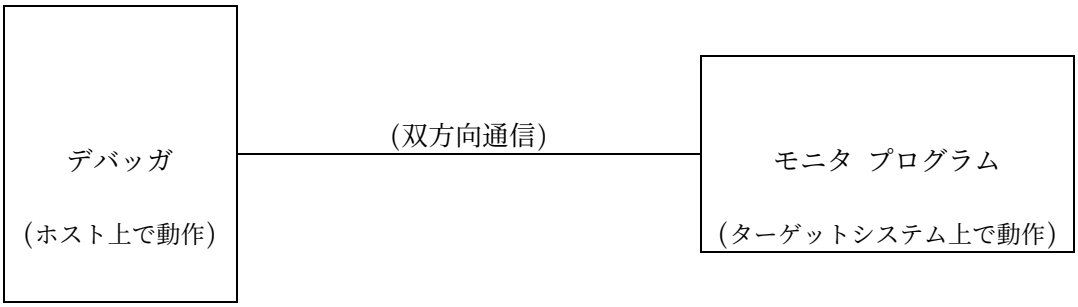


図 1-4 ソフトウェア構成

MDX700 を操作するためのデバッガは、デバッグ機能を実現するため、ターゲット システム上でモニタ プログラムを動作させています。

モニタ プログラムは、必要に応じて発生する、デバッガからの機能要求を実行し、その結果をデバッガへ返します。ROM 領域へのアクセスはデバッガが行ないますが、RAM 領域へのアクセスやユーザ プログラムの実行は、モニタ プログラムが行ないます。

モニタ プログラムとユーザ プログラムは、同じターゲット システム上で動作するため、メモリの競合をおこさないような工夫が必要になります。通常は、ユーザ プログラムが使用しない領域に、モニタ プログラムを配置するように環境設定をします。

詳しい動作原理については、「付録 H 動作原理」を参照してください。

第二章 ハードウェアの接続

MDX700 とホストの接続方法、MDX700 とターゲット システムの接続方法について記述しています。

ハードウェアの接続方法は、ホストやターゲット システムによって異なります。環境に合った項を参照してください。

【重要】 MDX700 と他の機器を接続するときは、必ず機器の電源を切ってから行なってください。

2.1 各部の名称

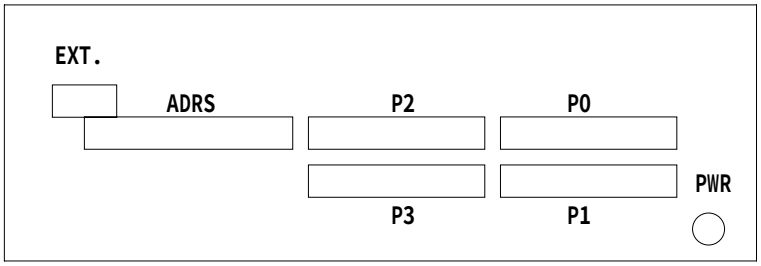


図 2-1 MDX700 フロントパネル

ADRS	ROM ケーブル接続用コネクタ
P0	ROM ケーブル接続用コネクタ
P1	ROM ケーブル接続用コネクタ
P2	ROM ケーブル接続用コネクタ
P3	ROM ケーブル接続用コネクタ
EXT.	外部トリガ ケーブル接続用コネクタ
PWR	MDX003 電源ケーブル接続用コネクタ

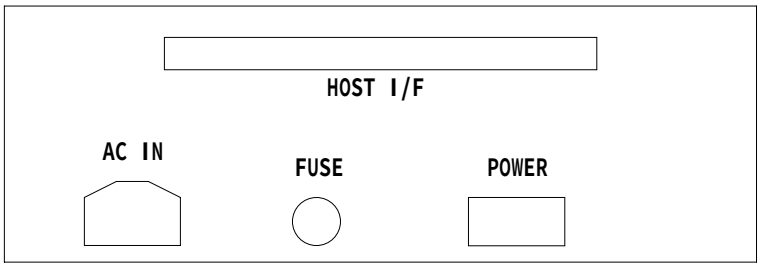


図 2-2 MDX700 リアパネル Parallel

AC IN	電源ケーブル接続用コネクタ
FUSE	フューズ
POWER	電源スイッチ
HOST I/F	パラレル インターフェース ケーブル接続用コネクタ

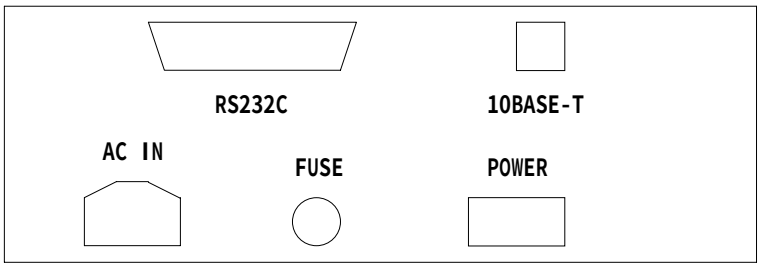


図 2-3 MDX700 リアパネル Ethernet

AC IN	電源ケーブル接続用コネクタ
FUSE	フューズ
POWER	電源スイッチ
10BASE-T	10BASE-T イーサネット ケーブル接続用コネクタ
RS232C	RS-232C ケーブル接続用コネクタ (IP アドレス設定用)

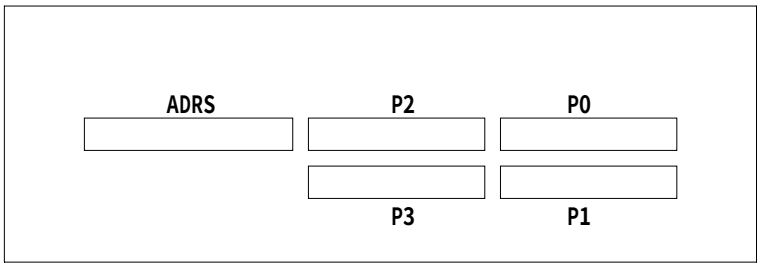


図 2-4 MDX003 フロントパネル

ADRS	ROM ケーブル接続用コネクタ
P0	ROM ケーブル接続用コネクタ
P1	ROM ケーブル接続用コネクタ
P2	ROM ケーブル接続用コネクタ
P3	ROM ケーブル接続用コネクタ

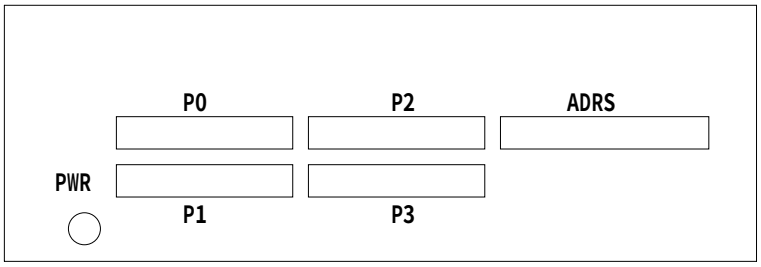


図 2-5 MDX003 リアパネル

ADRS	MDX700 接続用ケーブル
P0	MDX700 接続用ケーブル
P1	MDX700 接続用ケーブル
P2	MDX700 接続用ケーブル
P3	MDX700 接続用ケーブル
PWR	MDX700 接続用ケーブル

2.2 MDX700 とホストの接続 Parallel

MDX700 とホストは、パラレル インターフェース ボードとパラレル インターフェース ケーブルで接続します。接続手順は、次のとおりです。

出荷時のパラレルインターフェース ボードは、次の I/O アドレス空間を使用するように設定されています。既にこれらの I/O アドレスが使用されている場合は、パラレル インターフェースボードの I/O アドレスを変更してください。[*1] [*2]

PC/AT ISA ボード

- 0x0100～0x010F
- 0x0110～0x011F

PC-98 C-BUS ボード

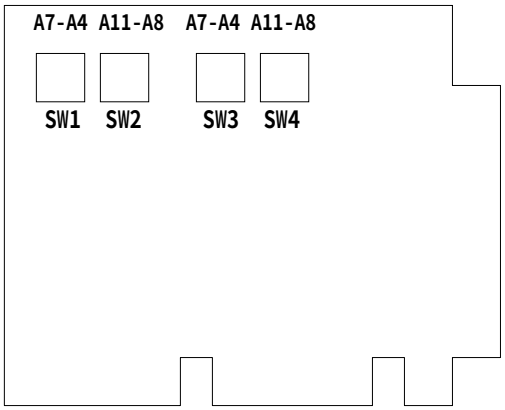
- 0x01D0～0x01DF
- 0x02D0～0x02DF

また、MDX700 を二台を同じホストに接続する場合は、それぞれのパラレルインターフェースボードに、異なる I/O アドレスを設定してください。(図 2-13-15～18 参照)

[*1] Windows 95 をお使いの場合、コントロール パネル→システム→デバイス マネージャ→コンピュータ→プロパティ→I/O ポート アドレスと選択すると、Windows 95 で使用している I/O アドレスをすべて表示させることができます。

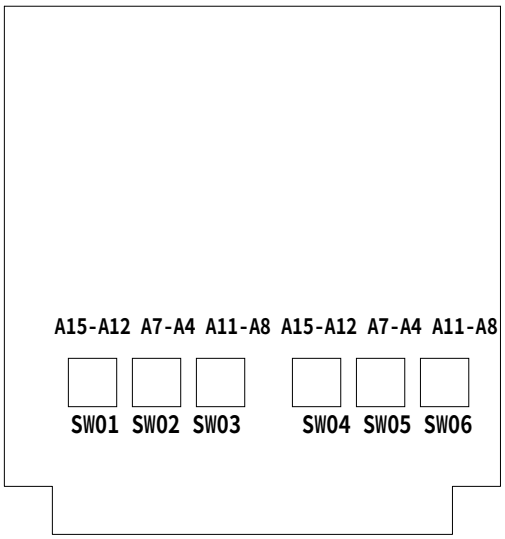
[*2] MDX700 のパラレル インターフェース ボードは、Windows 95 の Plug&Play には対応していません。

パラレルインターフェース ボードの I/0 アドレスは、ボード上のスイッチによって設定されています。スイッチの設定を変更することで、ボードの I/0 アドレスを変更することができます。



SW1 第一 I/0 アドレスの A7-A4(出荷時、0)
SW2 第一 I/0 アドレスの A11-A8(出荷時、1)
SW3 第二 I/0 アドレスの A7-A4(出荷時、1)
SW4 第二 I/0 アドレスの A11-A8(出荷時、1)

図 2-6 PC/AT パラレル インターフェース ボード (ISA ボード)



SW01 第一 I/0 アドレスの A15-A12(出荷時、0)
SW02 第一 I/0 アドレスの A11-A8(出荷時、1)
SW03 第一 I/0 アドレスの A7-A4(出荷時、D)
SW04 第二 I/0 アドレスの A15-A12(出荷時、0)
SW05 第二 I/0 アドレスの A11-A8(出荷時、2)
SW06 第二 I/0 アドレスの A7-A4(出荷時、D)

図 2-7 PC-98 パラレル インターフェース ボード (C-BUS ボード)

【注意】PC/AT ISA ボードの場合、I/O アドレスの A15-A12 は 0 固定です。

【注意】I/O アドレスを変更する場合は、第一 I/O アドレスと第二 I/O アドレスを、重複しないように設定してください。

【注意】I/O アドレスは、後述するコンフィグレーション ファイルの中で指定します。

【注意】現在の MDX700 では、第二 I/O アドレスを使用していません。コンフィグレーション ファイルの中では、第一 I/O アドレスを指定します。

I/O アドレスを設定したら、パラレルインターフェース ボードをホストの拡張スロットに組み込んでください。ボードの組み込み手順については、ホストのマニュアルを参照してください。

つぎに、パラレル インターフェース ケーブルで、パラレル インターフェース ボードと、MDX700 の HOST I/F コネクタを接続してください。

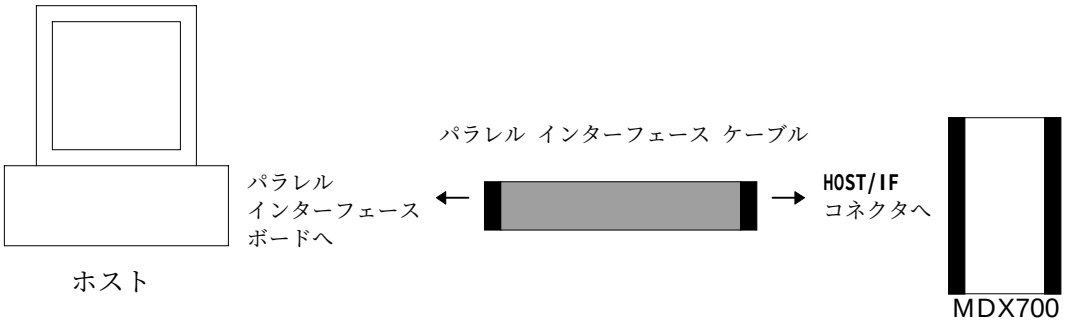


図 2-8 MDX700 とホストの接続 Parallel

2.3 MDX700 の IP アドレスの設定 Ethernet

MDX700 の IP アドレスを設定するためには、パソコン(PC/AT または PC-98)と MDX700 を RS-232C ケーブルで接続し、パソコン上で IP アドレス設定プログラム IPADDR.EXE を実行します。設定手順は、次のとおりです。

はじめに、パソコンのシリアル ポートと MDX700 の RS-232C コネクタを、RS-232C ストレート ケーブルで接続してください。PC/AT の場合は COM1 ポート、PC-98 の場合は標準の RS-232C ポートに接続してください。

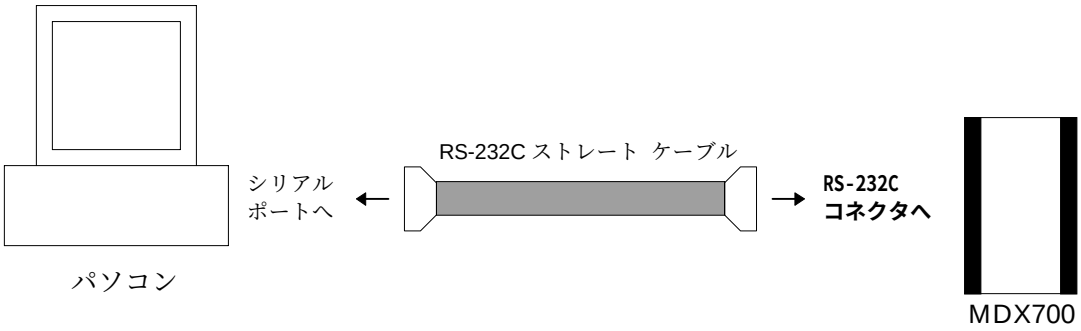


図 2-9 MDX700 の IP アドレスの設定のための接続

次に、パソコンと MDX700 の電源を入れてください。Windows95 をご使用の場合、MS-DOS モードでコンピュータを再起動してください。(EXIT コマンドで、Windows 95 に戻ることができます。)

MS-DOS が起動したら、IP アドレス設定プログラム IPADDR.EXE のフロッピー ディスクをドライブにセットし、次のコマンドを入力してください。
(フロッピー ディスク ドライブが A、設定する IP アドレスが 192.10.20.30 の場合)[*1]

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

設定した IP アドレスは、次のコマンドで確認することができます。

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

[*1] IPADDR.EXE は、MS-DOS プログラムです。IBM-PC と PC-98 の両方で動作します。

MDX700 の IP アドレスの設定が終了したら、MDX700 とパソコンの電源を切り、RS-232C ケーブルをはずしてください。

IP アドレスが正しく設定できない場合は、次のエラーメッセージが表示されます。シリアル ポートや RS-232C ケーブルの接続が間違っていないか、もう一度確認してください。

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

MDX700 の IP アドレスを設定できたら、その IP アドレスをホスト上に登録してください。
また、その際のホスト名は mdx としてください。[*1]

【重要】 具体的な作業については、ネットワーク管理者に依頼してください。

[*1] ホスト名を mdx 以外にすることもできますが、mdx とすることによって、デバッガ起動時のオプションを簡略化することができます。

2.4 MDX700 とホストの接続 Ethernet

MDX700 とホストは、ネットワークで接続します。接続手順は、次のとおりです。

MDX700 の 10BASE-T コネクタと、ホストとネットワーク接続している HUB を、10BASE-T ケーブルで接続してください。

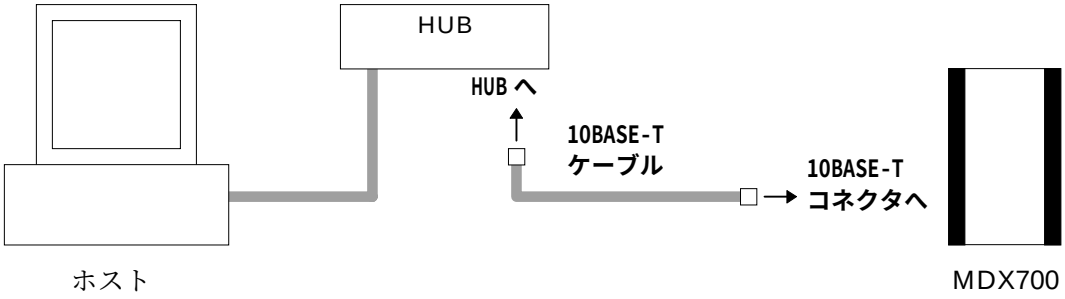


図 2-10-1 MDX700 とホストの接続 1 Ethernet

10BASE-T クロス ケーブルで接続する場合は、MDX700 とホストを直接接続してください。ただし、この場合は、他のネットワーク機器を接続することはできません。

