ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

**Artisan Technology Group**
(217) 352-9330 | sales@artisantg.com | artisantg.com

**Full-service, independent repair center**
with experienced engineers and technicians on staff.

**We buy your excess, underutilized, and idle equipment**
along with credit for buybacks and trade-ins.

**Custom engineering**
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

**Expert team | Trust guarantee | 100% satisfaction**

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Emerson / Motorola MVME712-533* at our website: ***Click HERE***

# MVME172FX
# 400/500-Series
# VME Embedded Controller
# Installation and Use

**VME172FXA/IH3**

## Notice

While reasonable efforts have been made to assure the accuracy of this document, Motorola, Inc. assumes no liability resulting from any omissions in this document, or from the use of the information obtained therein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

No part of this material may be reproduced or copied in any tangible medium, or stored in a retrieval system, or transmitted in any form, or by any means, radio, electronic, mechanical, photocopying, recording or facsimile, or otherwise, without the prior written permission of Motorola, Inc.

It is possible that this publication may contain reference to, or information about Motorola products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

## Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by Motorola, Inc.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282-9602

# Preface

This document provides general information and basic installation instructions for the 400/500-Series MVME172FX VME Embedded Controller (which is available in the versions listed below). The MVME172FX VME Embedded Controller will be referred to as MVME172 throughout this document.

| Assembly Item | Board Description |
|---|---|
| MVME172-410 | 4MB DRAM, 64MHz MC68LC060 |
| MVME172-411 | 4MB DRAM, 64MHz MC68LC060 and SCSI Interface |
| MVME172-412 | 4MB DRAM, 64MHz MC68LC060 and Ethernet Interface |
| MVME172-413 | 4MB DRAM, 64MHz MC68LC060, SCSI & Ethernet Interfaces |
| MVME172-430 | 8MB DRAM, 64 MHz MC68LC060 |
| MVME172-431 | 8MB DRAM, 64MHz MC68LC060 and SCSI Interface |
| MVME172-432 | 8MB DRAM, 64MHz MC68LC060 and Ethernet Interface |
| MVME172-433 | 8MB DRAM, 64MHz MC68LC060, SCSI & Ethernet interface |
| MVME172-450 | 16MB DRAM, 64MHz MC68LC060 |
| MVME172-451 | 16MB DRAM, 64MHz MC68LC060 and SCSI interface |
| MVME172-452 | 16MB DRAM, 64MHz MC68LC060 and Ethernet interface |
| MVME172-453 | 16MB DRAM, 64MHz MC68LC060, SCSI and Ethernet interface |
| MVME172-510 | 4MB DRAM, 60MHz MC68060 |
| MVME172-511 | 4MB DRAM, 60MHz MC68060 and SCSI interface |
| MVME172-512 | 4MB DRAM, 60MHz MC68060 and Ethernet interface |
| MVME172-513 | 4MB DRAM, 60MHz MC68060, SCSI & Ethernet interfaces |
| MVME172-520 | 8MB DRAM, 60MHz MC68060 |
| MVME172-521 | 8MB DRAM, 60MHz MC68060 and SCSI interface |
| MVME172-522 | 8MB DRAM, 60MHz MC68060 and Ethernet interface |
| MVME172-523 | 8MB DRAM, 60MHz MC68060, SCSI and Ethernet interfaces |
| MVME172-530 | 16MB DRAM, 60MHz MC68060 |
| MVME172-531 | 16MB DRAM, 60MHz MC68060 and SCSI interface |
| MVME172-532 | 16MB DRAM, 60MHz MC68060 and Ethernet interface |
| MVME172-533 | 16MB DRAM, 60MHz MC68060, SCSI & Ethernet interfaces |

This manual is intended for anyone who wants to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes.

Anyone using this manual should have a basic knowledge of computers and digital logic. Companion publications are listed in Chapter 1.

# Safety Summary
# Safety Depends On You

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment. Motorola, Inc. assumes no liability for the customer's failure to comply with these requirements.

The safety precautions listed below represent warnings of certain dangers of which Motorola is aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

## Ground the Instrument.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. The equipment is supplied with a three-conductor AC power cable. The power cable must be plugged into an approved three-contact electrical outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

## Do Not Operate in an Explosive Atmosphere.

Do not operate the equipment in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment constitutes a definite safety hazard.

## Keep Away From Live Circuits.

Operating personnel must not remove equipment covers. Only Factory Authorized Service Personnel or other qualified maintenance personnel may remove equipment covers for internal subassembly or component replacement or any internal adjustment. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

## Do Not Service or Adjust Alone.

Do not attempt internal service or adjustment unless another person capable of rendering first aid and resuscitation is present.

## Use Caution When Exposing or Handling the CRT.

Breakage of the Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, avoid rough handling or jarring of the equipment. Handling of the CRT should be done only by qualified maintenance personnel using approved safety mask and gloves.

## Do Not Substitute Parts or Modify Equipment.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that safety features are maintained.

## Dangerous Procedure Warnings.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed. You should also employ all other safety precautions which you deem necessary for the operation of the equipment in your operating environment.



**WARNING**

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

All Motorola PWBs (printed wiring boards) are manufactured by UL-recognized manufacturers, with a flammability rating of 94V-0.



**WARNING**

This equipment generates, uses, and can radiate electromagnetic energy. It may cause or be susceptible to electromagnetic interference (EMI) if not installed and used in a cabinet with adequate EMI protection.



European Notice: Board products with the CE marking comply with the EMC Directive (89/336/EEC). Compliance with this directive implies conformity to the following European Norms:

EN55022 (CISPR 22) Radio Frequency Interference

EN50082-1 (IEC801-2, IEC801-3, IEC801-4) Electromagnetic Immunity

The product also fulfills EN60950 (product safety) which is essentially the requirement for the Low Voltage Directive (73/23/EEC).

This board product was tested in a representative system to show compliance with the above mentioned requirements. A proper installation in a CE-marked system will maintain the required EMC/safety performance.

The computer programs stored in the Read Only Memory of this device contain material copyrighted by Motorola Inc., 1995, and may be used only under a license such as those contained in Motorola's software licenses.

Motorola® and the Motorola symbol are registered trademarks of Motorola, Inc.

All other products mentioned in this document are trademarks or registered trademarks of their respective holders.

©Copyright Motorola 1998
All Rights Reserved

Printed in the United States of America
June 1998

# Contents

## CHAPTER 1 Board Level Hardware Description


Introduction ..... 1-1
Overview ..... 1-1
Related Documentation ..... 1-3
Other Applicable Motorola Publications ..... 1-4
Applicable Non-Motorola Publications ..... 1-4
MCG Customer Services ..... 1-6
Requirements ..... 1-7
Features ..... 1-7
Specifications ..... 1-9
Cooling Requirements ..... 1-10
Special Considerations for Elevated Temperature Operation ..... 1-10
FCC Compliance ..... 1-12
Manual Terminology ..... 1-12
Block Diagram ..... 1-14
Functional Description ..... 1-15
Front Panel Switches and Indicators ..... 1-15
Data Bus Structure ..... 1-16
MC68060 or MC68LC060 MPU ..... 1-16
MC68xx060 Cache ..... 1-16
No-VMEbus-Interface Option ..... 1-17
Memory Options ..... 1-17
DRAM Options ..... 1-17
SRAM Options ..... 1-18
About the Battery ..... 1-19
EPROM and Flash Memory ..... 1-21
Battery Backed Up RAM and Clock ..... 1-22
VMEbus Interface and VMEchip2 ..... 1-22
I/O Interfaces ..... 1-23
Serial Communications Interface ..... 1-23
IndustryPack (IP) Interfaces ..... 1-26
Optional LAN Ethernet Interface ..... 1-27
Optional SCSI Interface ..... 1-28
SCSI Termination ..... 1-28
Local Resources ..... 1-28
Programmable Tick Timers ..... 1-28

Watchdog Timer ..... 1-28
Software-Programmable Hardware Interrupts ..... 1-29
Local Bus Timeout ..... 1-29
Local Bus Arbiter ..... 1-30
Timing Performance ..... 1-30
Local Bus to DRAM Cycle Times ..... 1-30
EPROM/Flash Cycle Times ..... 1-31
SCSI Transfers ..... 1-31
LAN DMA Transfers ..... 1-31
Connectors ..... 1-32
Remote Status and Control ..... 1-33
Memory Maps ..... 1-33
Local Bus Memory Map ..... 1-33
Normal Address Range ..... 1-34
VMEbus Memory Map ..... 1-38
VMEbus Accesses to the Local Bus ..... 1-38
VMEbus Short I/O Memory Map ..... 1-38
Software Initialization ..... 1-39
Multi-MPU Programming Considerations ..... 1-39
Local Reset Operation ..... 1-39

## CHAPTER 2 Hardware Preparation and Installation

Introduction ..... 2-1
Unpacking Instructions ..... 2-1
Hardware Preparation ..... 2-2
System Controller Select Header (J1) ..... 2-3
SIMM Selection (J15) ..... 2-5
Removal of Existing SIMM ..... 2-6
Installation of New SIMM ..... 2-8
Clock Select Header (J16) for Serial Port 1/Console ..... 2-9
Clock Select Header (J17) for Serial Port 2 ..... 2-9
SRAM Battery Backup Source Select Header (J22) ..... 2-10
IP DMA Snoop Control Header (J2) ..... 2-11
EPROM Size Select Header (J23) ..... 2-12
General Purpose Readable Jumpers Header (J28) ..... 2-12
Flash Write Protect Header (J24) ..... 2-13
IP Bus Clock Header (J14) ..... 2-14
IP Bus Strobe Select Header (J19) ..... 2-15
Installation Instructions ..... 2-16
IP Installation on the MVME172 ..... 2-16