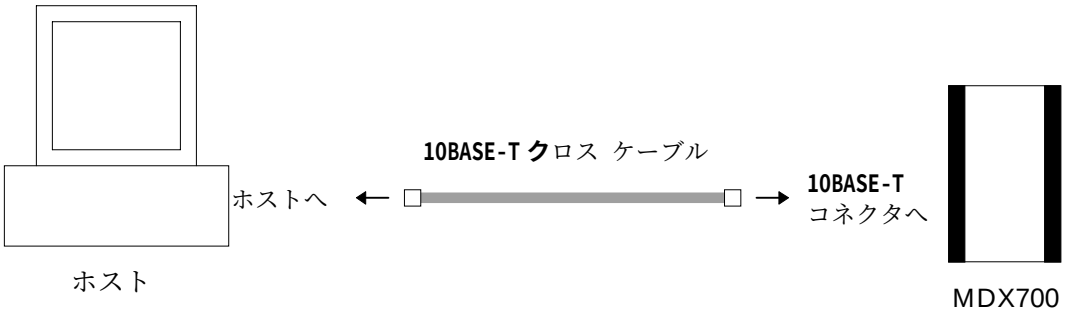


図 2-10-2 MDX700 とホストの接続 2 Ethernet

2.5 MDX700 と MDX003(オプション)の接続

MDX700 と電圧変換アダプタ MDX003 は、次のように接続してください。

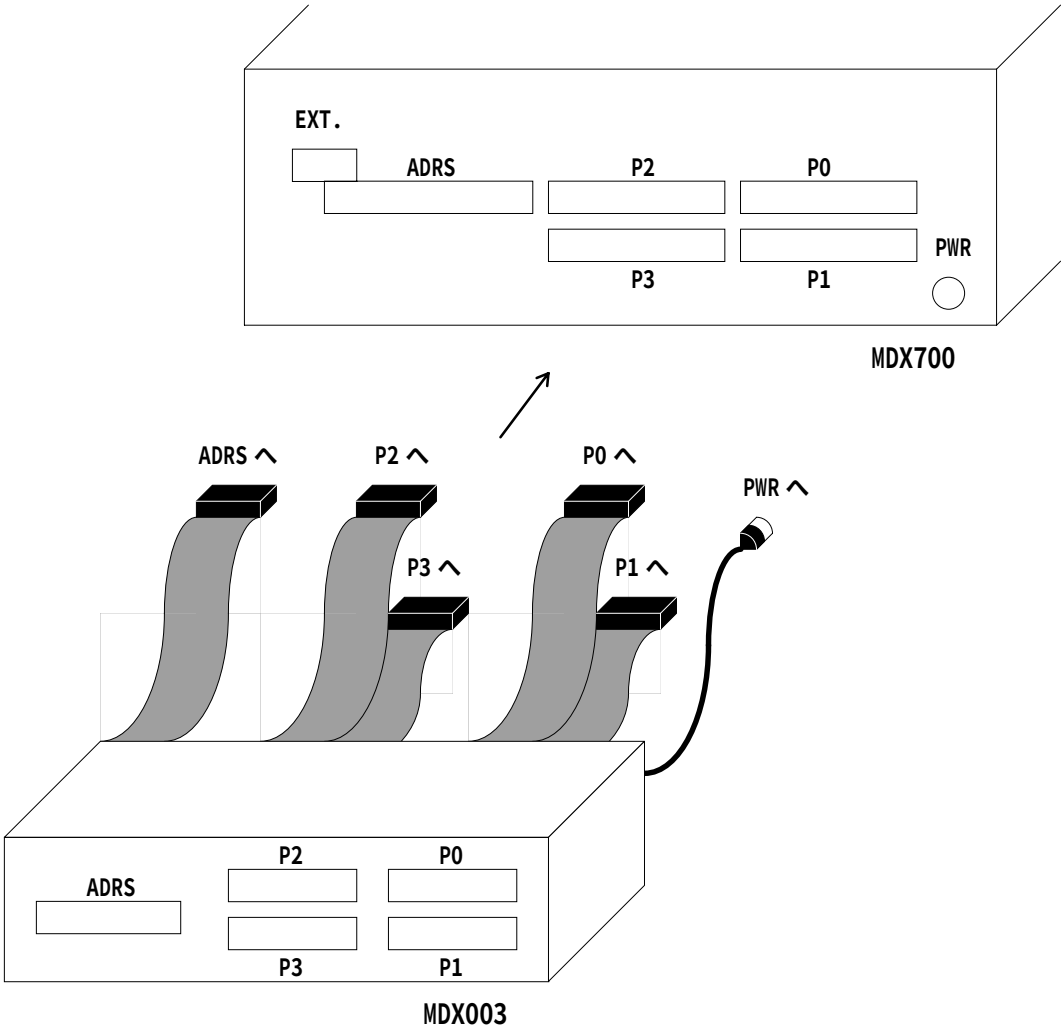


図 2-11 MDX700 と MDX003 の接続

2.6 MDX700 とターゲット システムの接続

MDX700(または MDX003)とターゲット システムは、ROM ケーブルと ROM プローブで接続します。接続手順は、次のとおりです。

はじめに、すべての ROM ケーブルに ROM プローブを装着してください。

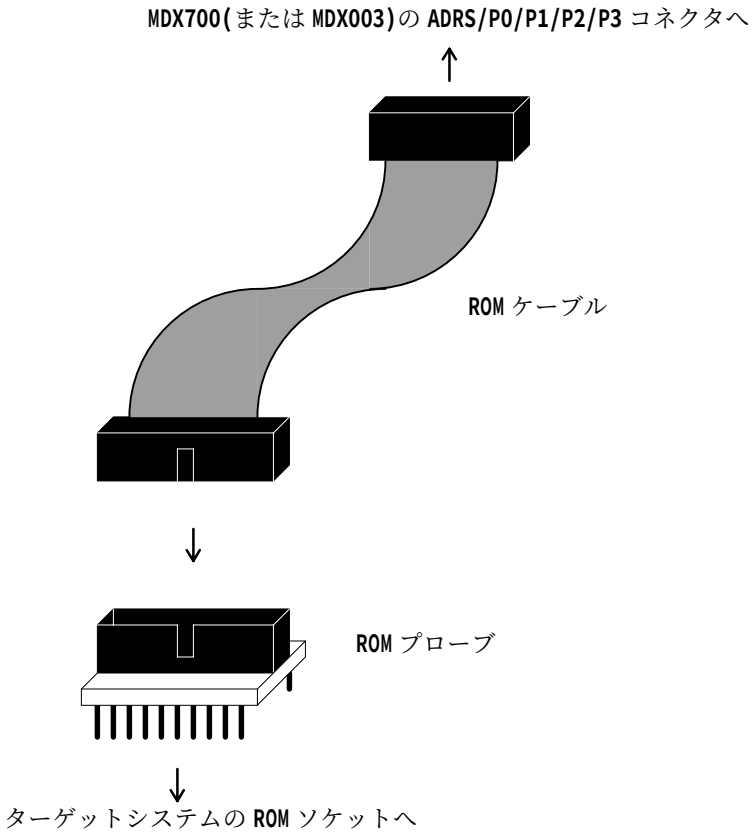


図 2-12 ROM ケーブルと ROM プローブの接続

つぎに、ROM プローブをターゲット システムの ROM ソケットへ接続し、もう一方の ROM ケーブルを MDX700(または MDX003)の ADRS/P0/P1/P2/P3 コネクタへ接続します。接続図(図 2-13-1～18)の中から、ターゲット システムと同じ環境を選び、図にしたがって接続してください。

【注意】MDX700 およびターゲット システムの電源を切ってから行なってください。

【注意】ROM プローブを逆差ししないでください。

【注意】接続図では MDX003 は省略されています。

【注意】PowerPC の場合、接続図の CPU のデータ バスのビットの昇順を逆にしてお読みください。PowerPC では MSB が bit0 となっていますが、接続図では LSB が bit0 となっています。

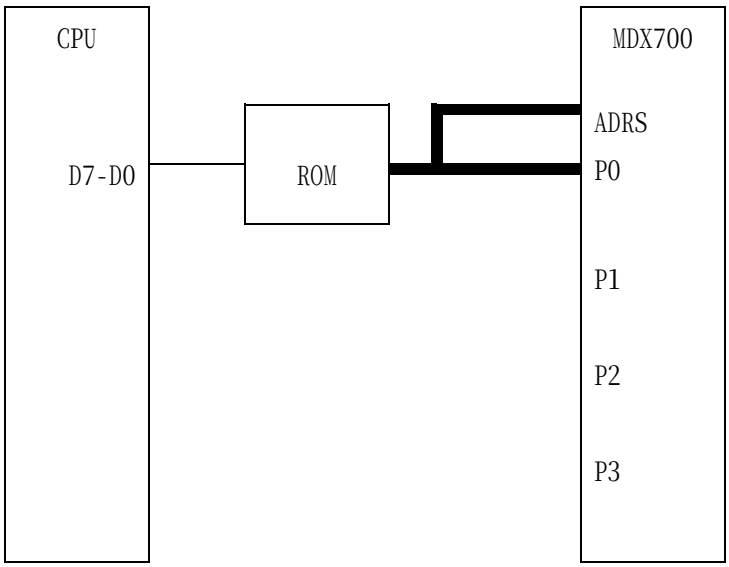


図 2-13-1 MDX700 とターゲット システムの接続 1

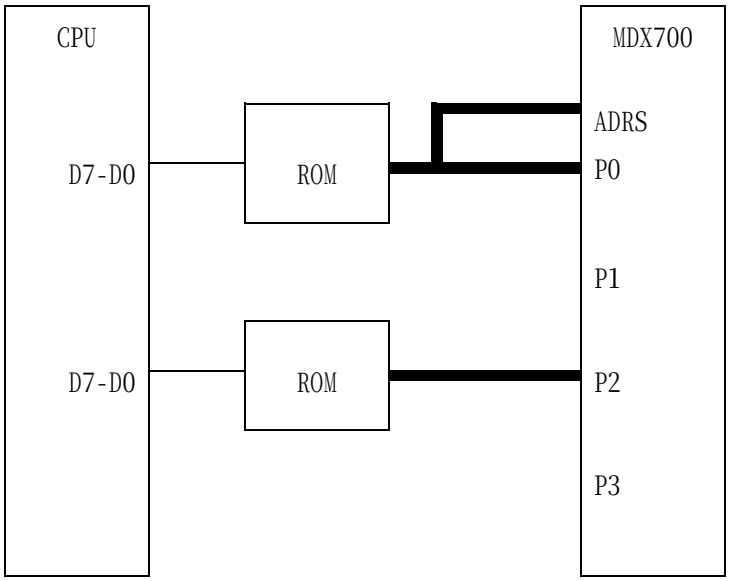


図 2-13-2 MDX700 とターゲット システムの接続 2

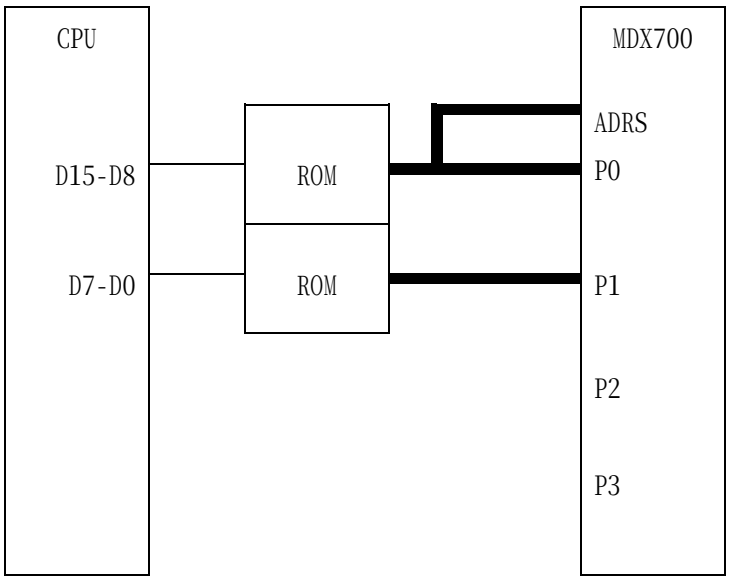


図 2-13-3 MDX700 とターゲット システムの接続 3

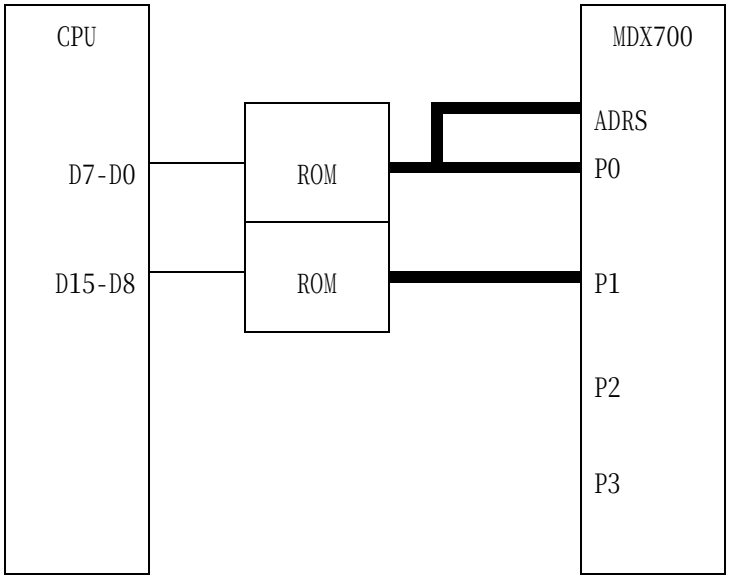


図 2-13-4 MDX700 とターゲット システムの接続 4

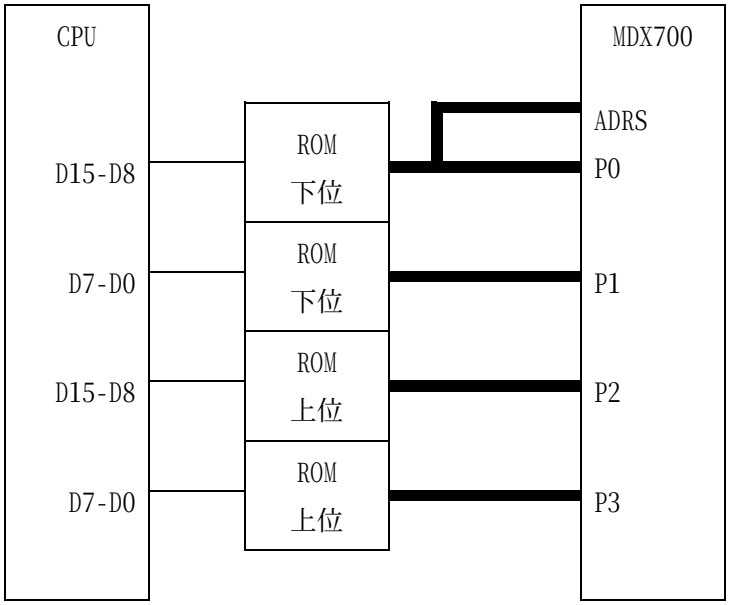


図 2-13-5 MDX700 とターゲット システムの接続 5

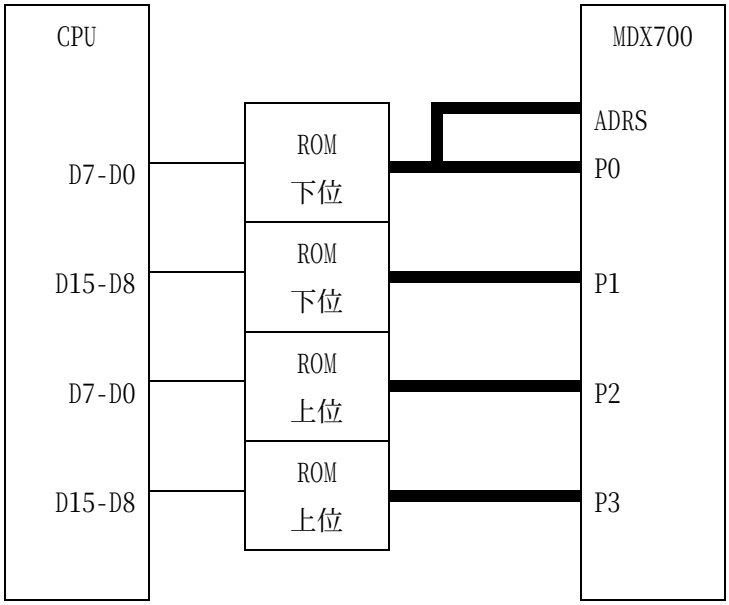


図 2-13-6 MDX700 とターゲット システムの接続 6

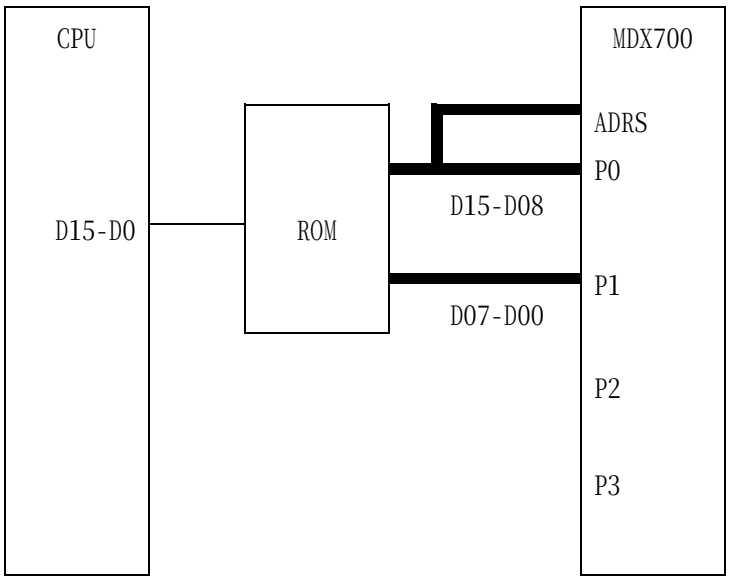


図 2-13-7 MDX700 とターゲット システムの接続 7

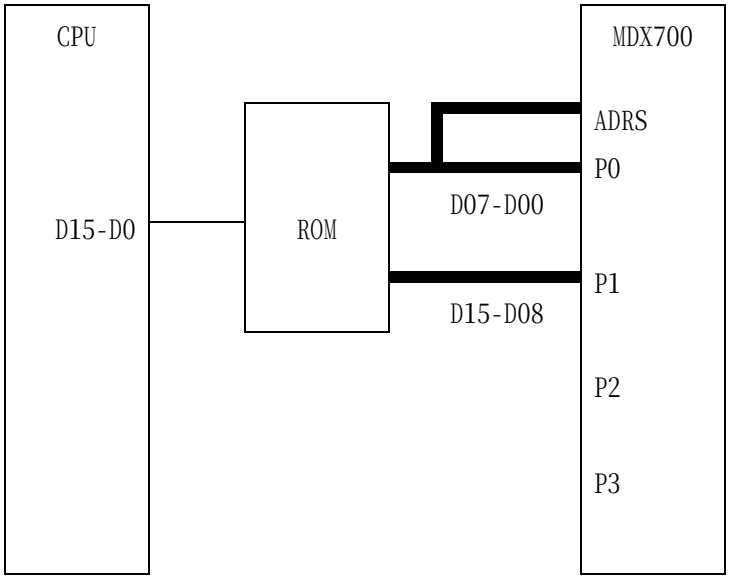
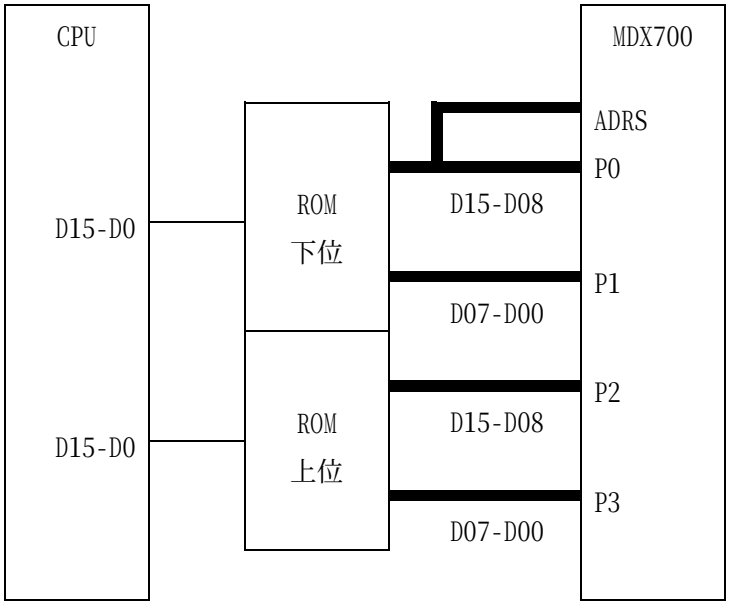
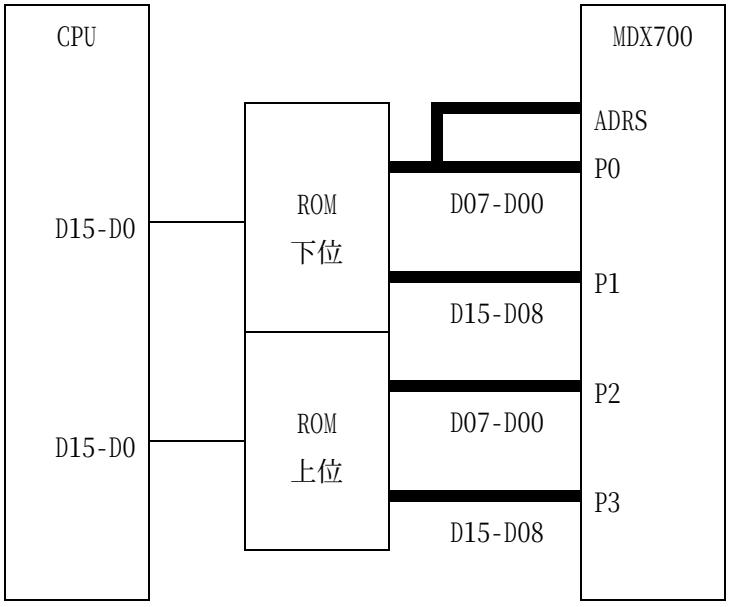


図 2-13-8 MDX700 とターゲット システムの接続 8



ROM 読み込みデータ バス 16 幅ビット
ROM データ バス 16 幅ビット
ROM2 個
CPU ビッグ エンディアン

図 2-13-9 MDX700 とターゲット システムの接続 9



ROM 読み込みデータ バス 16 幅ビット
ROM データ バス 16 幅ビット
ROM2 個
CPU リトル エンディアン

図 2-13-10 MDX700 とターゲット システムの接続 10

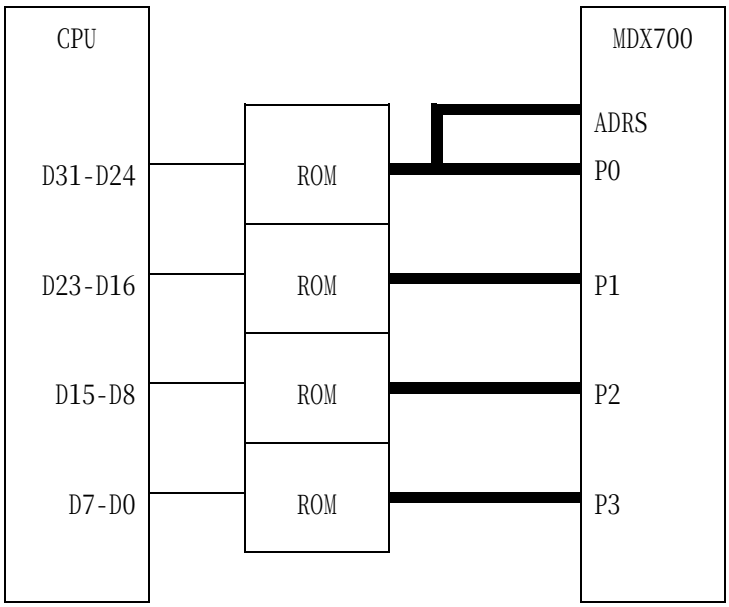


図 2-13-11 MDX700 とターゲット システムの接続 11

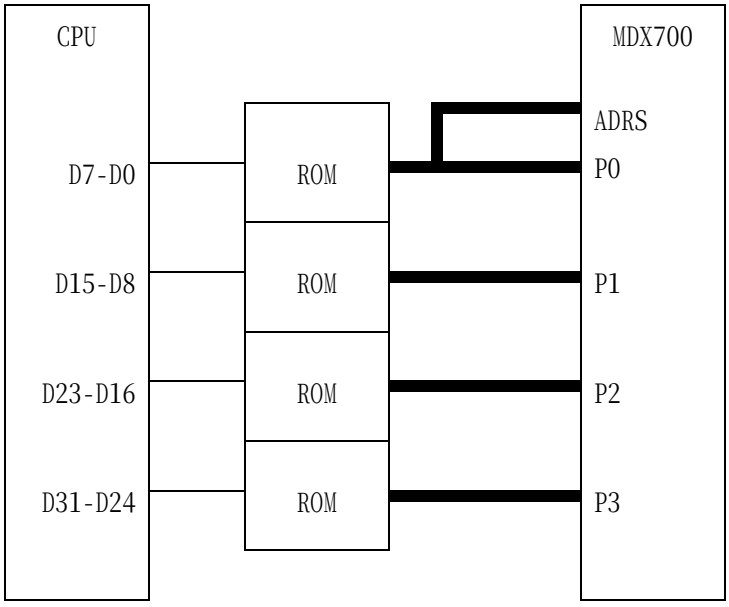
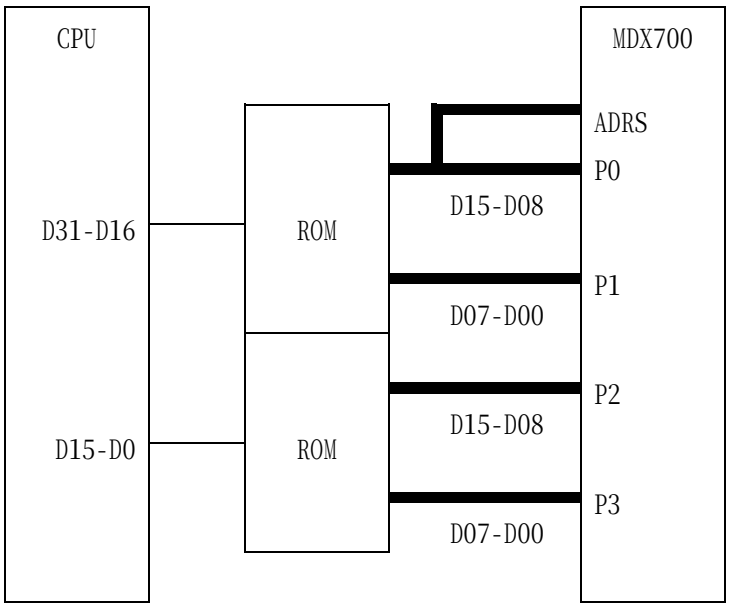
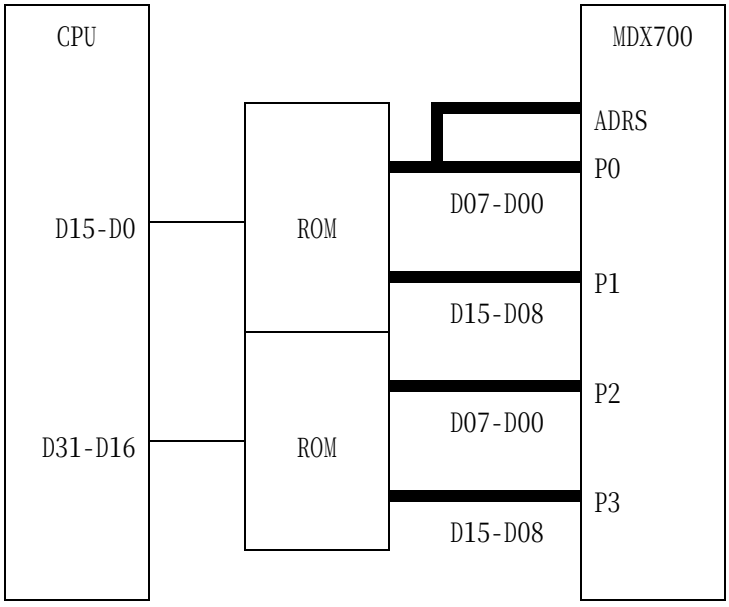


図 2-13-12 MDX700 とターゲット システムの接続 12



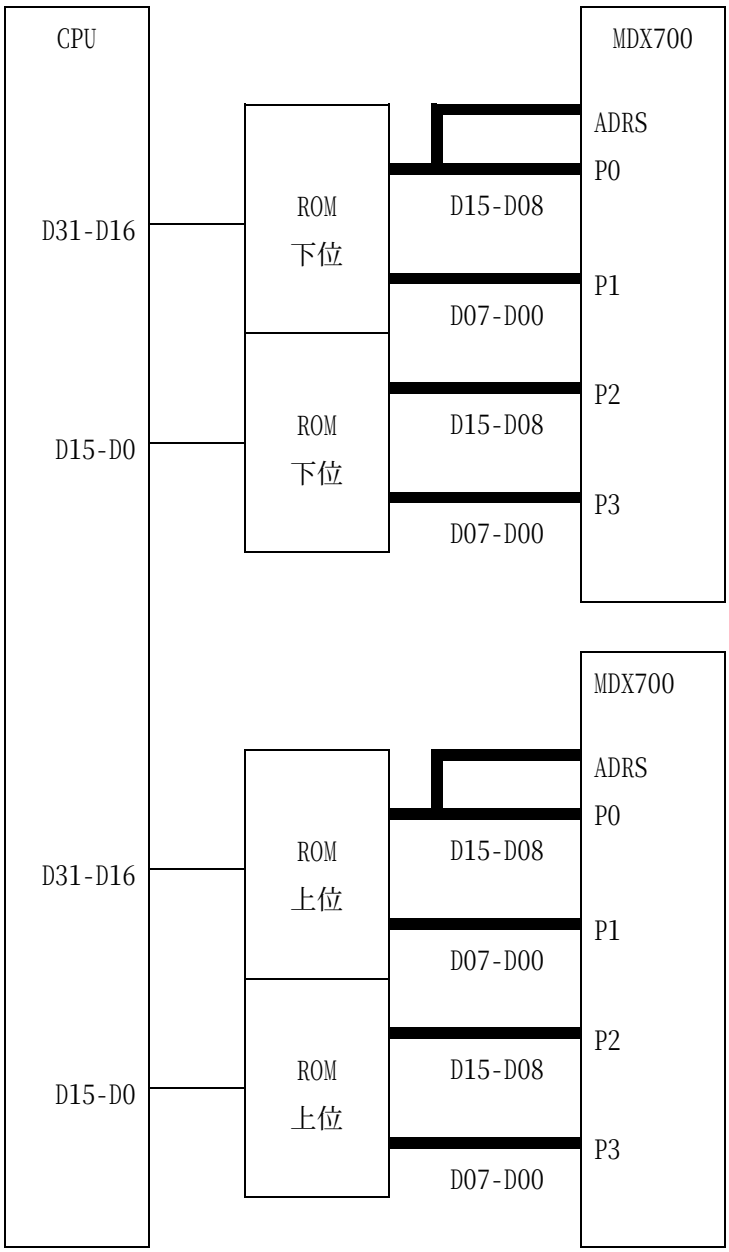
ROM 読み込みデータ バス 32 幅ビット
ROM データ バス 16 幅ビット
ROM2 個
CPU ビッグ エンディアン