MVME172 Module Installation....2-17
System Considerations....2-19

## CHAPTER 3 Debugger General Information

Overview of M68000 Firmware....3-1
Description of 172Bug....3-1
172Bug Implementation....3-3
Installation and Startup....3-4
Autoboot....3-9
ROMboot....3-10
Network Boot....3-10
Restarting the System....3-11
Reset....3-12
Abort....3-13
Break....3-13
SYSFAIL* Assertion/Negation....3-14
MPU Clock Speed Calculation....3-14
Memory Requirements....3-15
Disk I/O Support....3-16
Blocks Versus Sectors....3-16
Device Probe Function....3-17
Disk I/O via 172Bug Commands....3-17
IOI (Input/Output Inquiry)....3-17
IOP (Physical I/O to Disk)....3-18
IOT (I/O Teach)....3-18
IOC (I/O Control)....3-18
BO (Bootstrap Operating System)....3-18
BH (Bootstrap and Halt)....3-18
Disk I/O via 172Bug System Calls....3-18
Default 172Bug Controller and Device Parameters....3-20
Disk I/O Error Codes....3-20
Network I/O Support....3-21
Intel 82596 LAN Coprocessor Ethernet Driver....3-21
UDP/IP Protocol Modules....3-21
RARP/ARP Protocol Modules....3-22
BOOTP Protocol Module....3-22
TFTP Protocol Module....3-22
Network Boot Control Module....3-22
Network I/O Error Codes....3-23
Multiprocessor Support....3-23

Multiprocessor Control Register (MPCR) Method ..... 3-23
GCSR Method ..... 3-25
Diagnostic Facilities ..... 3-26
Manufacturing Test Process ..... 3-26

## CHAPTER 4 Using the 172Bug Debugger

This Chapter Covers ..... 4-1
Entering Debugger Command Lines ..... 4-1
Terminal Input/Output Control ..... 4-1
Debugger Command Syntax ..... 4-3
Syntactic Variables ..... 4-3
Expression as a Parameter ..... 4-3
Address as a Parameter ..... 4-5
Address Formats ..... 4-5
Offset Registers ..... 4-7
Port Numbers ..... 4-9
Entering and Debugging Programs ..... 4-10
Creating a Program with the Assembler/Disassembler ..... 4-10
Downloading an S-Record Object File ..... 4-10
Read the Program from Disk ..... 4-11
Calling System Utilities from User Programs ..... 4-11
Preserving the Debugger Operating Environment ..... 4-12
172Bug Vector Table and Workspace ..... 4-12
Examples ..... 4-13
Hardware Functions ..... 4-13
Exception Vectors Used by 172Bug ..... 4-14
Exception Vector Tables ..... 4-15
Using 172Bug Target Vector Table ..... 4-15
Creating a New Vector Table ..... 4-16
Floating Point Support ..... 4-18
Single Precision Real ..... 4-19
Double Precision Real ..... 4-20
Scientific Notation ..... 4-21
The 172Bug Debugger Command Set ..... 4-22

## APPENDIX A Configure and Environment Commands

Configure Board Information Block ..... A-1
Set Environment to Bug/Operating System ..... A-3
Configuring the IndustryPacks ..... A-16

## APPENDIX B Serial Interconnections


Introduction ..... B-1
EIA-232-D Connections ..... B-1
Interface Characteristics ..... B-4
EIA-530 Connections ..... B-5
Interface Characteristics ..... B-7
EIA-485/EIA-422 Connections ..... B-8
Interface Characteristics ..... B-10
Proper Grounding ..... B-11


## APPENDIX C IndustryPack Interconnections


Introduction ..... C-1


## APPENDIX D Disk/Tape Controller Data


Controller Modules Supported ..... D-1
Default Configurations ..... D-2
IOT Command Parameters ..... D-5


## APPENDIX E Network Controller Data


Network Controller Modules Supported ..... E-1


## APPENDIX F Troubleshooting the MVME172: Solving Startup Problems

# List of Figures

Figure 1-1. MVME172 Block Diagram ..... 1-14
Figure 2-1. MVME172 Switches, Headers, Connectors, and LEDs ..... 2-4
Figure 2-2. Serial Interface Module, Connector Side ..... 2-6
Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 1 of 6) ..... 2-22
Figure 2-4. MVME172 EIA-530 Connections (Sheet 1 of 2) ..... 2-28
Figure 2-5. MVME172 EIA-232-D Connections, MVME712A/AM/-12/-13 (Sheet 1 of 4) 2-30
Figure 2-6. MVME172 EIA-485/EIA-422 Connections ..... 2-34

# List of Tables

Table 1-1. MVME172 Specifications ..... 1-9
Table 1-2. Local Bus Arbitration Priority ..... 1-30
Table 1-3. DRAM Performance ..... 1-30
Table 1-4. Local Bus Memory Map ..... 1-34
Table 1-5. Local Bus I/O Devices Memory Map ..... 1-36
Table 2-1. Jumper-Configurable Options ..... 2-3
Table 2-2. Serial Interface Module Part Numbers ..... 2-6
Table 2-3. J2 Snoop Control Encoding ..... 2-11
Table 4-1. Debugger Address Parameter Formats ..... 4-5
Table 4-2. Exception Vectors Used by 172Bug ..... 4-14
Table 4-3. Debugger Commands ..... 4-22
Table A-1. ENV Command Parameters ..... A-4
Table B-1. EIA-232-D Interconnections ..... B-2
Table B-2. EIA-232-D Interface Transmitter Characteristics ..... B-4
Table B-3. EIA-232-D Interface Receiver Characteristics ..... B-4
Table B-4. Serial Port B EIA-530 Interconnect Signals ..... B-5
Table B-5. EIA-530 Interface Transmitter Characteristics ..... B-7
Table B-6. EIA-530 Interface Receiver Characteristics ..... B-8
Table B-7. Serial Port B EIA-485/EIA-422 Interconnect Signals ..... B-9
Table B-8. EIA-485 Interface Transmitter (Generator) Characteristics ..... B-10
Table B-9. EIA-485 Interface Receiver (Load) Characteristics ..... B-11
Table F-1. Troubleshooting Steps ..... F-1

# Board Level Hardware Description

1

## Introduction

This chapter describes the board level hardware features of the MVME172 VME Embedded Controller. The chapter is organized with a board-level overview and features list in this introduction, followed by a more detailed hardware functional description. Front panel switches and indicators are included in the detailed hardware functional description. The chapter closes with some general memory maps.

All programmable registers in the MVME172 that reside in ASICs are covered in the *MVME172 VME Embedded Controller Programmer's Reference Guide*.

### Overview

The MVME172 is based on the MC68060 or MC68LC060 microprocessor. Various versions of the MVME172 have 4, 8, or 16MB of unprotected DRAM, 8KB of SRAM (with battery backup), time-of-day clock (with battery backup), Ethernet transceiver interface, two serial ports with EIA-232-D or EIA-530 or EIA-485/422 interface, six tick timers, watchdog timer, a PROM socket, 2MB Flash memory (one Flash device), four IndustryPack (IP) interfaces with DMA, SCSI bus interface with DMA, VMEbus controller, and 512 KB of SRAM with battery backup.

The I/O on the MVME172 is connected to the VMEbus P2 connector. The main board is connected through a P2 transition board and cables to the transition boards. The MVME172 supports the transition boards MVME712-12, MVME712-13, MVME712M, MVME712A, MVME712AM, and MVME712B (referred to in this manual as MVME712*x*, unless separately specified). The MVME712*x* transition boards provide configuration headers and industry-standard connectors for I/O devices.

The I/O connection for the serial ports on the MVME172 is also implemented with two DB25 front panel I/O connectors. The MVME712 series transition boards were designed to support the MVME167 boards, but can be used on the MVME172 if you take some special precautions. Refer to the *Serial Communications Interface* section for more information. These transition boards provide configuration headers, serial port drivers and industry-standard connectors for the I/O devices.

The following ASICS are used on the MVME172:

- **VMEchip2.** (VMEbus interface). Provides two tick timers, a watchdog timer, programmable map decoders for the master and slave interfaces, and a VMEbus to/from local bus DMA controller, a VMEbus to/from local bus non-DMA programmed access interface, a VMEbus interrupter, a VMEbus system controller, a VMEbus interrupt handler, and a VMEbus requester.

  Processor-to-VMEbus transfers are D8, D16, or D32. VMEchip2 DMA transfers to the VMEbus, however, are D16, D32, D16/BLT, D32/BLT, or D64/MBLT.

- **MC2 chip.** Provides four tick timers, the interface to the LAN chip, SCSI chip, serial port chip, BBRAM, EPROM/Flash, parity-DRAM and SRAM.

- **IndustryPack Interface Controller (IP2).** The IP2 provides control and status information for up to two singlewide IPs or one doublewide IP that can be plugged into the MVME172 main board.

## Related Documentation

This manual includes installation instructions, jumper configuration information, memory maps, debugger/monitor commands, and any other information needed to start up the board.

If you wish to develop your own applications or need more detailed information about your MVME172FX VME Embedded Controller, you may purchase the additional documentation listed on the following pages.

The publications listed below are referenced in this document. To purchase manuals you may contact Motorola in these ways:

- Through your local Motorola sales office
- By contacting the Literature Center via phone or fax at the numbers listed under Product Literature at MCG's World Wide Web site, http://www.mcg.mot.com/literature.

| Motorola Publication Number | Description |
|---|---|
| 68KBUG1/D 68KBUG2/D | Debugging Package for Motorola 68K CISC CPUs User's Manual (Parts 1 and 2) |
| VME172A/PG | MVME172 VME Embedded Controller Programmer's Reference Guide |
| V172DIAA/UM | MVME172Bug Diagnostics Manual |
| SBCSCSI/D | Single Board Computers SCSI Software User's Manual |
| SIMM09A/IH | SIMM09 Serial Interface Module Installation Guide |
| VME712MA/IH | MVME712M Transition Module and P2 Adapter Board Installation and Use |
| MVME712A/D | MVME712-12, MVME712-13, MVME712A, MVME712AM, and MVME712B Transition Modules and LCP2 Adapter Board User's Manual |

Each Motorola Computer Group manual publication number is suffixed with characters that represent the revision level of the document, such as /IH2 (second revision).

To obtain the most up-to-date product information in PDF or HTML format, visit our Web site at http://www.mcg.mot.com/literature.

## Other Applicable Motorola Publications

The following publications are applicable to the MVME172 and may provide additional helpful information. They may be purchased from:

Motorola Semiconductor Products Sector
Literature Distribution
Telephone: 1-800-441-2447

FAX: (602) 994-6430 or (303) 675-2150

E-mail: ldcformotorola@hibbertco.com

| Motorola Publication Number | Description |
|---|---|
| M68000FR | M68000 Family Reference Manual |
| M68060UM | MC68060 Microprocessors User's Manual |
| M68040UM | MC68040 Microprocessors User's Manual |

## Applicable Non-Motorola Publications

The following non-Motorola publications are also available from the sources indicated.

| Document Title | Source |
|---|---|
| VME64 Specification, order number ANSI/VITA 1-1994<br><br>**Note:** An earlier version of the VME specification is available as:<br><br>Versatile Backplane Bus: VMEbus, ANSI/IEEE Std 1014-1987 (VMEbus Specification) (This is also Microprocessor System Bus for 1 to 4 Byte Data, IEC 821 BUS) | VITA (VMEbus International Trade Association)<br>7825 E. Gelding Dr., Ste. 104<br>Scottsdale, AZ 85260-3415 |
| ANSI Small Computer System Interface-2 (SCSI-2), Draft Document X3.131-198X, Revision 10c | Global Engineering Documents<br>15 Inverness Way East<br>Englewood, CO 80112-5704 |

| Document Title | Source |
|---|---|
| 82596CA Local Area Network Coprocessor Data Sheet, order number 290218; and 82596 User's Manual, order number 296853 | Intel Corporation<br>Literature Sales<br>P.O. Box 58130<br>Santa Clara, CA 95052-8130 |
| 28F016SA Flash Memory Data Sheet, order number 290435 | Intel Corporation<br>Literature Sales<br>P.O. Box 7641,<br>Mt. Prospect, IL 60056-7641 |
| NCR 53C710 SCSI I/O Processor Data Manual, order number NCR53C710DM | NCR Corporation<br>Microelectronics Products Division<br>1635 Aeroplaza Dr.<br>Colorado Springs, CO 80916 |
| NCR 53C710 SCSI I/O Processor Programmer's Guide, order number NCR53C710PG | |
| MK48T58(B) Timekeeper ™ and 8Kx8 Zeropower ™ RAM data sheet in Static RAMs Databook, order number DBSRAM71 | SGS-THOMSON<br>Microelectronics Group<br>Marketing Headquarters<br>1000 East Bell Rd.<br>Phoenix, AZ 85022-2699 |
| IndustryPack Logic Interface Specification, Revision 1.0, order number ANSI/VITA 4-1995 | VITA (VMEbus International Trade Association)<br>7825 E. Gelding Dr., Ste. 104<br>Scottsdale, AZ 85260-3415 |
| Z85230 Serial Communications Controller Data Sheet | Zilog Inc.<br>210 Hacienda Ave.<br>Campbell, CA 95008-6609 |

# MCG Customer Services

The Motorola Computer Group Customer Services organization provides numerous services that support the needs of our OEM customers throughout the qualification, development, deployment, and continued service phases of their product life cycles. Specific areas of support include:

- Helping you identify and address your unique needs for providing outstanding products to your customers
- Extended warranties
- 24 X 7 access to the high-level technical assistance that is crucial for mission critical applications around the world
- 24 X 7 access to the latest technical information on MCG products, including known problems and a solutions database
- Customized training available at the MCG campus or at any of your sites across the world
- Customized documentation and 24 X 7 Internet access to product documentation
- Services Central, a one-stop information source about customer services -- program content, pricing, and availability

For information on what services are available, or to purchase a support contract, call us at Services Central, at 1-800-624-6745 or 602-438-5875, or visit our website at http://www.mcg.mot.com/support.

## Requirements

These boards are designed to conform to the requirements of the following documents:

- VME64 Specification, VITA
- EIA-232-D Serial Interface Specification, EIA
- SCSI Specification, ANSI
- IndustryPack Specification, VITA

## Features

- The MC68060 has a clock frequency of 60 MHz; the MC68LC060 has a clock frequency of 64 MHz
- 4, 8, or 16MB of shared DRAM with no protection
- 512KB of SRAM with battery backup
- One JEDEC standard 32-pin PLCC EPROM socket
- One Intel 28F016SA 2M x 8 Flash memory device (2MB Flash memory total)
- 8K by 8 Non-Volatile RAM and time of day clock with battery backup
- Four 32-bit Tick Timers (in the MC2 chip ASIC) for periodic interrupts
- Two 32-bit Tick Timers (in the VMEchip2 ASIC) for periodic interrupts
- Watchdog timer
- Eight software interrupts (for MVME172 versions that have the VMEchip2)
- I/O

- Two serial ports (one EIA-232-D DCE; one EIA-232-D DCE/DTE or EIA-530 DCE/DTE or EIA-422 DCE/DTE or EIA-485)
- Serial port controller (Zilog Z85230)
- Optional Small Computer Systems Interface (SCSI) bus interface with 32-bit local bus burst Direct Memory Access (DMA) (NCR 53C710 controller)
- Optional LAN Ethernet transceiver interface with 32-bit local bus DMA (Intel 82596CA controller)
- Four IndustryPack interfaces with DMA

❑ VMEbus interface (boards may be special-ordered without the VMEbus interface)

- VMEbus system controller functions
- VMEbus interface to local bus (A24/A32, D8/D16/D32 (D8/D16/D32/D64 BLT) (BLT = Block Transfer)
- Local bus to VMEbus interface (A16/A24/A32, D8/D16/D32)
- VMEbus interrupter
- VMEbus interrupt handler
- Global CSR for interprocessor communications
- DMA for fast local memory - VMEbus transfers (A16/A24/A32, D16/D32 (D16/D32/D64 BLT)

❑ Switches and Light-Emitting Diodes (LEDs)

- Two pushbutton switches (ABORT and RESET)
- Eight LEDs (FAIL, STAT, RUN, SCON, LAN, FUSE, SCSI, and VME)

## Specifications

General specifications for the MVME172 are listed in Table 1-1.

**Table 1-1. MVME172 Specifications**

| Characteristics | Specifications |
|---|---|
| Power requirements (with PROM; without IPs) | +5 Vdc (± 5%), 3.5 A typical, 4.5 A max.<br>+12 Vdc (± 5%), 100 mA (max.)<br>-12 Vdc (± 5%), 100 mA (max.) |
| Operating temperature | 0° to 70° C exit air with forced air cooling* |
| Storage temperature | -40° to +85° C |
| Relative humidity | 5% to 90% (noncondensing) |
| Physical dimensions<br>PC board with mezzanine module only<br>Height<br>Depth<br>Thickness<br>PC board with connectors and front panel<br>Height<br>Depth<br>Thickness | Double-high VMEboard<br><br><br><br>9.187 inches (233.35 mm)<br>6.299 inches (160.00 mm)<br>0.662 inch (16.77 mm)<br><br><br><br>10.309 inches (261.85 mm)<br>7.4 inches (188 mm)<br>0.80 inch (20.32 mm) |

*Refer to *Cooling Requirements* and *Special Considerations for Elevated Temperature Operation*.

## Cooling Requirements

The Motorola MVME172 VME Embedded Controller is specified, designed, and tested to operate reliably with an incoming air temperature range from 0° to 55° C (32° to 131° F) with forced air cooling at a velocity typically achievable by using a 100 CFM axial fan. Temperature qualification is performed in a standard Motorola VMEsystem chassis. Twenty-five-watt load boards are inserted in two card slots, one on each side, adjacent to the board under test, to simulate a high power density system configuration. An assembly of three axial fans, rated at 100 CFM per fan, is placed directly under the VME card cage. The incoming air temperature is measured between the fan assembly and the card cage, where the incoming airstream first encounters the controller under test. Test software is executed as the controller is subjected to ambient temperature variations. Case temperatures of critical, high power density integrated circuits are monitored to ensure component vendors specifications are not exceeded.

While the exact amount of airflow required for cooling depends on the ambient air temperature and the type, number, and location of boards and other heat sources, adequate cooling can usually be achieved with 10 CFM and 490 LFM flowing over the controller. Less airflow is required to cool the controller in environments having lower maximum ambients. Under more favorable thermal conditions, it may be possible to operate the controller reliably at higher than 55° C with increased airflow. It is important to note that there are several factors, in addition to the rated CFM of the air mover, which determine the actual volume and speed of air flowing over the controller.

## Special Considerations for Elevated Temperature Operation

The following information is for users whose applications for the MVME172 may subject it to high temperatures.

The MVME172 uses commercial grade devices. Therefore, it can operate in an environment with ambient air temperature from 0° C to 70° C. There are many factors that affect the ambient temperature seen by components

on the MVME172: inlet air temperature; air flow characteristics; number, types, and locations of IndustryPack (IP) modules; power dissipation of adjacent boards in the system, etc.

A temperature profile of the MVME172 (MVME172-513) was developed in an MVME945 12-slot VME chassis. This board was loaded with one GreenSpring IP-Dual P/T module (position a) and three GreenSpring IP-488 modules (positions b, c, and d). One twenty-five watt load board was installed adjacent to each side of the board under test. The exit air velocity was approximately 200 LFM between the MVME172 and the IP-Dual P/T module. Under these circumstances, a 10° C rise between the inlet and exit air was observed. At 70° C exit air temperature (60° C inlet air), the junction temperatures of devices on the MVME172 were calculated (from the measured case temperatures) and did not exceed 100° C.



**Caution**

For elevated temperature operation, the user must perform similar measurements and calculations to determine what operating margin exists for any specific environment.

The following are some steps that the user can take to help make elevated temperature operation possible:

1. Position the MVME172 board in the chassis for maximum airflow over the component side of the board.
2. Avoid placing boards with high power dissipation adjacent to the MVME172.
3. Use low power IP modules only. The preferred locations for IP modules are position *a* (J4 and J5) and position *d* (J29 and J30).

## FCC Compliance

The MVME172 was tested *without* IndustryPacks in an FCC-compliant chassis and meets the requirements for Class A equipment. FCC compliance was achieved under the following conditions:

1. Shielded cables on all external I/O ports.
2. Cable shields connected to earth ground via metal shell connectors bonded to a conductive module front panel.
3. Conductive chassis rails connected to earth ground. This provides the path for connecting shields to earth ground.
4. Front panel screws properly tightened.

For minimum RF emissions, it is essential that the conditions above be implemented. Failure to do so could compromise the FCC compliance of the equipment containing the module.