図 2-13-13 MDX700 とターゲット システムの接続 13



ROM 読み込みデータ バス幅 32 ビット
ROM データ バス幅 16 ビット
ROM2 個
CPU リトル エンディアン

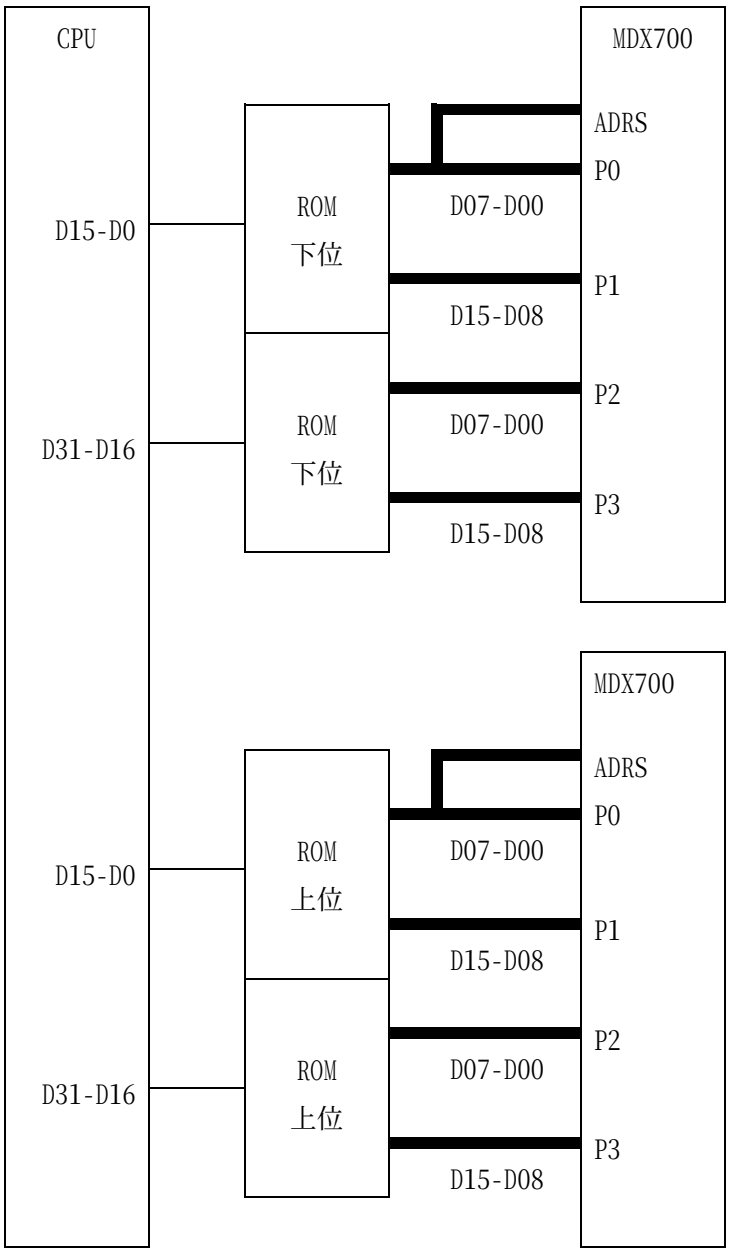
図 2-13-14 MDX700 とターゲット システムの接続 14



Parallel

ROM 読み込みデータ バス幅 32 ビット
ROM データ バス幅 16 ビット
ROM4 個
CPU ビッグ エンディアン

図 2-13-15 MDX700 とターゲット システムの接続 15



Parallel

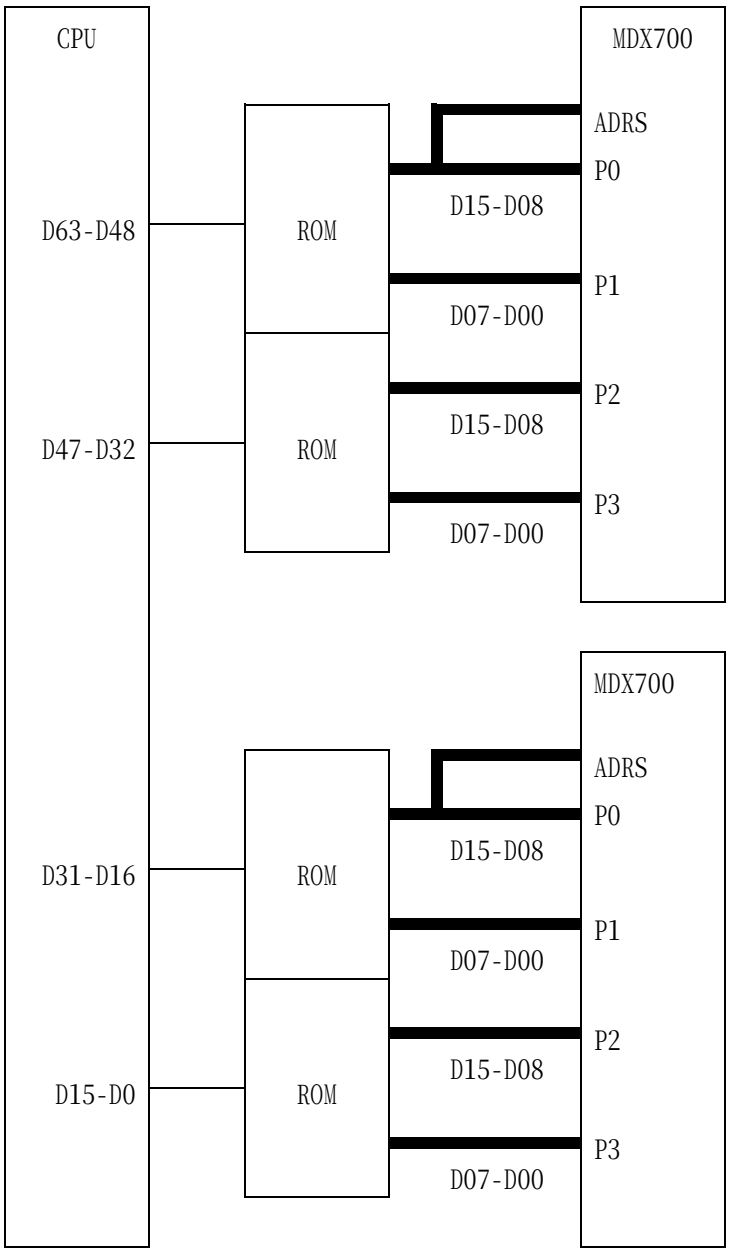
ROM 読み込みデータ バス幅 32 ビット

ROM データ バス幅 16 ビット

ROM4 個

CPU リトル エンディアン

図 2-13-16 MDX700 とターゲット システムの接続 16



Parallel

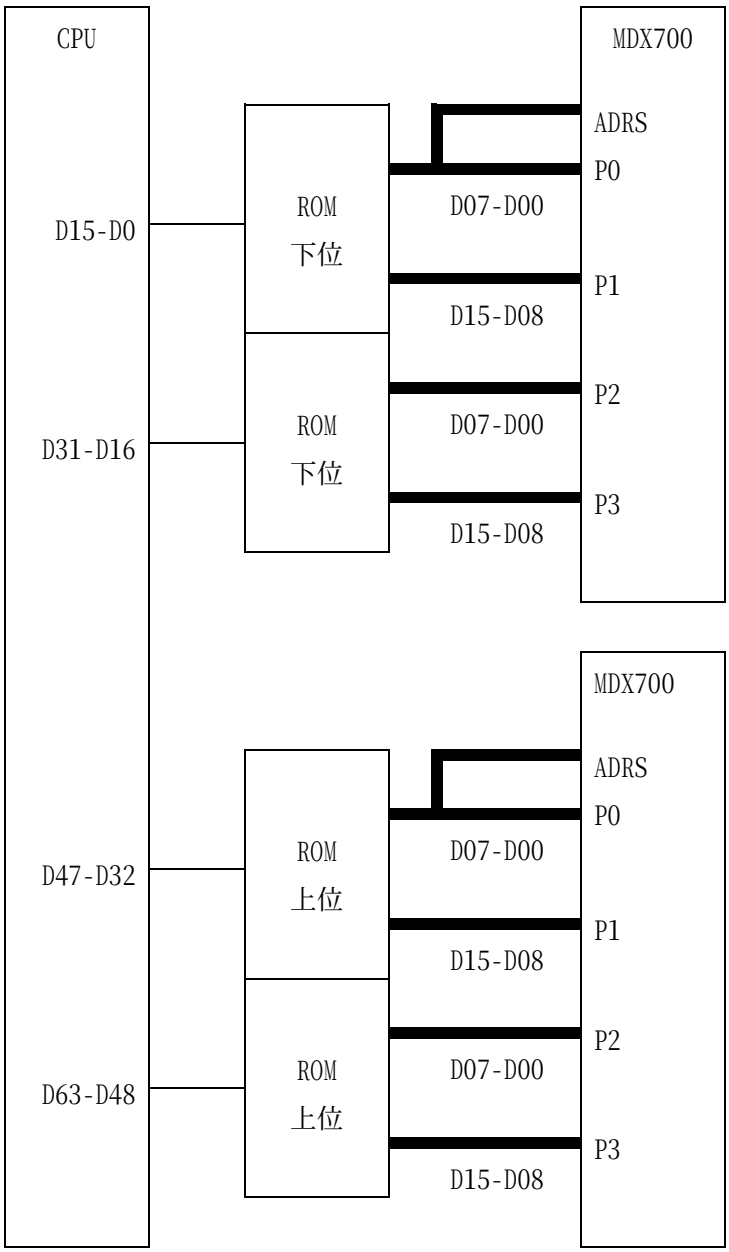
ROM 読み込みデータ バス幅 64 ビット

ROM データ バス幅 16 ビット

ROM4 個

CPU ビッグ エンディアン

図 2-13-17 MDX700 とターゲット システムの接続 17



Parallel

ROM 読み込みデータ バス幅 64 ビット
ROM データ バス幅 16 ビット
ROM4 個
CPU リトル エンディアン

図 2-13-18 MDX700 とターゲット システムの接続 18

2.7 外部トリガ ケーブルの接続

外部トリガ ケーブルは、MDX700 から出力される、RESET 信号と NMI 信号をターゲット システムへ接続するためのケーブルです。次のように、MDX700 の EXT.コネクタと、ターゲット システムの RESET/NMI 入力回路へ接続してください。

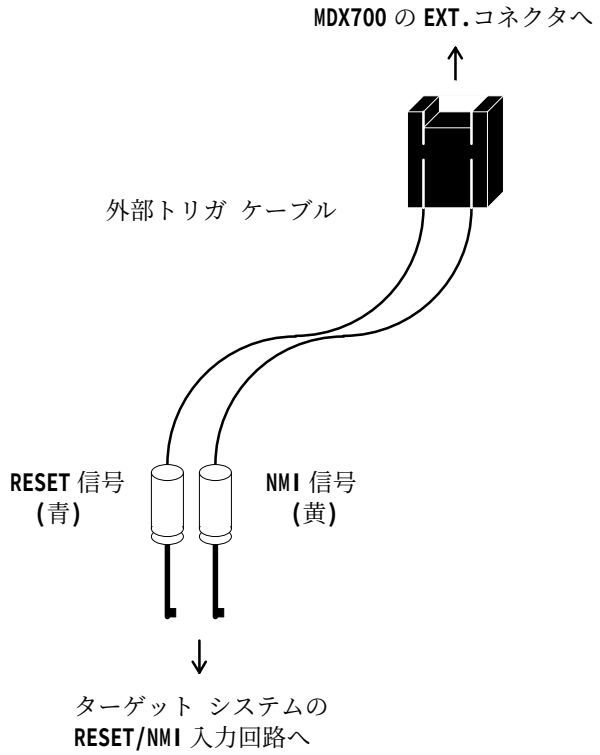


図 2-14 外部トリガ ケーブルの接続

外部トリガ ケーブルの接続は任意ですが、接続することによりデバッガの操作性を向上させることができます。

RESET 接続　　デバッガを起動した後、すぐデバッガが使用できる状態になります。[*1]

RESET 未接続　　デバッガを起動した後、すぐデバッガが使用できる状態になりません。使用できる状態にするためには、ターゲット システムの RESET スイッチを押すなどの手動操作が必要です。(電源投入時)

[*1] RESET 信号を接続する場合は、後述のコンフィグレーション ファイルの中の RESETVECTOR を設定してください。

NMI 接続　　ユーザ プログラムを実行中、デバッガのコマンドによる強制ブレークができます。*2

NMI 未接続　　ユーザ プログラムを実行中、デバッガのコマンドによる強制ブレークができません。強制ブレークさせるためは、ターゲット システムの ABORT スイッチを押すなどの手動操作が必要です。

RESET 信号と NMI 信号は、負論理オープン コレクタ出力(7406 相当)です。接続するターゲットシステムの信号は、プルアップしてある必要があります。MDX700 では、次のようなターゲットシステムの回路へ接続することを想定しています。

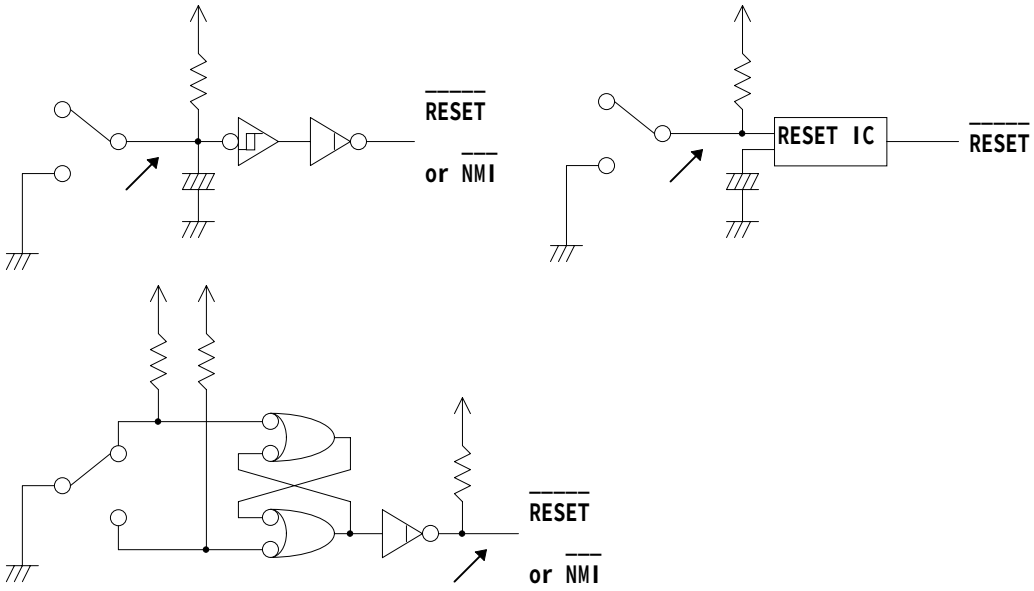


図 2-15 外部トリガ ケーブルが接続できるターゲット システムの回路

*2 PowerPC でかつ NMI 信号を接続する場合は、後述のコンフィグレーション ファイルの中の ABORTVECTOR を設定してください。

反対に、次のようなターゲット システムの回路へは接続できません。

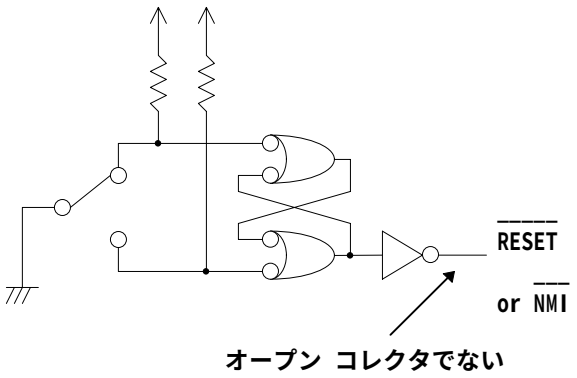


図 2-16 外部トリガ ケーブルが接続できないターゲット システムの回路

2.8 電源投入手順

MDX700 とホスト、MDX700 とターゲット システムの接続が終了したら、MDX700 の電源ケーブルを接続してください。

機器の電源投入は、次の手順で行なってください。

1. ホスト
2. MDX700
3. ターゲット システム

機器の電源切断は、次の手順で行なってください。

1. ターゲット システム
2. MDX700
3. ホスト

【重要】 電源投入および切断の手順を間違えると、機器が破壊される場合があります。

【重要】 電源投入時には、機器の接続および着脱をしないでください。

第三章 デバッガのインストール

デバッガのインストール方法について記述しています。

デバッガによってインストール方法が異なります。ご使用のデバッガの項を参照してください。また、合わせてデバッガのリリース ノートも参照してください。

MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 のインストール

1. コンパイラおよび MULTI をインストールします。「Green Hills Software のインストール手順」を参照してください。

2. MDXSERV のフロッピー ディスクをドライブにセットします。

3. エクスプローラなどを使って、フロッピーディスクの内容を、MULTI がインストールされたディレクトリへコピーします。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。 (フロッピーディスク ドライブが A、MULTI がインストールされたディレクトリが C:\GREEN の場合)

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

4. エクスプローラなどを使って、拡張子が.cfg のファイルをコピーして、mdx.cfg ファイルを作成します。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。

```
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
```

MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1 のインストール

「MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 のインストール」と同じです。

MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris のインストール

1. コンパイラおよび MULTI をインストールします。「Green Hills Software のインストール手順」を参照してください。

2. MDXSERV のフロッピー ディスクをドライブにセットします。

3. フロッピー ディスクの内容を、MULTI がインストールされたディレクトリへコピーします。次のコマンドを入力します。(フロッピー ディスク ドライブが/dev/rmt/0、MULTI がインストールされたディレクトリが/home/green の場合)

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

4. 拡張子が.cfg のファイルをコピーして、mdx.cfg ファイルを作成します。次のコマンドを入力します。

```
% 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
```

SingleStep 6.5 68K on Windows 3.1 のインストール

1. ラベルに AIC と書いてある SingleStep 68K 6.5(1/2)のフロッピー ディスクをドライブにセットします。

2. ファイル マネージャなどを使って、フロッピー ディスクの INSTALL.EXE プログラムを実行します。以降、画面の指示にしたがって作業します。

3. SingleStep 6.5 for MDX700/68K のフロッピー ディスクをドライブにセットします。

4. ファイル マネージャなどを使って、フロッピー ディスクの内容を、SingleStep がインストールされたディレクトリへコピーします。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。 (フロッピー ディスク ドライブが A、SingleStep がインストールされたディレクトリが C:\SDS65 の場合)

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

5. MDX68KP.EXE[*1] を実行します。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。 (SingleStep がインストールされたディレクトリが C:\SDS65 の場合)

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

6. プログラム マネージャを使って、MDX68K.EXE をアイコンに登録してください。

[*1] MDX68KP.EXE は、自己解凍型のプログラムです。実行後、MDX68K.EXE が作成されます。

SingleStep 6.5 PowerPC on Windows 3.1 のインストール

1. ラベルに AIC と書いてある SingleStep PowerPC 6.5(1/2)のフロッピー ディスクをドライブにセットします。

2. ファイル マネージャなどを使って、フロッピー ディスクの INSTALL.EXE プログラムを実行します。以降、画面の指示にしたがって作業します。

3. SingleStep 6.5 for MDX700/PowerPC のフロッピー ディスクをドライブにセットします。

4. ファイル マネージャなどを使って、フロッピー ディスクの内容を、SingleStep がインストールされたディレクトリへコピーします。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。 (フロッピー ディスク ドライブが A、SingleStep がインストールされたディレクトリが C:\SDS65 の場合)

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

5. MDXPPCP.EXE[*1] を実行します。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。 (SingleStep がインストールされたディレクトリが C:\SDS65 の場合)

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

6. プログラム マネージャを使って、MDXPPC.EXE をアイコンに登録してください。

[*1] MDXPPCP.EXE は、自己解凍型のプログラムです。実行後、MDXPPC.EXE が作成されます。

XHI68KMD(XRAY68K 2.2a) on MS-DOS PC/AT のインストール

1. XHI68KMD のフロッピー ディスクをドライブにセットします。

2. 次のコマンドを入力します。(フロッピー ディスク ドライブが A、ハード ディスク ドライブが C の場合)

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

3. 環境変数 PATH に、c:\xhi68kmd を追加します。

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

XHI68KMD(XRAY68K 2.2a) on MS-DOS PC-98 のインストール

1. XHI68KMD のフロッピー ディスクをドライブにセットします。

2. 次のコマンドを入力します。(フロッピー ディスク ドライブが C、ハード ディスク ドライブが A の場合)

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

3. 環境変数 PATH に、a:\xhi68kmd を追加します。

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

4. 環境変数 DOS16M を設定します。

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

XHI68KMD(XRAY68K 3.4) on SunOS/Solaris のインストール

1. XHI68KMD のカートリッジ テープをドライブにセットします。

2. 次のコマンドを入力します。 (カートリッジ テープ ドライブが/dev/rmt/0 の場合)

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

3. MasterWorks がすでにインストールされている場合は、次のコマンドを入力します。

```
% 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. MasterWorks がインストールされていない場合は、次のコマンドを入力します。