## Manual Terminology

Throughout this manual, a convention is used which precedes data and address parameters by a character identifying the numeric format as follows:

| | | |
|---|---|---|
| $ | dollar | specifies a hexadecimal character |
| % | percent | specifies a binary number |
| & | ampersand | specifies a decimal number |

For example, “12” is the decimal number twelve, and "$12" is the decimal number eighteen.

Unless otherwise specified, all address references are in hexadecimal.

An asterisk (*) following the signal name for signals which are level significant denotes that the signal is true or valid when the signal is low.

An asterisk (*) following the signal name for signals which are edge significant denotes that the actions initiated by that signal occur on high to low transition.

In this manual, assertion and negation are used to specify forcing a signal to a particular state. In particular, assertion and assert refer to a signal that is active or true; negation and negate indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent.

Data and address sizes are defined as follows:

- A byte is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- A word is 16 bits, numbered 0 through 15, with bit 0 being the least significant.
- A longword is 32 bits, numbered 0 through 31, with bit 0 being the least significant.

The terms control bit and status bit are used extensively in this document.

The term control bit is used to describe a bit in a register that can be set and cleared under software control. The term true is used to indicate that a bit is in the state that enables the function it controls. The term false is used to indicate that the bit is in the state that disables the function it controls. In all tables, the terms 0 and 1 are used to describe the actual value that should be written to the bit, or the value that it yields when read.

The term status bit is used to describe a bit in a register that reflects a specific condition. The status bit is read by software to determine operational or exception conditions.

## Block Diagram

Figure 1-1 is a general block diagram of the MVME172.

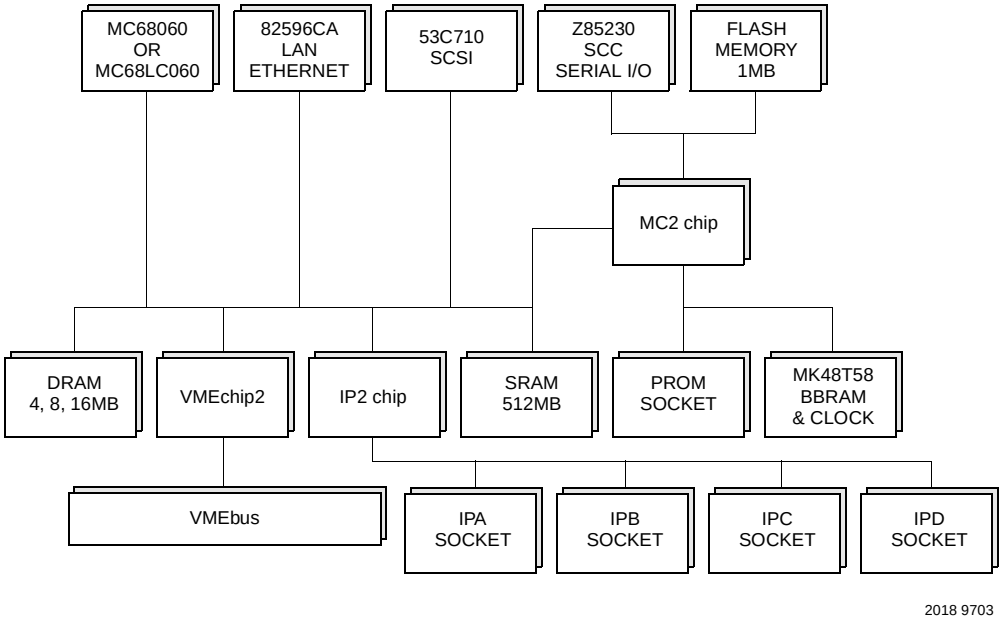



**Figure 1-1. MVME172 Block Diagram**

# Functional Description

This section contains a functional description of the major blocks on the MVME172.

## Front Panel Switches and Indicators

There are switches and LEDs on the front panel of the MVME172. The switches are RESET and ABORT. The RESET switch resets all onboard devices and drives SYSRESET* if the board is system controller. The RESET switch may be disabled by software.

When enabled by software, the ABORT switch generates an interrupt at a user-programmable level. It is normally used to abort program execution and return to the debugger.

There are eight LEDs on the MVME172 front panel: FAIL, STAT, RUN, SCON, LAN, FUSE (LAN power), SCSI, and VME.

The red FAIL LED (part of DS1) lights when the BRDFAIL signal line is active.

The MC68060 status lines are decoded, on the MVME172, to drive the yellow STAT (status) LED (part of DS1). In this case, a halt condition from the processor lights the LED.

The green RUN LED (part of DS2) lights when the local bus TIP* signal line is low. This indicates one of the local bus masters is executing a local bus cycle.

The green SCON LED (part of DS2) lights when the VMEchip2 is the VMEbus system controller.

The green LAN LED (part of DS3) lights when the LAN chip is local bus master.

The green FUSE LED lights when the solid-state fuses for the IP ±12V, IP +5V, and LAN transceiver +12V are functional.

The green SCSI LED (part of DS4) lights when the SCSI chip is local bus master.

The green VME LED (part of DS4) lights when the board is using the VMEbus (VMEbus AS* is asserted by the VMEchip2) or when the board is accessed by the VMEbus (VMEchip2 is the local bus master).

## Data Bus Structure

The local data bus on the MVME172 is a 32-bit synchronous bus that is based on the MC68060 bus, and which supports burst transfers and snooping. The various local bus master and slave devices use the local bus to communicate. The local bus is arbitrated by priority type arbiter and the priority of the local bus masters from highest to lowest is: 82596CA LAN, IndustryPack DMA, 53C710 SCSI, VMEbus, and MPU. Generally speaking, any master can access any slave; however, not all combinations pass the common sense test. Refer to the *MVME172 VME Embedded Controller Programmer's Reference Guide* and to the user's guide for each device to determine its port size, data bus connection, and any restrictions that apply when accessing the device.

## MC68060 or MC68LC060 MPU

The MVME172 may be ordered with an MC68060 or MC68LC060 microprocessor.

The MC68060 has on-chip instruction and data caches and a floating point processor. (The floating point coprocessor is the major difference between the MC68060 and MC68LC060.) Refer to the M68060 user's manual for more information.

### MC68*xx*060 Cache

The MVME172 local bus masters (VMEchip2, MC68*xx*060, 53C710 SCSI controller, and 82596CA Ethernet controller) have programmable control of the snoop/caching mode. The IP DMA local bus master's snoop control function is controlled by a jumper at J2. J2 controls the value of the snoop control signal for all IP DMA transfers. This includes the IP DMA which is executed when the DMA control registers are updated while the

IP DMA is operating in the command chaining mode. The MVME172 local bus slaves which support MC68*xx*060 bus snooping are defined in the Local Bus Memory Map table later in this chapter.

## No-VMEbus-Interface Option

The MVME172 can be operated as an embedded controller without the VMEbus interface. For this option, the VMEchip2 and the VMEbus buffers are not populated. Also, the bus grant daisy chain and the interrupt acknowledge daisy chain have zero-ohm bypass resistors installed.

To support this feature, certain logic in the VMEchip2 has been duplicated in the MC2 chip. This logic is inhibited in the MC2 chip if the VMEchip2 is present. The enables for these functions are controlled by software and MC2 chip hardware initialization.

Note that MVME172 models ordered without the VMEbus interface are shipped with Flash memory blank (the factory uses the VMEbus to program the Flash memory with debugger code). To use the 172Bug package, MVME172Bug, be sure that jumper header J28 is configured for the EPROM memory map. Refer to *Chapter 3* and *Chapter 4* for further details.

Contact your local Motorola sales office for ordering information.

## Memory Options

The following memory options are used on the different versions of MVME172 boards.

### DRAM Options

The MVME172 implementation includes a 4MB, 8MB, or 16MB DRAM option. The DRAM architecture is non-interleaved for 4MB and 8MB, while the 16MB architecture is interleaved. The 4MB DRAM option is located entirely on the MVME172 base board; the 8MB and 16MB options include 4MB or 12MB on a mezzanine module. The DRAM is not parity protected.

The following table defines the combinations of base board and mezzanine population options used for the MVME172 series of modules.

| Mezzanine MB | Base Board MB | Available DRAM | Interleaved | MC2 chip Size (@ register offset $25) |
|---|---|---|---|---|
| 0 | 4 | 4 | N | 100 |
| 4 (bank 2) | 4 | 8 | N | 101 |
| 12 (banks 2, 3, and 4) | 4 | 16 | Y | 111 |

## SRAM Options

The MVME172 implementation includes a 512KB SRAM option. SRAM architecture is single non-interleaved. SRAM performance is specified in the section on the SRAM Memory Controller in the MC2 chip Programming Model in the *MVME172 VME Embedded Controller Programmer's Reference Guide*. A battery supplies VCC to the SRAMs when main power is removed. The worst case elapsed time for battery protection is 200 days.

The SRAM arrays are not parity protected.

The MVME172 SRAM battery backup function is provided by a Dallas DS1210S. The DS1210S supports primary and secondary power sources. When the main board power fails, the DS1210S selects the source with the highest voltage. If one source should fail, the DS1210S switches to the redundant source. Each time the board is powered, the DS1210S checks power sources and if the voltage of the backup sources is less than two volts, the second memory cycle is blocked. This allows software to provide an early warning to avoid data loss. Because the DS1210S may block the second access, the software should do at least two accesses before relying on the data.

The MVME172 provides jumpers (on J22) that allow either power source of the DS1210S to be connected to the VMEbus +5V STDBY pin or to one cell of the onboard battery. For example, the primary system backup source may be a battery connected to the VMEbus +5V STDBY pin and the secondary source may be the onboard battery. If the system source should fail or the board is removed from the chassis, the onboard battery takes over. Refer to *Chapter 2* for the jumper configurations.

**Caution**

For proper operation of the SRAM, some jumper combinations must be installed on the Backup Power Source Select Header (J22). If one of the jumpers is used to select the battery, the battery must be installed on the MVME172. The SRAM may malfunction if inputs to the DS1210S are left unconnected.

The SRAM is controlled by the MC2 chip, and the access time is programmable. Refer to the MC2 chip description in the *MVME172 VME Embedded Controller Programmer's Reference Guide* for more detail.