```
% 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
```

MDXDEB 3.5 on MS-DOS/Windows 3.1/95 のインストール

1. MDXDEB のフロッピー ディスクをドライブにセットします。

2. エクスプローラなどを使って、任意のディレクトリを作成しフロッピーディスクの内容を、作成したディレクトリへコピーします。MS-DOS 互換ウィンドウで操作する場合は、次のコマンドを入力します。(フロッピーディスクドライブが A、コピーするディレクトリが C:\MDXDEB)

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

3. MS-DOS で使用する場合は、環境変数 PATH に、c:\mdxdeb を追加します。

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

4. Windows で使用する場合は、エクスプローラなどを使って、MDXDEBW.EXE のショートカットをデスクトップなどに作成してください。その際、コマンド行の引数にコンフィグ　レーション ファイルを指定してください。

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

MDXDEB 3.5 on SunOS/Solaris のインストール

1. MDXDEB のフロッピー ディスクをドライブにセットします。

2. 任意のディレクトリを作成し、フロッピー ディスクの内容を、作成したディレクトリへコピーします。次のコマンドを入力します。 (フロッピーディスク ドライブが/dev/rfd0a、コピーするディレクトリが/home/mdxdeb の場合)

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

3. 環境変数 PATH に、/home/mdxdeb を追加します。

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

第四章 デバッガの環境設定

デバッガを使用する前に必要な、環境設定の方法について記述しています。主に、コンフィグレーション ファイル(mdx.cfg)の変更方法について記述しています。

【重要】環境設定は、慎重に行なってください。環境設定が間違っていると、デバッガを使用することができません。

4.1 コンフィグレーション ファイルによるデバッガの環境設定

デバッガの環境設定は、コンフィグレーション ファイルで行ないます。

コンフィグレーション ファイルは、MDX700 やターゲット システムの環境を指定するためのファイルです。デバッガを使用する前には、ターゲット システムごとにコンフィグレーションファイルを作成しなければなりません。

デバッガには、次のようなコンフィグレーション ファイルのサンプル(ファイル名、mdx.cfg)が添付されています。

```
*
*  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
```

図 4-1 コンフィグレーション ファイルのサンプル(mdx.cfg)*1

通常は、mdx.cfg ファイルを変更して、ターゲット システム専用のコンフィグレーション ファイルを作成してください。*2

コンフィグレーション ファイルのシンタックス

1. 一行ごとにひとつの項目を指定します。「BUS 16 ;bus width」の行は、項目 BUS を 16 に指定しています。また、「;」以降は、コメントです。
2. 項目とその指定値の間は、スペースまたはタブで区切ります。
3. 16 進数を指定する場合は、数値の前に 0x を付加します。
4. コメント行は、行頭を「*」で始めます。「*」の代りに「#」や「;」を使用することもできます。

*1 図 4-1 は、SH のサンプルです。CPU によって、多少内容が異なります。

*2 コンフィグレーション ファイルは、テキスト ファイルです。Windows 環境では「メモ帳」、UNIX 環境では、vi などのエディタで変更してください。

【注意】コンフィグレーションファイルを変更するときは、コメント行以外の行を削除しないでください。

【注意】 コンフィグレーション ファイルを変更するときは、項目を変更しないでください。

コンフィグレーション ファイルの項目の詳細については、「4.2 コンフィグレーション ファイルの項目」を参照してください。

なお、コンフィグレーション ファイルを変更するにあたっては、次の情報が必要になります。

- パラレル インターフェース ボードの I/O アドレス Parallel
- CPU 名
- ROM 読み込みデータ バス幅
- ROM の先頭アドレスと容量
- モニタ プログラムに開放できる ROM 領域[*3]
- モニタ プログラムに開放できる RAM 領域

[*3] モニタプログラムに開放できる領域とは、ユーザ プログラムが使用しない領域のことです。

4.2 コンフィグレーション ファイルの項目

MONITOR **モニタ プログラムのファイル名** —— ターゲットシステムの CPU にしたがって、ファイル名を指定してください。

mdx68k.abs	68K
mdxarm.abs	ARM ビッグ エンディアン
mdxarml.abs	ARM リトル エンディアン
mdxmips.abs	MIPS ビッグ エンディアン
mdxmipsl.abs	MIPS リトル エンディアン
mdxppc.abs	PowerPC
mdxsh.abs	SH ビッグ エンディアン
mdxshl.abs	SH リトル エンディアン
mdxv850.abs	V850 V850E
mdxv800.abs	V850 V850E 以外の V800

CPU **CPU 名** —— ターゲット システムの CPU 名を指定してください。CPU 名については、「付録 F 対応 CPU」を参照してください。

PORT **パラレル インターフェース ボードの I/O アドレス** Parallel —— 使用しているホストにしたがって、I/O アドレスを指定してください。[*1]

0x0100	PC/AT
0x01D0	PC-98

また、ホストに関係なく、パラレル インターフェース ボードの I/O アドレスを変更した場合は、変更後の I/O アドレスを指定してください。

図 2-13-15～18 のように、二台の MDX700 をターゲットシステムへ接続している場合は、二つの I/O アドレスを 32 ビット形式で指定します。アドレスまたはデータ バスの下位側に接続している MDX700 の I/O アドレスを下位 16 ビットに、上位側の MDX700 の I/O アドレスを上位 16 ビットに指定してください。(例)上位側の I/O アドレスが 0x0120、下位側の I/O アドレスが 0x0100 の場合、0x01200100 を指定してください。

[*1] Ethernet の場合、PORT は無視されます。

BUS **ROM 読み込みデータ バス幅** —— ターゲットシステムの ROM が読み込まれるときのデータ バス幅(8、16、32、または 64)を指定してください。通常は、CPU データ バス幅を指定します。

【注意】ROM のバス サイジングをしているターゲット システムの場合は、CPU データ バス幅を指定しないでください。

(例)8 ビットのデータバス幅で ROM を 4 回読み込み、32 ビットデータにしてから CPU に読み込ませるターゲットシステムでは、8 を指定してください。

ROM **ROM の先頭アドレス** —— ターゲットシステムの ROM の先頭アドレスを指定してください。*2

複数の ROM を実装しているターゲットシステムの場合は、一番下位アドレスにマッピングされている ROM の先頭アドレスを指定してください。(例)0x00100000 番地から始まる ROM と、0x00180000 番地から始まる ROM が実装されているターゲットシステムの場合は、0x00100000 を指定してください。

【注意】キャッシュ領域のアドレスを指定しないでください。 MIPS V800

ROMSIZE **ROM の容量(バイト)** —— ターゲットシステムの ROM の容量をバイトで指定してください。*3

複数の ROM を実装しているターゲットシステムの場合は、ROM の総容量を指定してください。(例)512K バイトの ROM を二個実装しているターゲットシステムの場合は、0x00100000 を指定してください。

図 2-13-15～16 のように、ROM 読み込みデータ バス幅が 32 ビットで二台の MDX700 をターゲットシステムへ接続している場合は、ROM の総容量に 0x20000000 を加えた値を指定してください。(例)ROM の総容量が 0x00400000 の場合、0x20400000 を指定してください。

WORKROM **モニタ プログラム ROM 先頭アドレス** —— モニタ プログラムに開放する ROM の先頭アドレスを指定してください。

---

*2 MDX700 を接続していない ROM 領域は含めないでください。

*3 ROM の容量については、「付録 D 対応 ROM」を参照してください。

ROM の最後の領域をモニタ プログラムに開放する場合は、ROM の最終アドレス+1 から WORKROMSIZE を引いた値を指定します。(例)ROM の最終アドレスが 0x03FFFFFF、WORKROMSIZE が 0x00004000 の場合は、0x03FFC000 を指定してください。

【注意】ROM 領域以外のアドレスを指定しないでください。

WORKROMSIZE **モニタ プログラム ROM 容量(バイト)** —— モニタ プログラムに開放する ROM の容量をバイトで指定してください。通常は、コンフィグレーション ファイルのサンプルの値をそのまま使用してください。

WORKRAM **モニタ プログラム RAM 先頭アドレス** —— モニタ プログラムに開放する RAM の先頭アドレスを指定してください。できるだけ、初期化を必要としない RAM のアドレスを指定してください。*4

WORKRAMSIZE **モニタ プログラム RAM 容量(バイト)** ——モニタ プログラムに開放する RAM の容量をバイトで指定してください。通常は、コンフィグレーションファイルのサンプルの値をそのまま使用してください。

RESETVECTOR **リセット例外ベクタのアドレス** —— ターゲット システムをリセットした直後に読み込まれる、リセット例外ベクタのアドレスを指定してください。通常は、ROM と同じ値を指定してください。

V800 の場合は、ROM の最終アドレスから 0x0000000F を引いた値を指定してください。

リセット例外ベクタが ROM 領域外にある場合は、0xFFFFFFFF を指定してください。(ターゲットシステム上で、別のモニタプログラムが動作している場合など)

【注意】ROM 領域以外のアドレスを指定しないでください。

ABORTVECTOR **強制ブレーク例外ベクタのアドレス** ARM PowerPC —— ユーザプログラ

*4 初期化を必要とする RAM を指定した場合は、モニタ プログラムを変更する必要があります。

ムを強制ブレークさせるための、例外ベクタのアドレスを指定ください。[*5]

TIMER **ソフトウェア タイマ値** Parallel —— デバッガが使用するソフトウェアタイマの値を指定してください。通常は、コンフィグレーション ファイルのサンプルの値をそのまま使用してください。[*6]

デバッガの起動時や、メモリ アクセス時に「communication port timeout」エラーが起こる場合のみ、値を大きくしてください。

REG_xxx **レジスタの初期値（省略可）**——ユーザ プログラムのレジスタの初期値を指定してください。xxx はレジスタ名です。レジスタ名については、「付録 G レジスタ名一覧」を参照してください[*7]

最低限、以下のレジスタの初期値を指定してください。その他のレジスタの初期値は、コンフィグレーション ファイルのサンプルの値をそのまま使用してください。

- スタック ポインタの初期値を、RAM のアドレスに指定してください。
- VBR レジスタの初期値を、例外ベクタの先頭アドレスに指定してください。 68000 SH
- 例外ベクタのアドレスが 0x0000_0000 の場合は、MSR レジスタの初期値を、0x2000 に指定してください。また、0xFFF0_0000 の場合は、0x2040 に指定してください。 PowerPC 403 以外
- EVPR レジスタの初期値を、例外ベクタの先頭アドレスに指定してください。 PowerPC 403

[*5] 68000 の場合は、レベル 7 割り込みを強制ブレークに使用しています。
MIPS SH V800 の場合は、NMI を強制ブレークに使用しています。

[*6] Ethernet の場合、TIMER は無視されます。

[*7] モニタ プログラムのレジスタの初期値を指定することはできません。

4.3 モニタ プログラムの変更

ほとんどの場合デバッガの環境設定は、コンフィグレーションファイルの変更だけですみます。しかし、ターゲットシステムの条件によっては、さらに、モニタ プログラムの変更も必要な場合があります。

モニタ プログラムの変更が必要なターゲット システムの条件は、次のとおりです。

- DRAM コントローラなどを初期化しないと、RAM[*1] へのアクセスができない。
- CPU の内部レジスタなどを初期化しないと、RAM へのアクセスができない。

上記のようなターゲット システムの場合は、次の手順で、RAM アクセスするための初期化コードを、モニタ プログラムに追加してください。

1. モニタ プログラムのソース ファイルをエディタで開きます。[*2]
2. モニタ プログラムのソース ファイルの最後にある、USER_INIT ラベルの位置に、アセンブリ言語で初期化コードを入力します。
3. エディタを終了します。
4. モニタ プログラムのソース ファイルをアセンブルし、実行ファイルを作成します。[*3]

【注意】コンフィグレーション ファイルの WORKROM の指定によって、モニタ プログラムがロードされるアドレスは動的に変わります。したがって、初期化コードもモニタ プログラム同様、再配置可能なコード(ポジション インデペンデント コード)で記述する必要があります。

*1 厳密にいえば、WORKRAM と WORKRAMSIZE で指定した RAM 領域です。

*2 モニタ プログラムのソース ファイルのファイル名については、デバッガのリリース ノートを参照してください。

*3 デバッガによってアセンブル手順が異なります。アセンブル手順については、デバッガのリリース ノートやモニタ プログラムのソース ファイルのコメントを参照してください。

4.4 コンフィグレーション ファイルの変更による初期化コードの追加

初期化コードを追加する方法には、モニタ プログラムのソース ファイルを変更する方法と、コンフィグレーション ファイルを変更する方法の二つがあります。

モニタ プログラムを変更する方法では、初期化コードをアセンブリ言語で記述できますが、コンフィグレーション ファイルを変更する方法では、16 進数の機械語でしか初期化コードを記述できません。

コンフィグレーション ファイルを変更する方法は、手順が複雑ですが、モニタ プログラムがバージョン アップされたとき、モニタ プログラムを再度変更する必要がなくなります。

コンフィグレーション ファイルを変更して、初期化コードを追加する手順は次のとおりです。

1. アセンブラのソース ファイル(新規ファイル)をエディタで開きます。
2. アセンブリ言語で初期化コードを入力します。
3. 初期化コードの最後に、サブルーチンからのリターン命令を入力します。[*1]
4. エディタを終了します。
5. リスト ファイルを作成するオプションを指定し、ソース ファイルをアセンブルします。
6. リスト ファイルに出力された機械語を、コンフィグレーション ファイルに記述します。

コンフィグレーション ファイルに初期化コードの機械語を記述する場合は、次のように、INIT_CODE という項目を使用します。ひとつの機械語のサイズは、CPU の命令の最小サイズです。CPU のエンディアンに関係なく上位ビットから記述してください。

【注意】 コンフィグレーション ファイルに記述した初期化コードは、モニタ プログラムのソースに記述した初期化コードより優先されます。

[*1] 68000 の場合、「rts」命令ではなく「jmp (a6)」命令を使います。

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]
```

第五章 デバッガの起動

デバッガの起動方法について記述しています。

デバッガによって起動方法が異なります。ご使用のデバッガの項を参照してください。また、合わせてデバッガのマニュアルとリリース ノートも参照してください。

RESET 信号をターゲット システムに接続している場合は、そのままデバッガの起動方法を参照してください。

RESET 信号をターゲット システムに接続していない場合は、次の手順にしたがって、デバッガを起動してください。[*1]

1. デバッガを起動する。
2. タイムアウト エラーが表示される。
3. ターゲット システムのリセット スイッチを押す(または、電源を OFF→ON する)
4. デバッガを再初期化をする。

MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 の起動方法

1. MULTI 本体を起動します。
2. 次のコマンドで、サーバ プログラム MDXSERV をリモート接続します。[*2]

```
remote mdxserv
```

- デバッガを再初期化する場合は、もう一度、remote コマンドを入力してください。
- コンフィグレーション ファイルは、MDXSERV と同じディレクトリ内の mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように remote コマンドを入力してください。

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

[*1] この手順は電源投入直後にのみ必要です。

[*2] ビルダのモードになっている場合は、サーバ名に mdxserv と指定し、「リモート」ボタンを押します。

MULTI 1.8.7 + MDXSERV 3.0.5 on Windows 3.1 の起動方法

「MULTI 1.8.8 + MDXSERV 3.0.5 on Windows 95 の起動方法」と同じです。

MULTI 1.8.8 + MDXSERV 3.0.5 on SunOS/Solaris の起動方法

1. MULTI 本体を起動します。
2. 次のコマンドで、サーバ プログラム MDXSERV をリモート接続します。[*1]

```
remote mdxserv
```

- デバッガを再初期化する場合は、もう一度、remote コマンドを入力してください。
- コンフィグレーション ファイルは、MDXSERV と同じディレクトリ内の mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように remote コマンドを入力してください。

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

- mdx 以外のホスト名を指定する場合は、次のように-h オプションを使用してください。

```
remote mdxserv -h mdx1
```

[*1] ビルダのモードになっている場合は、サーバ名に mdxserv と指定し、「リモート」ボタンを押します。

SingleStep 6.5 68K on Windows 3.1 の起動方法

1. MDX68K.EXE を起動します。プログラム マネージャでアイコン登録した場合は、アイコンをダブル クリックします。

- デバッガを再初期化する場合は、デバッガを終了し、再度デバッガを起動してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように、環境変数 MDX_CFG を指定してから MDX68K.EXE を実行してください。

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

SingleStep 6.5 PowerPC on Windows 3.1 の起動方法

1. MDXPPC.EXE を起動します。プログラムマネージャでアイコン登録した場合は、アイコンをダブル クリックします。

- デバッガを再初期化する場合は、デバッガを終了し、再度デバッガを起動してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように、環境変数 MDX_CFG を指定してから MDXPPC.EXE を実行してください。

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

XHI68KMD(XRAY68K 2.2a) on MS-DOS PC/AT の起動方法

1. XHI68KMD.EXE を起動します。次のコマンドを入力します。

```
C:\>xhi68kmd
```

- デバッガを再初期化する場合は、デバッガを終了し、再度デバッガを起動してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように、環境変数 MDX_CFG を指定してから XHI68KMD.EXE を実行してください。

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

XHI68KMD(XRAY68K 2.2a) on MS-DOS PC-98 の起動方法

「XHI68KMD(XRAY68K 2.2a) on MS-DOS PC/AT の起動方法」と同じです。

XHI68KMD(XRAY68K 3.4) on SunOS/Solaris の起動方法

1. xhi68kmd を起動します。次のコマンドを入力します。

```
% xhi68kmd
```

- デバッガを再初期化する場合は、デバッガを終了し、再度デバッガを起動してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、次のように、環境変数 MDX_CFG を指定してから xhi68kmd を実行してください。

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

MDXDEB 3.5 on MS-DOS の起動方法

1. MDXDEB.EXE を実行します。次のコマンドを入力します。

```
C:\>mdxdeb
```

- デバッガを再初期化する場合は、I コマンドを入力してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、mdxdeb の後にファイル名を指定してください。