## About the Battery

The power source for the onboard SRAM is a RAYOVAC FB1225 battery with two BR1225 type lithium cells which is socketed for easy removal and replacement. A small capacitor is provided to allow the battery to be quickly replaced without data loss.

The lifetime of the battery is very dependent on the ambient temperature of the board and the power-on duty cycle. The lithium battery supplied on the MVME172 should provide at least two years of backup time with the board powered off and with an ambient temperature of 40° C. If the power-on duty cycle is 50% (the board is powered on half of the time), the battery lifetime is four years. At lower ambient temperatures the backup time is greatly extended and may approach the shelf life of the battery.

When a board is stored, the battery should be disconnected to prolong battery life. This is especially important at high ambient temperatures. The MVME172 is shipped with the batteries disconnected (i.e., with VMEbus +5V standby voltage selected as both primary and secondary power

source). If you intend to use the battery as a power source, whether primary or secondary, it is necessary to reconfigure the jumpers on J22 before installing the module. Refer to *SRAM Battery Backup Source Select Header (J22)* for available jumper configurations.

The power leads from the battery are exposed on the solder side of the board, therefore the board should not be placed on a conductive surface or stored in a conductive bag unless the battery is removed.

**Warning**

Lithium batteries incorporate inflammable materials such as lithium and organic solvents. If lithium batteries are mistreated or handled incorrectly, they may burst open and ignite, possibly resulting in injury and/or fire. When dealing with lithium batteries, carefully follow the precautions listed below in order to prevent accidents.

- Do not short circuit.
- Do not disassemble, deform, or apply excessive pressure.
- Do not heat or incinerate.
- Do not apply solder directly.
- Do not use different models, or new and old batteries together.
- Do not charge.
- Always check proper polarity.

To remove the battery from the module, carefully pull the battery from the socket.

Before installing a new battery, ensure that the battery pins are clean. Note the battery polarity and press the battery into the socket. When the battery is in the socket, no soldering is required.

## EPROM and Flash Memory

The MVME172 implementation includes 2MB of Flash memory (an 8-Mbit Flash device organized as a 2M x 8). For information on programming Flash, refer to the Intel documents listed in *Related Documentation*.

The Flash write enable is controlled by a bit in the Flash Access Time Control Register in the MC2 chip. Refer to the *MVME172 Embedded Controller Programmer's Reference Guide* for more detail.

The EPROM location is a standard JEDEC 32-pin PLCC capable of 4 Mbit densities organized as a 512KB x 8 device. Depending on a jumper setting (GPI3, pins 9-10 on J28), the MC68*xx*060 reset code can be fetched from either the Flash (GPI3 installed) or EPROM (GPI3 removed).

Note that MVME172 models ordered without the VMEbus interface are shipped with Flash memory blank (the factory uses the VMEbus to program the Flash memory with debugger code). To use the 172Bug package, MVME172Bug, be sure that jumper header J28 is configured for the EPROM memory map. Refer to *Chapter 3* and *Chapter 4* for further details.

## Battery Backed Up RAM and Clock

The MK48T58 RAM and clock chip is used on the MVME172. This chip provides a time-of-day clock, oscillator, crystal, power failure detection, 8KB of RAM, and a battery in one 28-pin package. The clock provides seconds, minutes, hours, day, date, month, and year in BCD 24-hour format. Corrections for 28- day, 29-day (leap year), and 30-day months are automatically made. No interrupts are generated by the clock. The MK48T58 is an 8 bit device; however, the interface provided by the MC2 chip supports 8-bit, 16-bit, and 32-bit accesses to the MK48T58. Refer to the MC2 chip description in the *MVME172 VME Embedded Controller Programmer's Reference Guide* and to the MK48T58 data sheet for detailed programming and battery life information.

## VMEbus Interface and VMEchip2

The local bus to VMEbus interface and the VMEbus to local bus interface are provided by the optional VMEchip2. The VMEchip2 can also provide the VMEbus system controller functions. Refer to the VMEchip2 description in the *MVME172 VME Embedded Controller Programmer's Reference Guide* for detailed programming information.

Note that the ABORT switch logic in the VMEchip2 is not used. The GPI inputs to the VMEchip2 which are located at $FFF40088 bits 7-0 are not used. The ABORT switch interrupt is integrated into the MC2 chip ASIC at location $FFF42043. The GPI inputs are integrated into the MC2 chip ASIC at location $FFF4202C bits 23-16.

# I/O Interfaces

The MVME172 provides onboard I/O for many system applications. The I/O functions include serial ports, IndustryPack (IP) interfaces, optional LAN Ethernet transceiver interface, and optional SCSI mass storage interface.

## Serial Communications Interface

The MVME172 uses a Zilog Z85230 serial communications controller to implement the two serial communications interfaces. Each interface supports CTS, DCD, RTS, and DTR control signals; as well as the TxD and RxD transmit/receive data signals, and TxC/RxC synchronous clock signals.

The Z85230 supports synchronous (SDLC/HDLC) and asynchronous protocols. The MVME172 hardware supports asynchronous serial baud rates of 110b/s to 38.4Kb/s.

The Z85230 supplies an interrupt vector during interrupt acknowledge cycles. The vector is modified based upon the interrupt source within the Z85230. Interrupt request levels are programmed via the MC2 chip. (One MC2 chip can handle up to four Z85230 chips.) Refer to the Z85230 data sheet listed in this chapter, and to the MC2 chip Programming Model in the *MVME172 VME Embedded Controller Programmer's Reference Guide*, for information.

### MVME172 Serial Port 1

The A port of the Z85230 is interfaced as DCE (data circuit-terminating equipment) with the EIA-232-D interface and is routed to:

- The DB-25 connector marked SERIAL PORT 1/CONSOLE on the front panel of the **MVME172**. SERIAL PORT 1/CONSOLE is an EIA-232-D DCE port.

**Note** This port can be connected to the TX and RX clocks which may be present on the DB-25 connector. These connections are made via jumper header J16 on the MVME172 board. (The TxC and RxC clock lines are not available on the MVME712*x* transition modules.)

- One of the following output connectors on the MVME712*x* transition module:

  **MVME712M**: The DB-25 connector marked SERIAL PORT 2 on the front panel. SERIAL PORT 2 can be configured as an EIA-232-D DTE or DCE port, via jumper headers J16 and J17.

  **MVME712A or MVME712-12**: The DB-9 connector marked SERIAL PORT 2 on the front panel. SERIAL PORT 2 is hardwired as an EIA-232-D DTE port.

  **MVME712AM or MVME712-13**: The DB-9 connector marked SERIAL PORT 2 ***OR*** the RJ-11 jack on the front panel. SERIAL PORT 2 is hardwired as EIA-232-D DTE; the RJ-11 jack utilizes the built-in modem. Setting the jumper headers J16 and J17 on the MVME712AM/-13 configures the output as EIA-232-D DTE at SERIAL PORT 2 or as a modem at the RJ-11 jack.

Figure 2-3 (sheets 1 and 2) in Chapter 2 illustrates the two configurations available for Port A when the MVME172 is used with an MVME712M. Figure 2-5 (sheets 1 and 2) shows the two configurations available for Port A when the MVME172 is used with an MVME712A/AM/-12/-13.

### MVME172 Serial Port 2

The configuration of the B port of the Z85230 is determined via a Serial Interface Module (SIMM) which is installed at connector J15 on the MVME172 board. There are five SIMMs available:

SIMM05 -- DTE with EIA-232-D interface
SIMM06 -- DCE with EIA-232-D interface
SIMM07 -- DTE with EIA-530 interface
SIMM08 -- DCE with EIA-530 interface
SIMM09 -- EIA-485 interface, or DCE or DTE with EIA-422 interface

Port B is routed, via the SIMM, to:

- The DB-25 connector marked SERIAL PORT 2 on the front panel of the **MVME172**. SERIAL PORT 2 will be an EIA-232-D DCE or DTE port, or an EIA-530 DCE or DTE port, or an EIA-485 port, or an EIA-422 DCE or DTE port, depending upon which SIMM is installed.

**Note** This port can be connected to the TX and RX clocks which may be present on the DB-25 connector. These connections are made via jumper header J17 on the MVME172 board. (The TxC and RxC clock lines are available at the MVME712M SERIAL PORT 4 via header J15, but are not available on the other MVME712*x* transition modules.)

- One of the following output connectors on the MVME712*x* transition module:

  **MVME712M**: The DB-25 connector marked SERIAL PORT 4 on the front panel. SERIAL PORT 4 can be configured as an EIA-232-D DTE or DCE port, via the jumper headers J18 and J19 on the MVME712M.

  **MVME712A, AM, -12, or -13**: The DB-9 connector marked SERIAL PORT 4 on the front panel. SERIAL PORT 4 is hard-wired as an EIA-232-D DTE port.

Figure 2-3 (sheets 3 through 6) in Chapter 2 illustrates the four configurations available for Port B when the MVME172 is used with an MVME712M. Note that the port configurations shown in Figure 2-3 sheets 5 and 6 are not recommended for synchronous applications because of the incorrect clock direction. Figure 2-4 (sheets 1 and 2) shows an MVME172 with the two configurations available with EIA-530 SIMMs. Figure 2-5 (sheets 3 and 4) shows the two configurations available for Port B when the MVME172 is used with an MVME712A/AM/-12/-13. Figure 2-6 shows an MVME172 with the configuration available with the EIA-485 /EIA-422 SIMM.

**Caution**

Do not simultaneously connect serial data devices to the equivalent ports on the MVME712 series transition module and the MVME172 front panel. This could result in simultaneous transmission of conflicting data.

**Caution**

Do not connect peripheral devices to Port 1, Port 3, or the Centronics printer port on the MVME712x transition module. In the EIA-232-D case, none of these ports are connected to any MVME172 circuits. In the EIA-530 case, attempting to use these ports would produce certain connections with the potential to damage the MVME172 or the peripherals.

**Caution**

When using an EIA-530 SIMM or an EIA-485/EIA-422 SIMM, do not connect the MVME172 to an MVME712*x* board. The EIA-530, EIA-485, and EIA-422 signals are not supported by the P2 adapter and the transition boards.

### IndustryPack (IP) Interfaces

The IP2 chip ASIC on the MVME172 supports four IndustryPack (IP) interfaces: these are accessible from the front panel. The IP2 also includes four DMA channels (one for each IP, or two for each double size IP), 32 or 30MHz (32 MHz for MC68LC060 or 30 MHz for MC68060) or 8MHz IndustryPack clock selection (jumper selectable), and one programmable timebase strobe which is connected to the four interfaces. Refer to the IP2 chip Programming Model in the *MVME172 VME Embedded Controller Programmer's Reference Guide* for details of the IP interface. Refer to *Appendix B* for the pin assignments of the IP connectors.

**Notes** The MVME172 boards do *not* monitor power supply +5 Vdc power and assert IP reset if the power falls too low. Instead, IP reset is handled by the **ENV** command of the 172Bug debugger, as described in Appendix A. The IP reset is also driven active by the power-up reset signal.

Two IP modules plugged into the same MVME172 board can *not* use the Strobe∗ unless J19 is removed. This will disconnect the Strobe* output from the IP2 chip ASIC.

## Optional LAN Ethernet Interface

The MVME172 uses the 82596CA to implement the Ethernet transceiver interface. The 82596CA accesses local RAM using DMA operations to perform its normal functions. Because the 82596CA has small internal buffers and the VMEbus has an undefined latency period, buffer overrun may occur if the DMA is programmed to access the VMEbus. Therefore, the 82596CA should not be programmed to access the VMEbus.

Every MVME172 that has the Ethernet interface is assigned an Ethernet Station Address. The address is $08003E2*xxxxx* where *xxxxx* is the unique 5-nibble number assigned to the board (i.e., every MVME172 has a different value for *xxxxx*).

Each board has an Ethernet Station Address displayed on a label attached to the VMEbus P2 connector. In addition, the six bytes including the Ethernet address are stored in the configuration area of the BBRAM. That is, 08003E2*xxxxx* is stored in the BBRAM. At an address of $FFFC1F2C, the upper four bytes (08003E2*x*) can be read. At an address of $FFFC1F30, the lower two bytes (*xxxx*) can be read. The MVME172 debugger has the capability to retrieve or set the Ethernet address.

If the data in the BBRAM is lost, the user should use the number on the VMEbus P2 connector label to restore it.

The Ethernet transceiver interface is located on the MVME172 main board, and the industry DB15 standard connector is located on the MVME712*x* transition board.

Support functions for the 82596CA are provided by the MC2 chip ASIC. Refer to the 82596CA user's guide for detailed programming information.

### Optional SCSI Interface

The MVME172 may provide for mass storage subsystems through the industry-standard SCSI bus. These subsystems may include hard and floppy disk drives, streaming tape drives, and other mass storage devices. The SCSI interface is implemented using the NCR 53C710 SCSI I/O controller.

Support functions for the 53C710 are provided by the MC2 chip ASIC. Refer to the 53C710 user's guide for detailed programming information.

### SCSI Termination

It is important that the SCSI bus is properly terminated at both ends.

On the MVME172, sockets are provided for the terminators on the P2 adapter board or the LCP2 adapter board. If the SCSI bus ends at the adapter board, then termination resistors must be installed on the adapter board. +5V power to the SCSI bus TERM power line and termination resistors is provided through a fuse located on the adapter board.

## Local Resources

The MVME172 includes many resources for the local processor. These include tick timers, software-programmable hardware interrupts, watchdog timer, and local bus timeout.

### Programmable Tick Timers

Six 32-bit programmable tick timers with 1 µs resolution are provided, two in the VMEchip2 and four in the MC2 chip. The tick timers can be programmed to generate periodic interrupts to the processor. Refer to the VMEchip2 and MC2 chip in the *MVME172 VME Embedded Controller Programmer's Reference Guide* for detailed programming information.

### Watchdog Timer

A watchdog timer function is provided in the VMEchip2 and the MC2 chip. When the watchdog timer is enabled, it must be reset by software within the programmed time or it times out. The watchdog timer can be

programmed to generate a SYSRESET signal, local reset signal, or board fail signal if it times out. Refer to the VMEchip2 and the MC2 chip in the *MVME172 VME Embedded Controller Programmer's Reference Guide* for programming information.

The watchdog timer logic is duplicated in the VMEchip2 and MC2 chip ASICs. Because the watchdog timer function in the VMEchip2 is a super-set of that function in the MC2 chip (system reset function), the timer in the VMEchip2 is used in all cases except for the version of the MVME172 which does not include the VMEbus interface (No VMEbus Interface option).

### Software-Programmable Hardware Interrupts

Eight software-programmable hardware interrupts are provided by the VMEchip2. These interrupts allow software to create a hardware interrupt.

### Local Bus Timeout

The MVME172 provides a timeout function in the VMEchip2 and the MC2 chip for the local bus. When the timer is enabled and a local bus access times out, a Transfer Error Acknowledge (TEA) signal is sent to the local bus master. The timeout value is selectable by software for 8 µsec, 64 µsec, 256 µsec, or infinity. The local bus timer does not operate during VMEbus bound cycles. VMEbus bound cycles are timed by the VMEbus access timer and the VMEbus global timer.

The access timer logic is duplicated in the VMEchip2 and MC2 chip ASICs. Because the local bus timer in the VMEchip2 can detect an offboard access and the MC2 chip local bus timer cannot, the timer in the VMEchip2 is used in all cases except for the version of the MVME172 which does not include the VMEbus interface (No-VMEbus-Interface option).

## Local Bus Arbiter

The local bus arbiter implements a fixed priority which is described in the following table.

**Table 1-2. Local Bus Arbitration Priority**

| Device | Priority | Note |
|---|---|---|
| LAN | 0 | Highest |
| IP DMA | 1 | ... |
| SCSI | 2 | ... |
| VMEbus | 3 | Next Lowest |
| MC68060 | 4 | Lowest |

## Timing Performance

This section gives performance information for the MVME172. The MVME172 is designed to operate at 25 MHz or 32 MHz.

### Local Bus to DRAM Cycle Times

The DRAM base address, array size, and device size are programmable. The DRAM controller assumes an interleaved architecture if the DRAM size requires eight physical devices (that is, when memory array size is 16MB and DRAM technology is 16 Mbits per device.)

Parity checking and parity exception action is also programmable. The DRAM array size and device size are initialized in the DRAM Space Size Register. However, parity may not be supported by the installed DRAM.

**Table 1-3. DRAM Performance**

| Clock Budget | Operating Conditions |
|---|---|
| 4,2,2,2 | Non-interleaved, read, 25 MHz |
| 4,1,1,1 | Interleaved, read, 25 MHz |

**Table 1-3. DRAM Performance**

| Clock Budget | Operating Conditions |
|---|---|
| 3,2,2,2 | Write, 25 MHz |
| 5,3,3,3 | Non-interleaved, read, 32 MHz |
| 5,2,2,2 | Interleaved, read, 32 MHz |
| 4,2,2,2 | Write, 32 MHz |

## EPROM/Flash Cycle Times

The EPROM/Flash cycle time is programmable from 3 to 10 bus clocks/byte (4 bytes = 12 to 40). (The actual cycle time may vary depending on the device speed.) The data transfers are 32 bits wide. Refer to the *MVME172 VME Embedded Controller Programmer's Reference Guide*.

## SCSI Transfers

The MVME172 includes an SCSI mass storage bus interface with DMA controller. The SCSI DMA controller uses a FIFO buffer to interface the 8-bit SCSI bus to the 32-bit local bus. The FIFO buffer allows the SCSI DMA controller to efficiently transfer data to the local bus in four longword bursts. This reduces local bus usage by the SCSI device. Refer to the MC2 chip Programming Model in the *MVME172 VME Embedded Controller Programmer's Reference Guide*.

The transfer rate of the DMA controller is 44MB/sec at 32 MHz with parity off and interleaved DRAM and read cycles. Assuming a continuous transfer rate of 5MB/sec on the SCSI bus, 12% of the local bus bandwidth is used by transfers from the SCSI bus.

## LAN DMA Transfers

The MVME172 includes a LAN interface with DMA controller. The LAN DMA controller uses a FIFO buffer to interface the serial LAN bus to the 32-bit local bus. The FIFO buffer allows the LAN DMA controller to efficiently transfer data to the local bus.

The 82596CA does not execute MC68060 compatible burst cycles, therefore the LAN DMA controller does not use burst transfers. DRAM write cycles require 3 clock cycles at 25 MHz or 4 clock cycles at 32 MHz, and read cycles require 5 clock cycles.

The transfer rate of the LAN DMA controller is 20MB/sec at 25 MHz or 32 MHz. Assuming a continuous transfer rate of 1MB/sec on the LAN bus, 5% of the local bus bandwidth is used by transfers from the LAN bus.

## Connectors

The MVME172 has two 96-position DIN connectors: P1 and P2. P1 rows A, B, C, and P2 row B provide the VMEbus interconnection. P2 rows A and C provide the connection to the SCSI bus, serial ports, and Ethernet. The serial ports on the MVME172 are also connected to two 25-pin DB-25 female connectors J18 and J25 on the front panel. The four IPs connect to the MVME172 by four pairs of 50-pin connectors. Four 50-pin connectors behind the front panel are for external connections to IP signals. The memory chip mezzanine board is plugged into two 40-pin connectors.

### Remote Status and Control

The remote status and control connector, J6, is a 20-pin connector located behind the front panel of the MVME172. It provides system designers with flexibility in accessing critical indicator and reset functions. When the MVME172 board is enclosed in a chassis and the front panel is not visible, this connector allows the RESET, ABORT, and LED functions to be extended to the control panel of the system, where they are visible. Alternatively, this allows a system designer to construct a RESET/ABORT/LED panel that can be located remotely from the MVME172.

# Memory Maps

There are two points of view for memory maps: 1) the mapping of all resources as viewed by local bus masters (local bus memory map), and 2) the mapping of onboard resources as viewed by VMEbus Masters (VMEbus memory map).

The memory and I/O maps which are described in the following tables are correct for all local bus masters. There is some address translation capability in the VMEchip2. This allows multiple MVME172s on the same VMEbus with different virtual local bus maps as viewed by different VMEbus masters.