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

MDXDEB 3.5 on Windows 3.1/95 の起動方法

1. MDXDEBW.EXE のショートカットをダブル クリックしてください。

- デバッガを再初期化する場合は、I コマンドを入力してください。
- コンフィグレーションファイルは、コマンド行の引数で指定されたファイルが検索されます。

MDXDEB 3.5 on SunOS/Solaris の起動方法

1. mdxdeb を実行します。次のコマンドを入力してください。

```
% mdxdeb
```

- デバッガを再初期化する場合は、I コマンドを入力してください。
- コンフィグレーション ファイルは、環境変数 PATH で指定されたディレクトリ内にある mdx.cfg が検索されます。明示的にコンフィグレーション ファイルを指定する場合は、mdxdeb コマンドの後にファイル名を指定してください。

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

第六章 MDXDEB コマンド

簡易デバッガ MDXDEB のコマンドの使い方について記述しています。

MDXDEB 以外のデバッガは、MDXDEB コマンドの一部を、拡張コマンドとしてサポートしています。詳しくは、デバッガのリリース ノートを参照してください。

A *addr* MIPS メモリの内容をアセンブリ言語で変更します。

addr: メモリ変更開始アドレス (16 進数)

(例)

```
> A A0000000
A0000000 add r1,r2,r3
A0000004 or r4,r5,r6
A0000008          (リターンのみで終了)
```

B ブレークポイントを表示します。

(例)

```
> B
0 0000800C        (先頭の数字は、ブレークポイント番号)
1 0001E8D4        (続く数字は、ブレークポイント アドレス)
```

B *addr* ブレークポイントを設定します。命令上に、最大 64 個のブレークポイントを設定できます。

addr: ブレークポイント設定アドレス (16 進数)

(例)

```
> B 1E8D4
```

B/C {*num*|*} ブレークポイントを解除します。

num: ブレークポイント番号 (10 進数)

*: すべてのブレークポイント

(例)

```
> B/C 10                  (ブレークポイント番号 10 を解除)
> B/C *           (すべてのブレークポイントを解除)
```

C　　MDX700 のコンフィグレーションを表示します。

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

メモリの内容を表示します。*addr1* を省略すると、連続してメモリの内容を表示します。*addr2* を省略すると、64 バイト分のメモリを表示します。

/B: 8 ビット　/W: 16 ビット　/L: 32 ビット

addr1: メモリ表示開始アドレス (16 進数)

addr2: メモリ表示終了アドレス (16 進数)

(例)

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

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

メモリの内容を変更します。**=*data*** を省略すると、対話型でメモリを変更します。

/B: 8 ビット　**/W:** 16 ビット　**/L:** 32 ビット

addr: メモリ変更開始アドレス (16 進数)

data: メモリ変更データ (16 進数)

(例)

```
> E/B 1000=55
> E/W 3000=11,22,33        (data はコンマで区切って指定可能)
> E/L 2000
00002000 00000000 11223344
00002004 00000000 55667788
00002008 00000000 .        (ピリオドで終了)
```

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

メモリの内容をフィルします。

/B: 8 ビット　**/W:** 16 ビット　**/L:** 32 ビット

addr1: メモリ フィル開始アドレス (16 進数)

addr2: メモリ フィル終了アドレス (16 進数)

data: フィル データ (16 進数)

(例)

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

G [*addr*] ユーザプログラムを実行します。***addr*** を省略した場合は、現在の PC からユーザプログラムを実行します。ブレークポイントで停止しない場合は、何かキーを押してください。NMI 信号を出力し、ユーザプログラムを停止できます。[*1]

addr: ユーザ プログラム開始アドレス (16 進数)

(例)

```
> G
> G 1000
```

H ヘルプ メッセージを表示します。

I [*config*] RESET 信号を出力し、MDX700 を再初期化します。***config*** を省略した場合は、現在使用しているコンフィグレーション ファイルで再初期化します。

config: 再初期化するためのコンフィグレーション ファイル名

(例)

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

K MDX700 のエミュレーション メモリをテストします。エラーがない場合は、テスト終了後、OK を表示します。エラーがある場合は、エラーが起こったアドレス、データ、期待値のデータを表示します。

L *file*[,*offset*] MDX バイナリ ファイル、S レコード ファイル、インテルヘキサ ファイル、または COFF ファイルをメモリへダウンロードします。ファイル形式は自動認識されます。ファイル名の拡張子を省略した場合は、.mdx が補われます。

file: ダウンロードするファイル名

offset: オフセット アドレス (MDX バイナリ ファイルの場合は無効)

(例)

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

[*1] ターゲット システムへ NMI 信号を接続していない場合は、キー操作でユーザ プログラムを停止することはできません。代りに、ターゲット システムの ABORT スイッチなどを使用してください。

M *addr1,addr2,addr3*

メモリの内容をブロック転送します。

addr1: 転送元メモリ開始アドレス（16 進数)

addr2: 転送元メモリ終了アドレス（16 進数)

addr3: 転送先メモリ アドレス（16 進数)

(例)

> M 1000,10FF,2000

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

I/O ポートの内容を変更します。***=data*** を省略すると I/O の内容を表示します。

/B: 8 ビット /W: 16 ビット /L: 32 ビット

addr: I/O ポート アドレス（16 進数)

data: I/O ポート変更データ（16 進数)

(例)

> P/W 1000=55

> P/L 2000

Q MDXDEB を終了します。

R レジスタの内容を表示します。

R/F MIPS PowerPC

FPU レジスタの内容を表示します。

R *reg=data* レジスタの内容を変更します。

reg: レジスタ名（「付録 G レジスタ名」参照)

data: レジスタ変更データ（16 進数)

(例)

> R PC=2000

S [*num*] ユーザ プログラムをステップ実行します。**num** を省略した場合は、一回ステップ実行します。

num: ステップ実行回数（10 進数)

(例)

> S

> S 10

V　　デバッガとモニタ プログラムのバージョンを表示します。

第七章 MDXCVT

ファイル変換ツール MDXCVT の操作方法について記述しています。MDXCVT は、S レコード ファイルまたは IEEE695 ファイルを高速ダウンロードするためのツールです。

Windows の場合は、MS-DOS 互換ウィンドウから実行してください。

MDXCVT の起動方法

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

infile	入力ファイル名
outfile	出力ファイル名 (省略時は、**infile** の拡張子を.mdx にしたファイル名)
-c config	コンフィグレーション ファイル名
-s	S レコード ファイルを入力する
-i	IEEE695 ファイルを入力する (デフォルト)
-v	コンフィグレーション ファイルの情報を表示する
-o offset	出力ファイルにオフセット アドレスを加算する

MDXCVT は、S レコード ファイルまたは IEEE695 ファイルを変換し、MDX バイナリ ファイルを作成します。MDX バイナリ ファイルは、MDXDEB の L コマンドで高速ダウンロードすることができます。

【注意】ROM 領域外にあるオブジェクトは変換できません。入力ファイルに ROM 領域外のオブジェクトがある場合、以下の警告メッセージが表示されます。

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

【注意】ROM 領域内のオブジェクトでも、WORKROM 領域と重複するオブジェクトは変換できません。入力ファイルに WORKROM と重複するオブジェクトがある場合、以下のエラー メッセージが表示されます。

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

【注意】以下の、コンフィグレーション ファイルの項目を変更した場合は、MDXCVT を再実行してください。これは、MDXCVT が以下の項目を参照しているためです。

ROM
ROMSIZE
WORKROM
WORKROMSIZE

付録 A 仕様

本体寸法	75mm(高さ)x235mm(幅)x175mm(奥行き)
本体重量	2.3Kg
電源	AC100V 50Hz/60Hz
消費電力	20W Parallel　　30W Ethernet
ROM ケーブル	300mm
外部トリガ ケーブル	300mm
使用温度範囲	0°C～35°C
保存温度範囲	-10°C～55°C
周囲湿度範囲	30%～85%
対応 ROM	「付録 D 対応 ROM」参照
対応 ROM 個数	8 ビット 1 個、2 個、4 個　16 ビット 1 個、2 個
エミュレーションメモリ容量	512K バイト、2MB または 4MB[*1]
アクセス タイム	CS から 75n 秒
インターフェース	専用パラレル Parallel　イーサネット 10BASE-T Ethernet
対応 CPU	「付録 F 対応 CPU」参照
ダウンロード速度	256K バイト/秒 Parallel　1.2M バイト/分 Ethernet
制限事項	「付録 B ターゲット システムの制限事項」参照

対応デバッガ

- 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 (製品名 XHI68KMD)
- XRAY68K 2.2a on MS-DOS PC-98 (製品名 XHI68KMD)
- XRAY68K 3.4 on SunOS/Solaris (製品名 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)

[*1] ROM 読み込みデータ バス幅が 8 ビットの場合、使用できるエミュレーション メモリの容量は半分になります。

付録 B ターゲット システムの制限事項

1. ROM と RAM が完全に動作している。

2. ROM に 16K バイト、RAM に 4K バイトの空き容量がある。

3. RESET 信号と NMI 信号が接続できる方が望ましい。

4. ROM ソケットが実装されている。

5. ROM がバンク化されていない。

6. ROM がキャッシュされていない、または ROM がページアクセスされない。

7. ROM をバイトでアクセスできる。(CPU が ROM をバイトアクセスした場合、ROM の指定番地のみアクセスをする)

8. ROM のアドレス信号が変化するとき、CS または OE が非アクティブになる。(次ページ参照)

9. ROM の CS または OE が上位アドレスをデコードした信号である。(次ページ参照)

10. ROM が複数個実装されている場合、すべての ROM のアドレス信号が同一信号である。

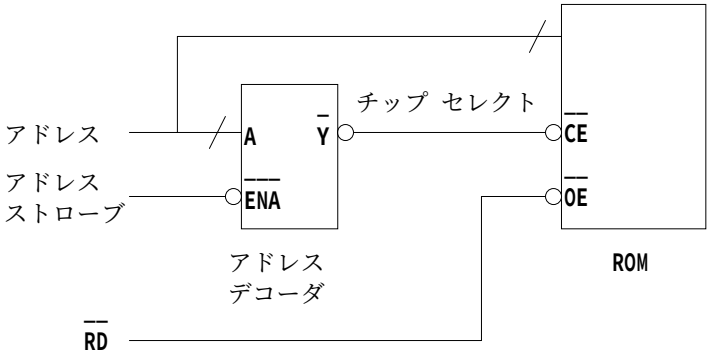
11. RAM 上でプログラムが実行できる。

12. 68000 ROM または RAM がファンクション コード信号(FC0～FC2)で分割されていない。

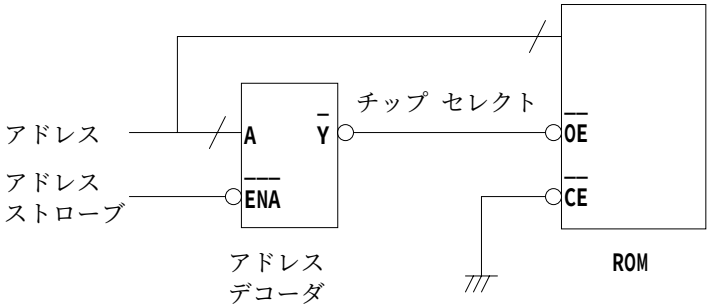
13. PowerPC CPU はビッグ エンディアンのみで動作する。

14. エンディアンを途中で変更しない。

MDX700 が動作するターゲット システム

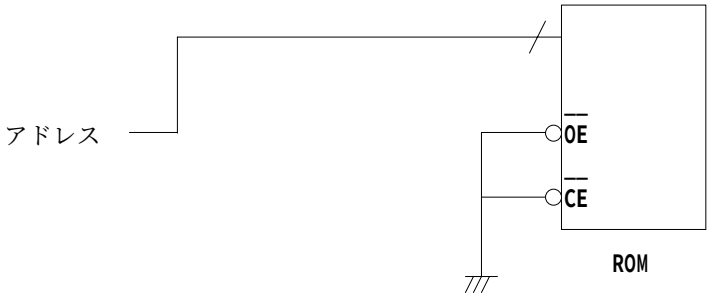


MDX700 が条件付きで動作するターゲット システム



- ROM 読み込みデータ バス幅が 16 ビットの場合、8 ビット ROM なら二個、16 ビット ROM なら一個までしか使用できません。
- ROM 読み込みデータ バス幅が８ビットの場合、ROM を一個しか使用できません。

MDX700 が動作しないターゲット システム



付録 C 注意事項

1. コンフィグレーション ファイルの WORKROM と WORKRAM で指定したメモリ領域に、ユーザ プログラムをダウンロードすると、デバッガが使用できなくなります。

2. NMI 信号をターゲット システムへ接続していない場合は、デバッガの操作[*1] で、ユーザ プログラムを強制ブレークさせないでください。

3. 68000 SH VBR レジスタを、コンフィグレーション ファイル中で初期化してください。ユーザ プログラム実行中に VBR レジスタの値が変更されると、デバッガが使用できなくなります。

4. PowerPC (403 のみ) EVPR レジスタを、コンフィグレーション ファイル中で初期化してください。ユーザ プログラム実行中に EVPR レジスタの値が変更されると、デバッガが使用できなくなります。

5. MIPS ユーザ プログラムは、カーネル モードで TLB を使用しないメモリ空間を使用してください。(0x80000000~0xBFFFFFFF)

6. MIPS ユーザ プログラムは、R27 レジスタを使用できません。R27 レジスタはモニタ プログラムが使用しています。

7. MIPS ユーザ プログラム実行中に、ステータス レジスタの BEV ビット(通常 1)を変更しないでください。

8. V830 以外の V800 次の命令はステップ実行できません。

   HALT
   LDSR
   RETI
   TRAP

9. ARM MIPS PowerPC SH-3 V800 例外ハンドラ内では、次のレジスタをスタックに退避した以降の命令上にしか、ブレークポイントを設定できません。

[*1] デバッガが MULTI の場合、halt コマンドなど。

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

10. MIPS 例外ハンドラ内では、ステータス レジスタの EXL をクリアした以降の命令上にしか、ブレークポイントを設定できません。(多重割り込みを許可する)

11. SH-3 TRAPA 命令上にブレークポイントを設定できません。ブレークポイントで止まることはできますが、そこから継続して実行することができません。継続実行する場合には、TRAPA 命令上のブレークポイントを解除する必要があります。

付録 D 対応 ROM

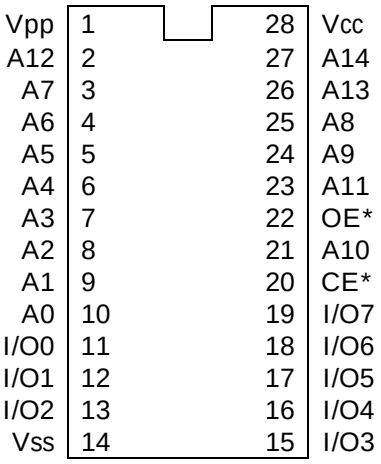
ROM プローブ	ROM 容量	ROM 型番	ROM メーカ
B106 27256	32K x 8bit (0x8000byte)	HN27C256AG HN27C256HG μPD27C256AD TC57256AD TC57H256D M5L27256K M5M27C256K MBM27C256A-nnCZ 27256 27C256	Hitachi Hitachi NEC Toshiba Toshiba Mitsubishi Mitsubishi Fujitsu intel intel
B107 27512	64K x 8bit (0x10000byte)	HN27512G HN27C512AG 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 プローブ	ROM 容量	ROM 型番	ROM メーカ
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

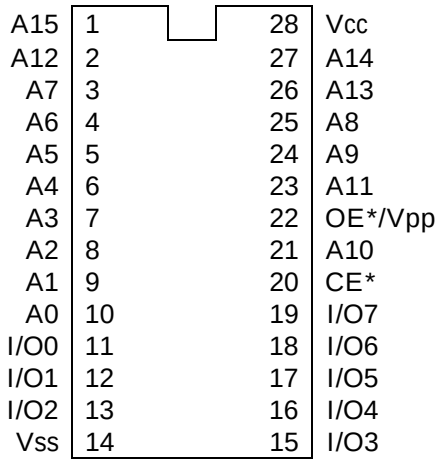
ROM プローブ	ROM 容量	ROM 型番	ROM メーカ
B113 274096 DIP	256K x 16bit (0x80000byte)	HN27C4096G HN27C4096HG HN27C4096AG HN27C4096AHG TC574096D MBM27C4096-nnZ 27C240	Hitachi Hitachi Hitachi Hitachi Toshiba Fujitsu intel
PL44 271024 PLCC PL44 274096 PLCC 表「PL44 ジャンパ」参照	64K x 16bit (0x20000byte) 256K x 16bit (0x80000byte)	HN27C1024HCC MBM27C1024-nnTV HN27C4096CC HN27C4096HCC HN27C4096ACC HN27C4096AHCC	Hitachi Fujitsu Hitachi Hitachi Hitachi 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 ジャンパ	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