## Local Bus Memory Map

The local bus memory map is split into different address spaces by the transfer type (TT) signals. The local resources respond to the normal access and interrupt acknowledge codes.

## Normal Address Range

The memory map of devices that respond to the normal address range is shown in the following tables. The normal address range is defined by the Transfer Type (TT) signals on the local bus. On the MVME172, Transfer Types 0, 1, and 2 define the normal address range. Table 1-4 is the entire map from $00000000 to $FFFFFFFF. Many areas of the map are user-programmable, and suggested uses are shown in the table. The cache inhibit function is programmable in the MC68*xx*060 MMU. The onboard I/O space must be marked cache inhibit and serialized in its page table. Table 1-5 further defines the map for the local I/O devices.

**Table 1-4. Local Bus Memory Map**

| Address Range | Devices Accessed | Port Width | Size | Software Cache Inhibit | Note(s) |
|---|---|---|---|---|---|
| Programmable | DRAM on board | D32 | 4MB-16MB | N | 2 |
| Programmable | SRAM | D32 | 128KB-2MB | N | 2 |
| Programmable | VMEbus A32/A24 | D32/D16 | -- | ? | 4 |
| Programmable | IP a Memory | D32-D8 | 64KB-8MB | ? | 2, 4 |
| Programmable | IP b Memory | D32-D8 | 64KB-8MB | ? | 2, 4 |
| Programmable | IP c Memory | D32-D8 | 64KB-8MB | ? | 2, 4 |
| Programmable | IP d Memory | D32-D8 | 64KB-8MB | ? | 2, 4 |
| $FF800000-$FF9FFFFF | Flash/PROM | D32 | 2MB | N | 1, 5 |
| $FFA00000-$FFBFFFFF | PROM/Flash | D32 | 2MB | N | 6 |
| $FFC00000-$FFCFFFFF | Not Decoded | -- | 1MB | N | 7 |
| $FFD00000-$FFDFFFFF | Not Decoded | -- | 1MB | N | 7 |
| $FFE00000-$FFE7FFFF | SRAM Default | D32 | 512KB | N | -- |
| $FFE80000-$FFEFFFFF | Not Decoded | -- | 512KB | N | 7 |
| $FFF00000-$FFFEFFFF | Local I/O | D32-D8 | 878KB | Y | 3 |
| $FFFF0000-$FFFFFFFF | VMEbus A16 | D32/D16 | 64KB | ? | 2, 4 |

## Table 1-4. Local Bus Memory Map (Continued)

| Address Range | Devices Accessed | Port Width | Size | Software Cache Inhibit | Note(s) |
|---|---|---|---|---|---|

**Notes**

1. Reset enables the decoder for this space of the memory map so that it will decode address spaces $FF800000 - $FF9FFFFF and $00000000 - $003FFFFF. The decode at 0 must be disabled in the MC2 chip before DRAM is enabled. DRAM is enabled with the DRAM Control Register at address $FFF42048, bit 24. PROM/Flash is disabled at the low address space with PROM Control Register at address $FFF42040, bit 20.
2. This area is user-programmable. The DRAM and SRAM decoder is programmed in the MC2 chip, the local-to-VMEbus decoders are programmed in the VMEchip2, and the IP memory space is programmed in the IP2 chip.
3. Size is approximate.
4. Cache inhibit depends on devices in area mapped.
5. The PROM and Flash are sized by the MC2 chip ASIC from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.
6. If jumper GPI3 is installed, the Flash device is accessed. If GPI3 is not installed, the PROM is accessed.
7. The Flash and PROM are sized by the MC2 chip ASIC from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.
8. If jumper GPI3 is installed, the PROM is accessed. If GPI3 is not installed, the Flash device is accessed.
9. These areas are not decoded unless one of the programmable decoders are initialized to decode this space. If they are not decoded, an access to this address range will generate a local bus timeout. The local bus timer must be enabled.
10. The following table focuses on the Local I/O Devices portion of the local bus Main Memory Map.

## Table 1-5. Local Bus I/O Devices Memory Map

| Address Range | Device | Port Width | Size | Note(s) |
|---|---|---|---|---|
| $FFF00000 - $FFF3FFFF | Reserved | -- | 256KB | 4 |
| $FFF40000 - $FFF400FF | VMEchip2 (LCSR) | D32 | 256B | 1, 3 |
| $FFF40100 - $FFF401FF | VMEchip2 (GCSR) registers | D32-D8 | 256B | 1, 3 |
| $FFF40200 - $FFF40FFF | Reserved | -- | 3.5KB | 4, 5 |
| $FFF41000 - $FFF41FFF | Reserved | -- | 4KB | 4 |
| $FFF42000 - $FFF42FFF | MC2 chip | D32-D8 | 4KB | 1 |
| $FFF43000 - $FFF44FFF | Reserved | -- | 8KB | 4 |
| $FFF45000 - $FFF45FFF | SCC (Z85230) | D8 | 4KB | 1, 2 |
| $FFF46000 - $FFF46FFF | LAN (82596CA) | D32 | 4KB | 1, 6 |
| $FFF47000 - $FFF47FFF | SCSI (53C710) | D32-D8 | 4KB | 1 |
| $FFF48000 - $FFF57FFF | Reserved | -- | 64KB | 4 |
| $FFF58000 - $FFF5807F | IP2 chip IP a I/O | D16 | 128B | 1 |
| $FFF58080 - $FFF580FF | IP2 chip IP a ID | D16 | 128B | 1 |
| $FFF58100 - $FFF5817F | IP2 chip IP b I/O | D16 | 128B | 1 |
| $FFF58180 - $FFF581FF | IP2 chip IP b ID Read | D16 | 128B | 1 |
| $FFF58200 - $FFF5827F | IP2 chip IP c I/O | D16 | 128B | 1 |
| $FFF58280 - $FFF582FF | IP2 chip IP c ID | D16 | 128B | 1 |
| $FFF58300 - $FFF5837F | IP2 chip IP d I/O | D16 | 128B | 1 |
| $FFF58380 - $FFF583FF | IP2 chip IP d ID Read | D16 | 128B | 1 |
| $FFF58400 - $FFF584FF | IP2 chip IP ab I/O | D32-D16 | 256B | 1 |
| $FFF58500 - $FFF585FF | IP2 chip IP cd I/O | D32-D16 | 256B | 1 |
| $FFF58600 - $FFF586FF | IP2 chip IP ab I/O repeated | D32-D16 | 256B | 1 |
| $FFF58700 - $FFF587FF | IP2 chip IP cd I/O repeated | D32-D16 | 256B | 1 |
| $FFF58800 - $FFF5887F | Reserved | -- | 128B | 1 |
| $FFF58880 - $FFF588FF | Reserved | -- | 128B | 1 |
| $FFF58900 - $FFF5897F | Reserved | -- | 128B | 1 |
| $FFF58980 - $FFF589FF | Reserved | -- | 128B | 1 |
| $FFF58A00 - $FFF58A7F | Reserved | -- | 128B | 1 |
| $FFF58A80 - $FFF58AFF | Reserved | -- | 128B | 1 |
| $FFF58B00 - $FFF58B7F | Reserved | -- | 128B | 1 |

## Table 1-5. Local Bus I/O Devices Memory Map (Continued)

| Address Range | Device | Port Width | Size | Note(s) |
|---|---|---|---|---|
| $FFF58B80 - $FFF58BFF | Reserved | -- | 128B | 1 |
| $FFF58C00 - $FFF58CFF | Reserved | -- | 256B | 1 |
| $FFF58D00 - $FFF58DFF | Reserved | -- | 256B | 1 |
| $FFF58E00 - $FFF58EFF | Reserved | -- | 256B | 1 |
| $FFF58F00 - $FFF58FFF | Reserved | -- | 256B | 1 |
| $FFFBC000 - $FFFBC01F | IP2 chip registers | D32-D8 | 2KB | 1 |
| $FFFBC800 - $FFFBC81F | Reserved | -- | 2KB | 1 |
| $FFFBD000 - $FFFBFFFF | Reserved | -- | 12KB | 4 |
| $FFFC0000 - $FFFC7FFF | MK48T58 (BBRAM, TOD clock) | D32-D8 | 32KB | 1 |
| $FFFC8000 - $FFFCBFFF | MK48T58 | D32-D8 | 16KB | 1, 7 |
| $FFFCC000 - $FFFCFFFF | MK48T58 | D32-D8 | 16KB | 1, 7 |
| $FFFD0000 - $FFFEFFFF | Reserved | -- | 128KB | 4 |

**Notes**

1. For a complete description of the register bits, refer to the *MVME172 VME Embedded Controller Programmer's Reference Guide* or to the data sheet for the specific chip.
2. The SCC is an 8-bit device located on an MC2 chip private data bus. Byte access is required.
3. Writes to the LCSR in the VMEchip2 must be 32 bits. LCSR writes of 8 or 16 bits terminate with a TEA signal. Writes to the GCSR may be 8, 16 or 32 bits. Reads to the LCSR and GCSR may be 8, 16 or 32 bits. Byte reads should be used to read the interrupt vector.
4. This area does not return an acknowledge signal. If the local bus timer is enabled, the access times out and is terminated by a TEA signal.
5. Size is approximate.
6. Port commands to the 82596CA must be written as two 16-bit writes: upper word first and lower word second.
7. Refer to the Flash and PROM Interface section in the MC2 chip description in the *MVME172 VME Embedded Controller Programmer's Reference Guide*.

# VMEbus Memory Map

This section describes the mapping of local resources as viewed by VMEbus masters. Default addresses for the slave, master, and GCSR address decoders are provided by the **ENV** command. Refer to *Appendix A*.

## VMEbus Accesses to the Local Bus

The VMEchip2 includes a user-programmable map decoder for the VMEbus to local bus interface. The map decoder allows you to program the starting and ending address and the modifiers the MVME172 responds to.

## VMEbus Short I/O Memory Map

The VMEchip2 includes a user-programmable map decoder for the GCSR. The GCSR map decoder allows you to program the starting address of the GCSR in the VMEbus short I/O space.

# Software Initialization

Most functions that have been done with switches or jumpers on other modules are done by setting control registers on the MVME172. At powerup or reset, the EPROMs that contain the 172Bug debugging package set up the default values of many of these registers.