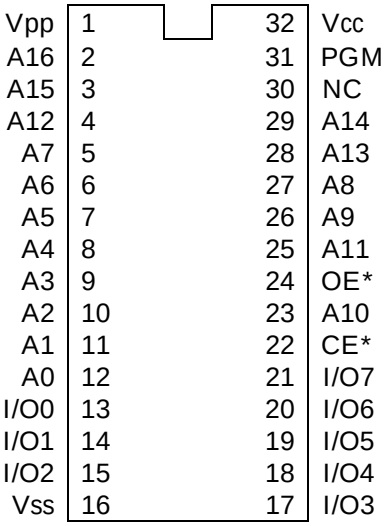
付録 E 対応 ROM ピンアサイン



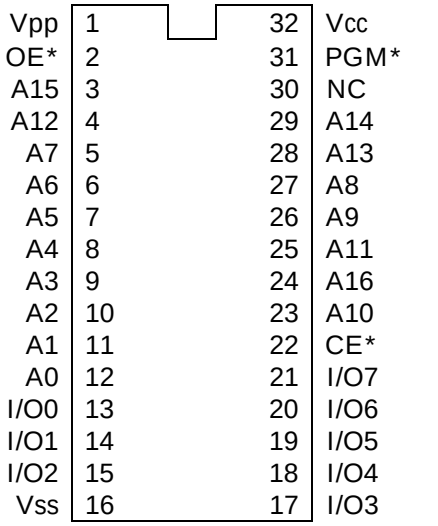
B106 27256



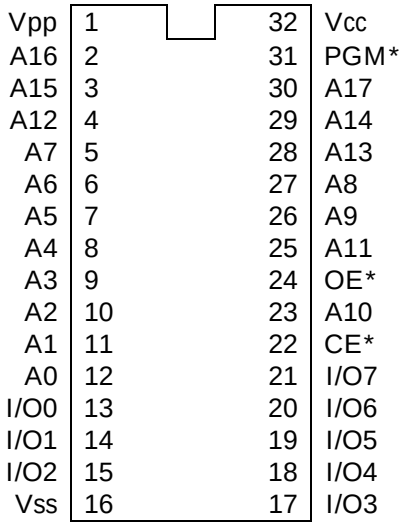
B107 27512



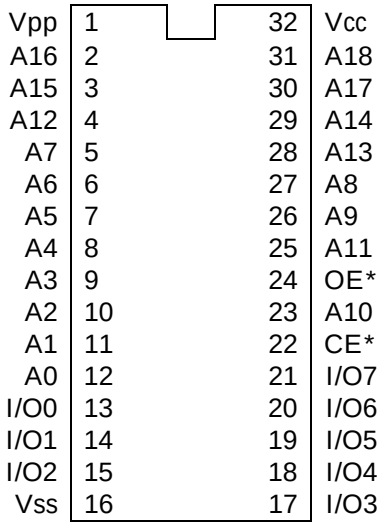
B108 27010



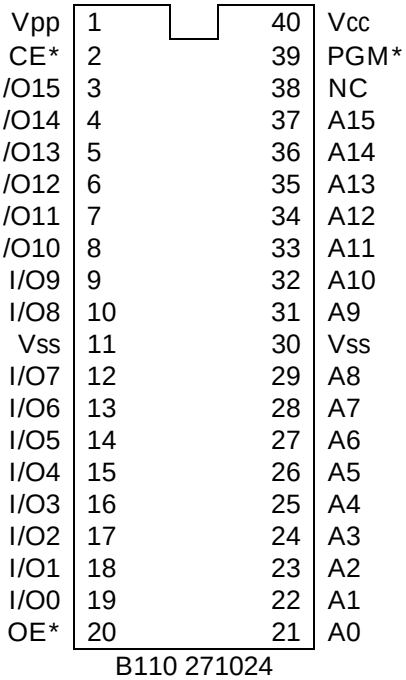
B109 271000



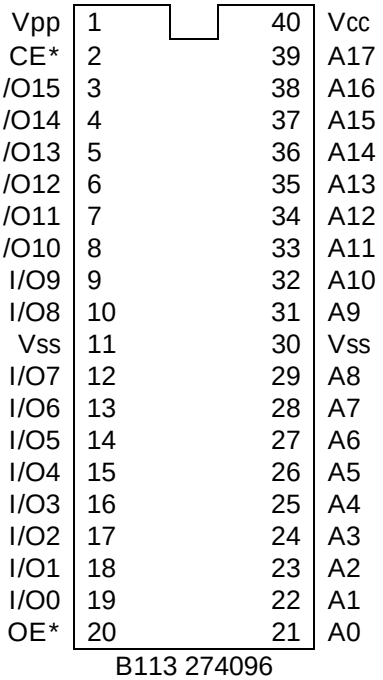
B111 27020



B112 27040



B110 271024



B113 274096

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

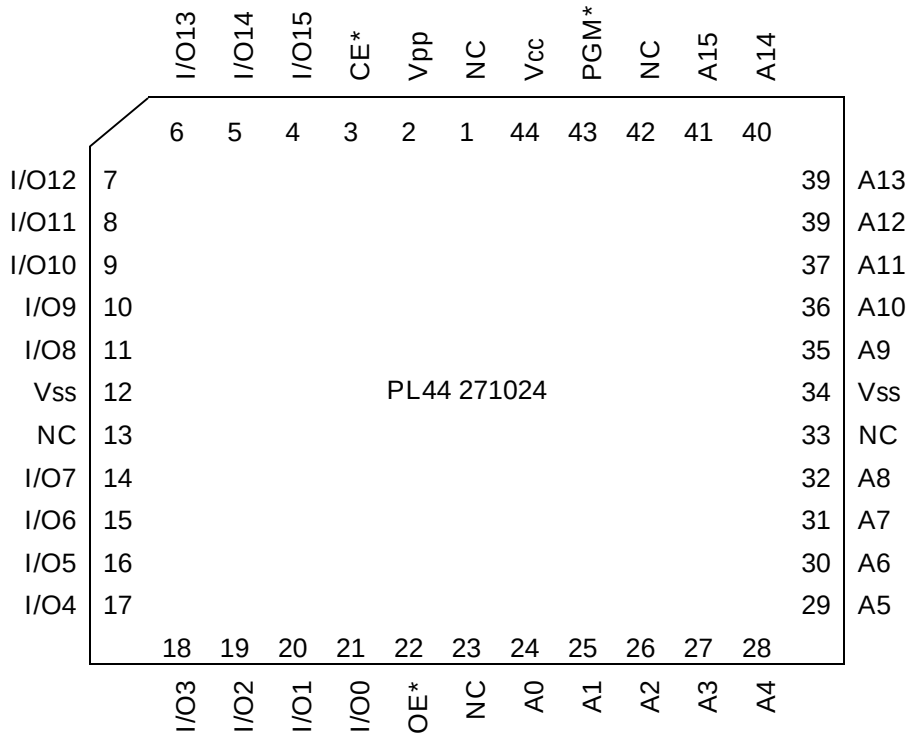
B117/B118
27C4000

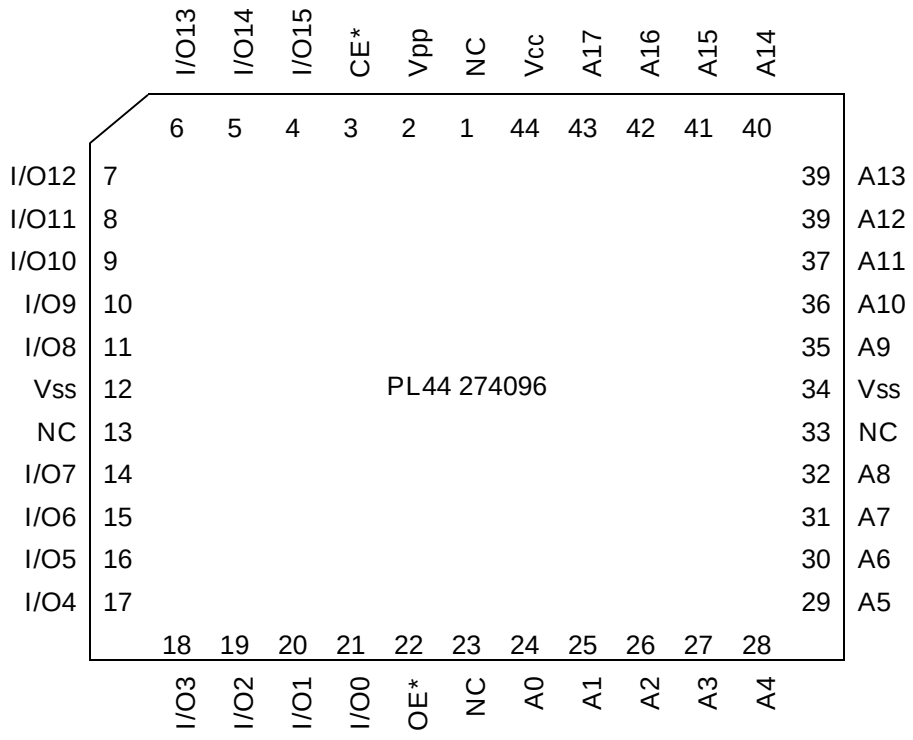
A18	1	42	NC
A17	2	41	A8
A7	3	40	A9
A6	4	39	A10
A5	5	38	A11
A4	6	37	A12
A3	7	36	A13
A2	8	35	A14
A1	9	34	A15
A0	10	33	A16
CE*	11	32	BYTE*/Vpp
Vss	12	31	Vss
OE*	13	30	I/O15/A-1
I/O0	14	29	I/O7
I/O8	15	28	I/O14
I/O1	16	27	I/O6
I/O9	17	26	I/O13
I/O2	18	25	I/O5
I/O10	19	24	I/O12
I/O3	20	23	I/O4
I/O11	21	22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


I/O13 6
I/O14 5
I/O15 4
CE* 3
Vpp 2
NC 1
Vcc 44
PGM* 43
NC 42
A15 41
A14 40
I/O12 7
I/O11 8
I/O10 9
I/O9 10
I/O8 11
Vss 12
NC 13
I/O7 14
I/O6 15
I/O5 16
I/O4 17
39 A13
39 A12
37 A11
36 A10
35 A9
34 Vss
33 NC
32 A8
31 A7
30 A6
29 A5
PL44 271024
18 I/O3
19 I/O2
20 I/O1
21 I/O0
22 OE*
23 NC
24 A0
25 A1
26 A2
27 A3
28 A4


I/O13 6
I/O14 5
I/O15 4
CE* 3
Vpp 2
NC 1
Vcc 44
A17 43
A16 42
A15 41
A14 40
I/O12 7
I/O11 8
I/O10 9
I/O9 10
I/O8 11
Vss 12
NC 13
I/O7 14
I/O6 15
I/O5 16
I/O4 17
39 A13
39 A12
37 A11
36 A10
35 A9
34 Vss
33 NC
32 A8
31 A7
30 A6
29 A5
PL44 274096
18 I/O3
19 I/O2
20 I/O1
21 I/O0
22 OE*
23 NC
24 A0
25 A1
26 A2
27 A3
28 A4

付録 F 対応 CPU

- 大文字と小文字は区別します。_LE はリトル エンディアン、_64 は 64 ビット レジスタ、_LE64 はリトル エンディアンで 64 ビット レジスタをあらわします。
- 表内にない CPU の場合、コア CPU が同じ CPU を指定してください。同様に対応できます。

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					

付録Ｇ レジスタ名一覧

- _HI は、64 ビットレジスタの上位 32 ビットをあらわします。
- _LO は、64 ビットレジスタの下位 32 ビットをあらわします。
- CPn は、コプロセッサ０レジスタの n 番のレジスタをあらわします。 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

付録 H 動作原理

デバッガ起動時の内部動作

1. コンフィグレーション ファイルを読み込みます。
2. コンフィグレーション ファイルの WORKROM で指定されたアドレスへ、モニタプログラムとコンフィグレーション ファイルの内容をダウンロードします。*1
3. コンフィグレーション ファイルの RESETVECTOR で指定されたアドレスへ、モニタ プログラムをリセット スタートさせるための、リセットベクタをダウンロードします。
4. RESET 信号を出力します。RESET 信号をターゲット システムに接続している場合は、モニタ プログラムがリセット スタートし、モニタ プログラムとの通信を開始します。
5. モニタプログラムから例外発生時のアドレスを取得します。*2
6. 例外発生時のアドレスから例外ベクタを作成します。
7. 例外ベクタをターゲット システムへダウンロードします。

上記の内部動作により、モニタ プログラムはコンフィグレーション ファイルの WORKROM の変更だけで移動できます。また、モニタ プログラムのソース ファイルの中に例外ベクタを記述しないですみます。

モニタ プログラムの機能

モニタ プログラムは、デバッガからの機能要求を受け取り、それを実行し、結果をデバッガに返す、という処理を繰り返しています。モニタ プログラムは、次の機能要求を実行します。

- RAM の読み出し
- RAM の書き込み
- ユーザ プログラムの実行
- ユーザ プログラムのステップ実行 [68000]
- 次の PC の計算 [ARM] [MIPS] [PowerPC] [SH] [V800]

デバッガは、これらの機能要求を、必要に応じてモニタ プログラムへ送信します。

---

*1 モニタ プログラムからコンフィグレーション ファイルの内容を参照できるようにします。

*2 モニタ プログラムがダウンロードされるアドレスは WORKROM によって変化するため、デバッガは動的に例外ベクタのアドレスを取得しています。

ブレークポイント機能

ブレークポイント機能は、CPU の不当命令例外を利用して実現しています。ブレークポイントが設定されたアドレスの命令は、ユーザ プログラム実行直前に、不当命令に置き換えられ、ユーザ プログラムがブレークした後、自動的に元の命令に戻されます。

ブレークポイントに使用している不当命令は、次のとおりです。

- 0x4AFC [68000] (ILLEGAL 命令)
- 0xE7FFE7FF [ARM]
- 0xDEDE [THUMB]
- 0x0000000D [MIPS] (BREAK 命令)
- 0x00000000 [PowerPC]
- 0xF00F [SH]
- 0x6C6C [V830] (BRKPNT 命令)
- 0xFFFFFFFF [V850]
- 0xF840 [V850E]
- 0x5858 [V830] [V850] [V850E]以外の[V800]

ステップ実行機能

[68000]の場合、ステップ実行機能は、CPU のトレース例外を利用して実現しています。

[ARM] [MIPS] [PowerPC] [SH] [V800]の CPU の場合、ブレークポイントと同様に不当命令を利用して実現しています。詳しくは、命令実行後の PC を計算し、その位置にブレークポイント(不当命令)を設定し、ユーザ プログラムを実行します。

強制ブレーク機能

強制ブレーク機能は、次の例外を使用して実現しています。

- レベル 7 割り込み [68000]
- コンフィグレーション ファイルの ABORTVECTOR で指定された例外 [ARM] [PowerPC]

- NMI 例外 MIPS SH V800

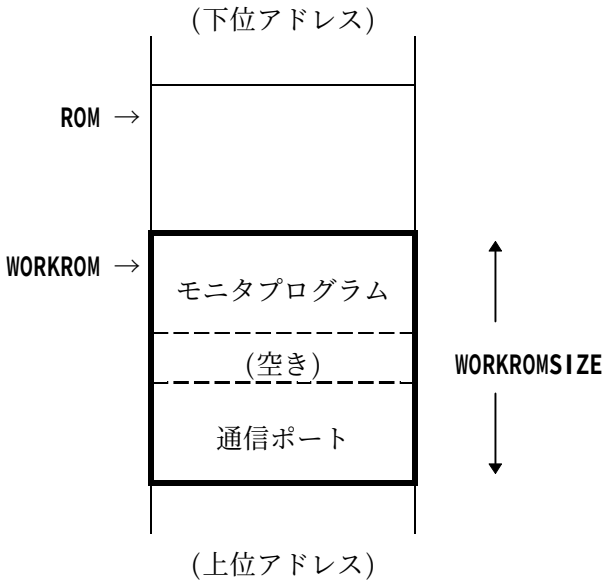
デバッガから、実行中のユーザ プログラムを停止させる操作[*3] を行った場合、デバッガは外部トリガ ケーブルの NMI 信号を出力します。これにより、ユーザプログラムが停止します。[*4]

[*3] デバッガが MULTI の場合、halt コマンドなど。

[*4] NMI 信号をターゲット システムへ接続していない場合は、デバッガによる強制ブレークはできません。

付録 I 通信ポート領域

コンフィグレーション ファイルの WORKROM と WORKROMSIZE で指定した領域には、先頭からモニタ プログラムがダウンロードされます。また、この領域の最後からは、モニタ プログラムとホストとの通信ポートとしても使用されます。



通信ポートの大きさは、コンフィグレーション ファイルの BUS の指定によって、次のように変わります。

BUS 8	512 バイト
BUS 16	1024 バイト
BUS 32	2048 バイト
BUS 64	4096 バイト

したがって、通信ポートの先頭アドレスは、次の式で計算できます。

通信ポート先頭アドレス **= (WORKROM + WORKROMSIZE) - 512 * (BUS / 8)**

【注意】通信ポートの先頭アドレスは、通信ポートの大きさのバウンダリ上にある必要があります。

【注意】WORKROM と WORKROMSIZE を変更する場合は、モニタ プログラムと通信ポートの領域が重複しないようにしてください。

【注意】WORKROM と WORKROMSIZE を変更する場合は、通信ポートの先頭アドレスがバウンダリ上に配置されるようにしてください。

付録 J モニタ プログラムの例外ベクタ

- (大文字)は、コンフィグレーション ファイル内で指定された値をあらわします。
- 大文字は、モニタ プログラムのソース ファイル内のラベルをあらわします。
- スラッシュで区切られているラベルは、CPU によって切り替わります。
- (nop)は、次の命令まで nop が連続していることをあらわします。

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      同上
(REG_VBR)+0x7C      同上
```

MIPS R3xxx

```
(RESETVECTOR)       j       (WORKROM)       リセット時のみ
(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)        リセット時のみ
(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 (403 以外) MSR[IP]=0

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

PowerPC (403 以外) 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
```

V830 V850 V850E 以外　V800

```
(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

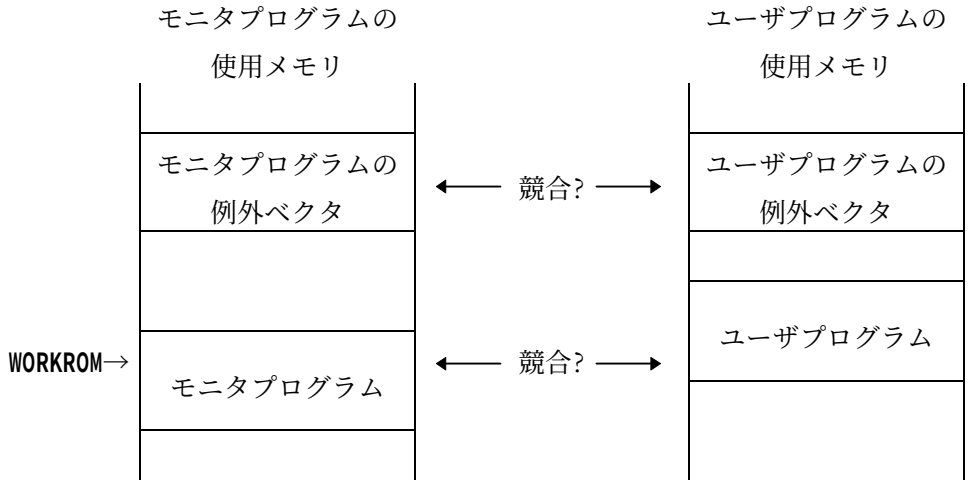
```
(RESETVECTOR)        movhi   hi(WORKROM), r0, r1
(RESETVECTOR)+0x04   ori     lo(WORKROM), r1, r1
(RESETVECTOR)+0x08   jmp     [r1]
0x00000010           st.w    r1, -8[sp]
0x00000014           movhi   hi(ENTRY_V800), r0, r1
0x00000018           ori     lo(ENTRY_V800), r1, r1
0x0000001C          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

付録 K ユーザプログラムとモニタプログラムの共存

ユーザ プログラムを作成するときは、モニタ プログラムとメモリが競合しないように、以下の点に注意してください。メモリが競合している場合、デバッガが使用できません。

- コンフィグレーション ファイルの WORKROM/WORKROMSIZE と WORKRAM/WORKRAMSIZE で指定した領域を、ユーザ プログラムで使用しないこと。

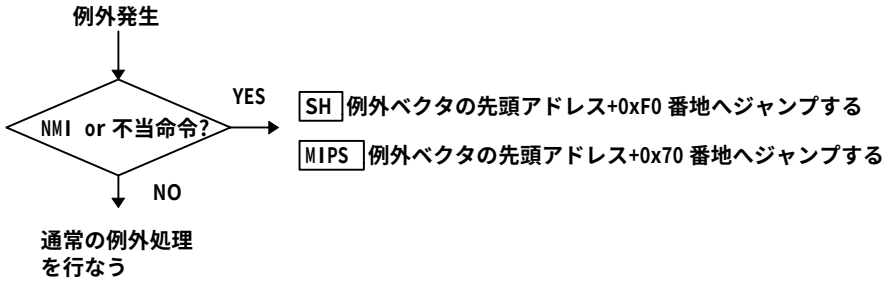
- モニタ プログラムの例外ベクタ領域を、ユーザ プログラムで使用しないこと。モニタ プログラムの例外ベクタについては、「付録 J モニタ プログラムの例外ベクタ」を参照してください。



ユーザ プログラムとモニタ プログラムが競合している場合は、ユーザ プログラムのロードアドレスをモニタ プログラムと競合しない位置へ移動してください。

例外ベクタのみが競合している場合は、デバッガのコマンドで競合を回避できる場合があります。ユーザ プログラムをダウンロードした後、I コマンドを実行すると、モニタ プログラムの例外ベクタが再ロードされます。

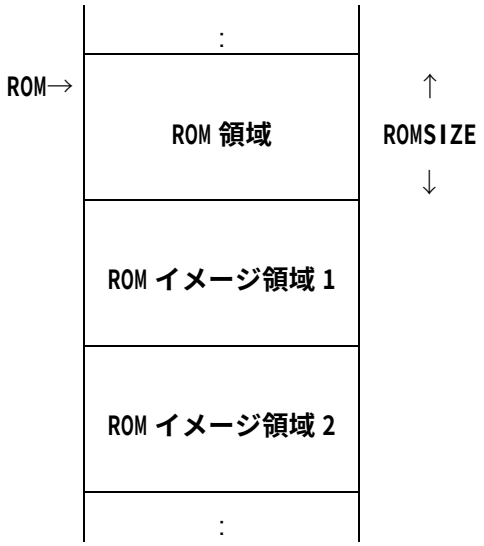
MIPS SH-3 の場合、複数の例外がひとつの例外ベクタを使用しているため、どうしても例外ベクタが競合します。これらの CPU を使用する場合は、NMI と不当命令で使用する例外ベクタの先頭に、次の処理を追加してください。



上記の処理は、ブレークポイントのための不当命令と、強制ブレークのための NMI が発生した場合のみ、モニタ プログラムへジャンプさせるための処理です。これらの処理を追加できるように、モニタ プログラムの例外ベクタは、二つの分岐命令で構成されています。詳しくは、「付録 J モニタ プログラムの例外ベクタ」を参照してください。

付録 L ROM イメージ領域を使用する

アドレスをフル デコードしていないターゲット システムでは、ROM 領域と同じようにアクセスできる ROM イメージ領域が存在します。



通常の環境設定を行なうと、ROM イメージ領域は RAM と同様に扱われるため、データを読み出すことはできますが、書き込むことはできません。

ROM イメージ領域も ROM 領域と同じように書き込み可能にするためには、コンフィグレーション ファイルの中で、ROM_IMAGE という項目を追加指定します。次のように、ROM_IMAGE に ROM イメージ領域の先頭アドレスを指定してください。

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

【注意】ROM_IMAGE は、コンフィグレーション ファイルの中で、一回だけ指定することができます。

【注意】ROM イメージ領域の大きさは、ROMSIZE と同じになります。

付録 M キャッシュ ROM 領域を使用する

キャッシュ ROM 領域を使用する場合は、次の手順で行なってください。

1. SH MIPS V800 の場合は、キャッシュ ROM 領域を ROM イメージ領域として指定してます。詳しくは、「付録 L ROM イメージ領域を使用する」を参照してください。

2. モニタ プログラムを変更します。モニタ プログラムのソース ファイルをオープンし、次のラベル部分に、すべてのキャッシュをフラッシュするコードを追加してください。

   USER_UPDATE
   USER_RUN

3. 変更したソース ファイルをアセンブルします。アセンブル手順は、デバッガのリリース ノートを参照してください。

USER_UPDATE は、メモリが変更された後、呼ばれるサブルーチンです。USER_RUN は、ユーザ プログラムを実行する直前に呼ばれるサブルーチンです。これらのサブルーチンの中でキャッシュをフラッシュすることにより、ROM 領域が書き換られた場合でも、CPU にそれを認識させることができます。

付録 N エラー メッセージ

MDXERR: bad host name: xxxx Ethernet

ホスト名が間違っています。ホスト名が登録されているか、確認してください。

MDXERR: connection refused: Ethernet

コネクションが拒絶されました。MDX700 の IP アドレスが、指定したホスト名の IP アドレスと一致しているか、確認してください。

MDXERR: bad communication port Parallel

パラレル インターフェース ボードにアクセスできません。MDX700 の電源が投入されているか、確認してください。また、パラレル インターフェース ボードのスイッチの設定とコンフィグレーション ファイルの PORT の値が一致しているか、確認してください。

MDXERR: file not found: xxxx

ファイルが見つかりません。ファイル名やディレクトリ名が正しいか、確認してください。

MDXERR: bad configuration file: xxxx

コンフィグレーション ファイルの内容が間違っています。変更した内容が正しいか、確認してください。また、(boundary)が表示された場合、通信ポートの先頭アドレスが要求バウンダリ上になるように、WORKROM と WORKROMSIZE を設定してください。
詳しくは、「付録 I 通信ポート領域」を参照してください。

MDXERR: bad monitor file:

モニタ プログラムの内容が正しく読み込めません。モニタ プログラムが、S レコード ファイルになっているか、確認してください。

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

ファイルの形式が、MDX バイナリ ファイルではありません。指定ファイルが、MDXCVT で変換したファイルであるかを確認してください。

MDXERR: communication port timeout:

デバッガとモニタ プログラムとの通信が行なえません。MDX700 とターゲット システムの接続が正しいか、確認してください。
詳しくは「付録 O トラブル シューティング」を参照してください。

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

デバッガとモニタ プログラムとの通信が行なえません。しかし、モニタ プログラムの起動には成功していました。WORKRAM のアドレスに RAM が存在するか、確認してください。
また、モニタ プログラムを変更た場合、変更したコードが正しいか、確認してください。

付録 O トラブル シューティング

「**MDXERR: communication port timeout:**」は、デバッガとモニタ プログラム間の通信が行なえないことをあらわすエラー メッセージです。このエラー メッセージが表示された場合は、まず、次の項目を確認してください。

- MDX700 とターゲット システムの電源が投入されているか
- MDX700 とターゲット システムの接続が正しいか、ROM プローブを逆差ししていないか
- コンフィグレーション ファイルの設定が正しいか

もし、エラーの原因がわからない場合は、次の手順にしたがってください。エラーの原因をある程度特定できます。

ステップ 1

1. RESET 信号を接続している場合は、それを外す
2. モニタ プログラムを変更している場合は、出荷時のものに戻す

ステップ 2

3. MDXDEB を起動する(エラーは無視する)
4. E コマンドで ROM 領域のメモリを書き換える
5. D コマンドで書き換えたメモリの内容を確認する

ROM 領域の書き換えができる場合、次の項目が正しく設定されています。ROM 領域の書き換えができない場合、これらの項目を、もう一度確認してください。

- MDX700 とホストの接続
- パラレル インターフェース ボードのスイッチの設定 Parallel
- コンフィグレーション ファイルの PORT Parallel

ステップ 3

6. 手動でターゲット システムをリセットする
7. V コマンドでバージョンを表示する

V コマンドがエラーなく実行できる場合は、さらに、次の項目が正しく設定されてます。エラーが表示される場合、これらの項目を、もう一度確認してください。

- MDX700 とターゲット システムの接続
- コンフィグレーション ファイルの BUS
- コンフィグレーション ファイルの ROM
- コンフィグレーション ファイルの ROMSIZE
- コンフィグレーション ファイルの WORKROM
- コンフィグレーション ファイルの WORKROMSIZE
- コンフィグレーション ファイルの RESETVECTOR

ステップ 4

8. I コマンドで再初期化をする

I コマンドがエラーなく実行できる場合は、さらに、次の項目が正しく設定されてます。エラーが表示される場合、これらの項目を、もう一度確認してください。

- コンフィグレーション ファイルの WORKRAM
- コンフィグレーション ファイルの WORKRAMSIZE
- コンフィグレーション ファイルの TIMER

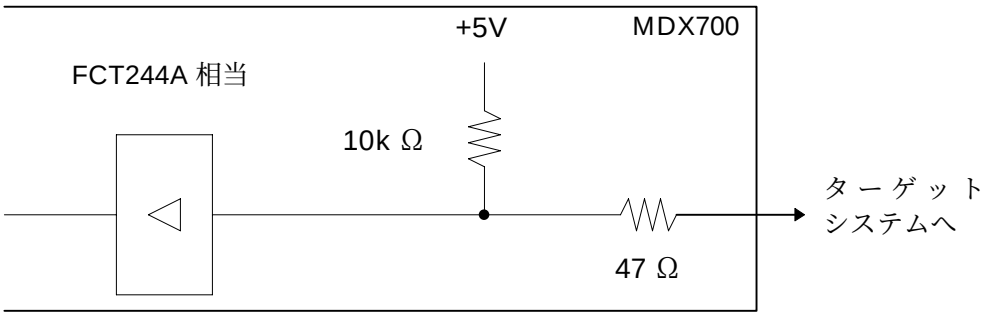
ステップ 5

9. RESET 信号を使用する場合、ターゲット システムへ接続してから、もう一度、ステップ 2 から繰り返してください。エラーが起こる場合、ターゲット システムの接続先を確認してください。また、コンフィグレーション ファイルの TIMER を増やしてみてください。

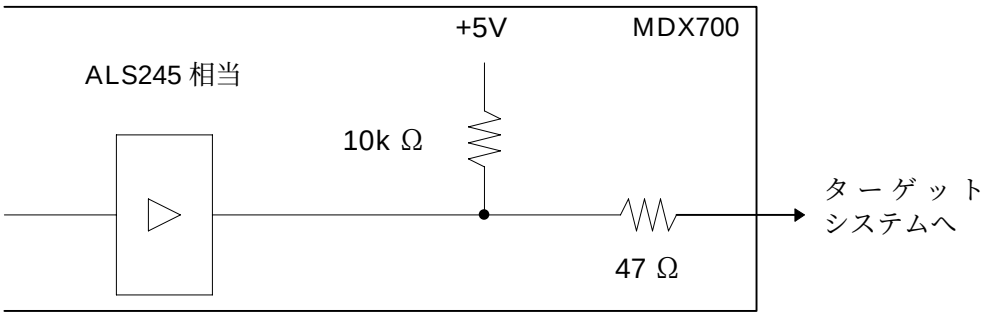
10. モニタ プログラムを変更している場合、変更後のモニタ プログラムを指定して、もう一度、ステップ 2 から繰り返してください。エラーが起こる場合、変更した部分が間違っていないか確認してください。

付録 P ターゲット システムへのプロービング

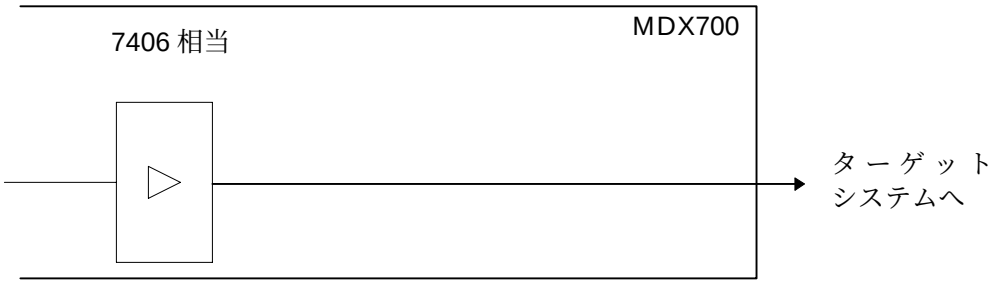
ADDRESS, CS, OE



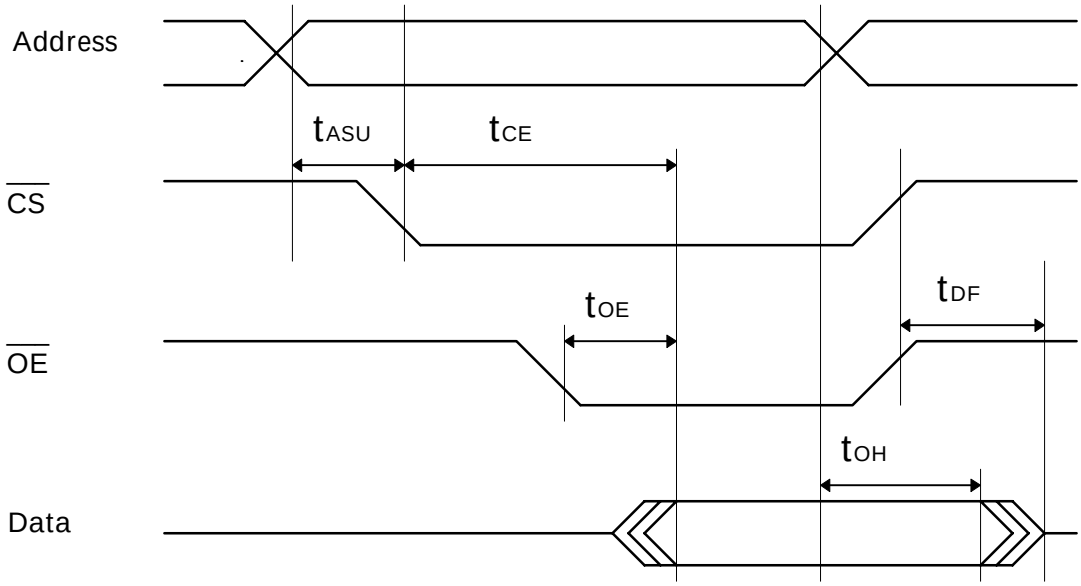
Data



RESET, NMI



付録 Q データ アクセスのタイミング



	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

付録 R ROM ケーブルのコネクタのピンアサイン

ROM プローブへ接続する側のコネクタのピンアサインです。( )内は、コネクタの型番です。

RC28AD/RC28D

(ヒロセ 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

(ヒロセ 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

RC40AD/RC40D

(ヒロセ HIF6A-68PA-1.27DSA)

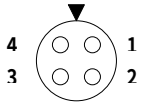
ROMCODE0	A01	B01	DATA00
ROMCODE1	A02	B02	GND
ROMCODE2	A03	B03	DATA01
ROMCODE3	A04	B04	GND
ADDR00	A05	B05	DATA02
ADDR01	A06	B06	GND
ADDR02	A07	B07	DATA03
ADDR03	A08	B08	GND
ADDR04	A09	B09	DATA04
ADDR05	A10	B10	GND
ADDR06	A11	B11	DATA05
ADDR07	A12	B12	GND
ADDR08	A13	B13	DATA06
ADDR09	A14	B14	GND
ADDR10	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:開放

ROMCODE 3	2	1	0	ROM プローブ
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

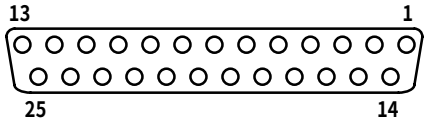
付録 S PWR コネクタのピンアサイン

(ヒロセ HR10A-7R-4S)



ピン番号	信号名	説明
1	+5V	
2	GND	
3	未接続	
4	未接続	

付録 T RS-232C コネクタのピンアサイン Ethernet



ピン番号	信号名	説明
2	RxD	受信データ
3	TxD	送信データ
4	CTS	クリア ツー センド
5	RTS	リクエスト ツー センド
6	20 ピンと短絡	
7	GND	グランド
20	6 ピンと短絡	
その他	未接続	

付録 U コンフィグレーション ファイル例

```
*
*  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
```