Specific programming details may be determined by study of the *M68060 Microprocessor User's Manual*. Then check the details of all the MVME172 onboard registers as given in the *MVME172 VME Embedded Controller Programmer's Reference Guide*.

## Multi-MPU Programming Considerations

Good programming practice dictates that only one MPU at a time have control of the MVME172 control registers. Of particular note are:

- Registers that modify the address map
- Registers that require two cycles to access
- VMEbus interrupt request registers

## Local Reset Operation

Local reset (LRST) is a subset of system reset (SRST). Local reset can be generated five ways:

- Expiration of the watchdog timer
- Pressing the front panel RESET switch (if the system controller function is disabled)
- By asserting a bit in the board control register in the GCSR
- By SYSRESET*
- By powerup reset



**Caution**

The GCSR allows a VMEbus master to reset the local bus. This feature is very dangerous and should be used with caution. The local reset feature is a partial system reset, not a

complete system reset such as powerup reset or SYSRESET*. When the local bus reset signal is asserted, a local bus cycle may be aborted. The VMEchip2 is connected to both the local bus and the VMEbus and if the aborted cycle is bound for the VMEbus, erratic operation may result. Communications between the local processor and a VMEbus master should use interrupts or mailbox locations; reset should not be used in normal communications. Reset should be used only when the local processor is halted or the local bus is hung and reset is the last resort.

Any VMEbus access to the MVME172 while it is in the reset state is ignored. If a global bus timer is enabled, a bus error is generated.

# Hardware Preparation and Installation

# 2

## Introduction

This chapter provides unpacking instructions, hardware preparation, and installation instructions for the MVME172 VME Embedded Controller. Hardware preparation for the MVME712 series transition modules is provided in separate manuals. Refer to *Related Documentation* for more information.

## Unpacking Instructions

**Note** If the shipping carton is damaged upon receipt, request carrier's agent be present during unpacking and inspection of equipment.

Unpack the equipment from the shipping carton. Refer to the packing list and verify that all items are present. Save the packing material for storing and reshipping of equipment.



**Caution**

Avoid touching areas of integrated circuitry; static discharge can damage circuits.

2

# Hardware Preparation

To select the desired configuration and ensure proper operation of MVME172, modifications to certain options may be necessary before you install the board. The MVME172 provides software control for most options. Modifications are performed by setting bits in control registers after the MVME172 has been installed in a system. (For more information on the MVME172 registers, refer to the *MVME172 VME Embedded Controller Programmer's Reference Guide* listed in *Related Documentation*.) Some options cannot be set in software; these are modified by installing or removing header jumpers or interface modules.

Figure 2-1 illustrates the locations of the switches, jumper headers, connectors, and LEDs on the MVME172. The MVME172 has been factory tested and is shipped with the factory jumper settings described in the following sections. The MVME172 operates with its required and factory-installed debug monitor, MVME172Bug (172Bug), with these factory jumper settings.

Manually configurable items are listed in Table 2-1. Note: Jumper J3 is not listed as it is for manufacturing purposes only and cannot be configured.

**Table 2-1. Jumper-Configurable Options**

| Option | Factory Default |
|---|---|
| *System Controller Select Header (J1)* | 2-3 |
| *IP DMA Snoop Control Header (J2)* | 1-2, 3-4 |
| *IP Bus Clock Header (J14)* | 1-2 |
| *SIMM selection for serial port B configuration (J15)* | SIMM06 |
| *Clock Select Header (J16) for Serial Port 1/Console* | No jumpers |
| *Clock Select Header (J17) for Serial Port 2* | No jumpers |
| *IP Bus Strobe Select Header (J19)* | 1-2 |
| *SRAM Battery Backup Source Select Header (J22)* | 1-3, 2-4 |
| *EPROM Size Select Header (J23)* | 2-3 |
| *Flash Write Protect Header (J24)* | 1-2 |
| *General Purpose Readable Jumpers Header (J28)* | 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16 |

## System Controller Select Header (J1)

The MVME172 is factory-configured as a VMEbus system controller by a jumper across J1 pins 1 and 2. If you select the "automatic" system controller function by moving the jumper to J1 pins 2 and 3, the MVME172 determines whether it is the system controller by its position on the bus. If the board is in the first slot from the left, it configures itself as the system controller. If the MVME172 is not to be system controller under any circumstances, remove the jumper from J1. When the board is functioning as system controller, the SCON LED is turned on.

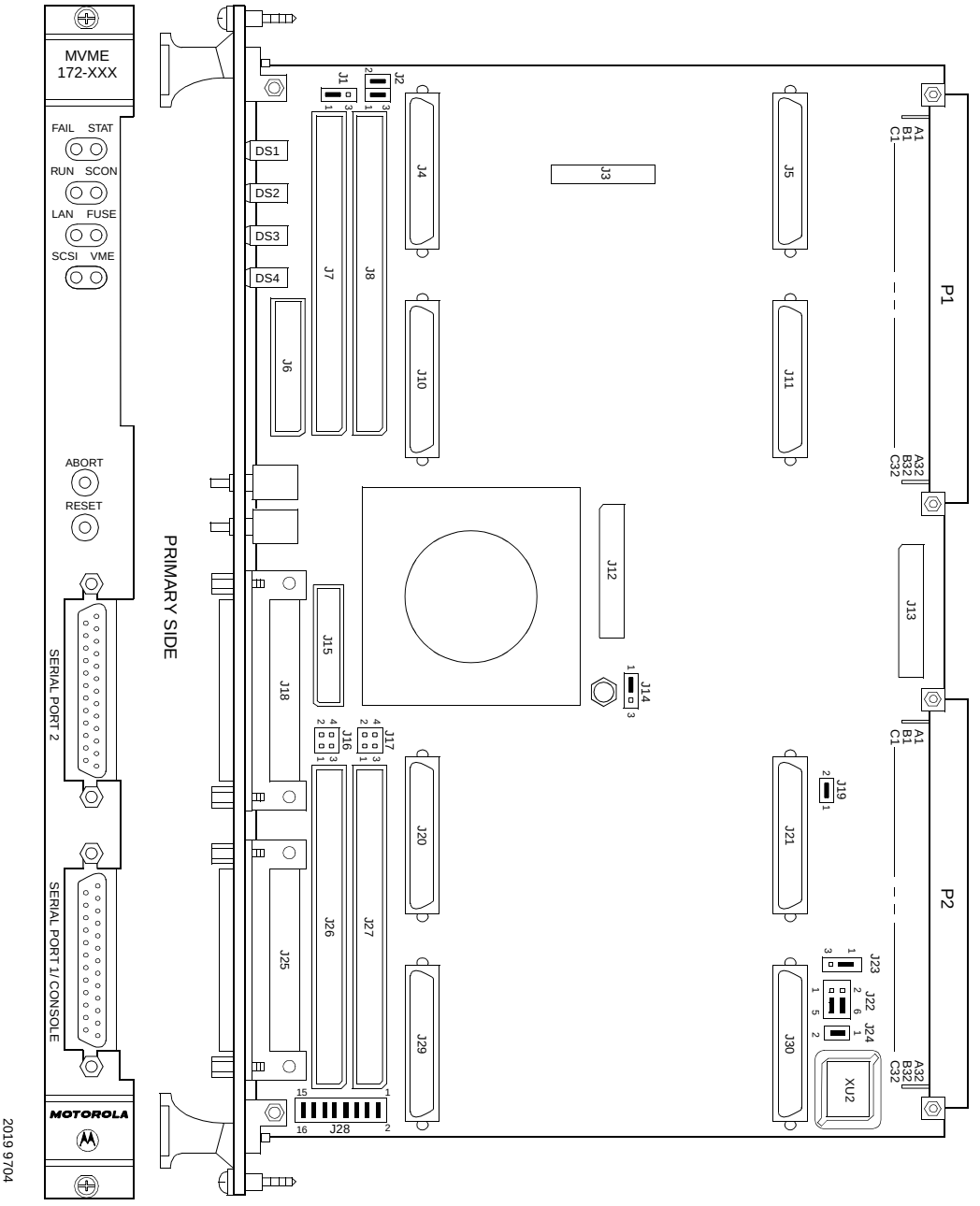
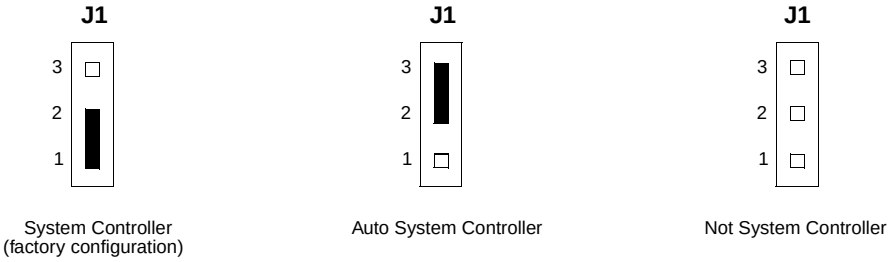



**Figure 2-1. MVME172 Switches, Headers, Connectors, and LEDs**

**Note** For MVME172s without the optional VMEbus interface (i.e., no VMEchip2), the jumper may be installed or removed without affecting normal operation.




## SIMM Selection (J15)

Port B of the MVME172 Z85230 serial communications controller is configurable via a serial interface module (SIMM) that is installed at connector J15 on the MVME172 board. Five serial interface modules are available:

- EIA-232-D (DCE and DTE)
- EIA-530 (DCE and DTE)
- EIA-485/EIA-422 (DCE or DTE)

You can change Port B from an EIA-232-D to an EIA-530 interface or to an EIA-485/EIA-422 interface (or vice-versa) by mounting the appropriate serial interface module. Port B is routed (via the SIMM at J15) to the 25-pin DB25 front panel connector marked SERIAL PORT 2.

For the location of SIMM connector J15 on the MVME172, refer to Figure 2-1. Figure 2-2 illustrates the secondary side (bottom) of a serial interface module, showing the J1 connector which plugs into SIMM connector J15 on the MVME172. Figure 2-3 (sheets 3-6), Figure 2-4, Figure 2-5 (sheets 3 and 4), and Figure 2-6 illustrate the nine configurations available for Port B.

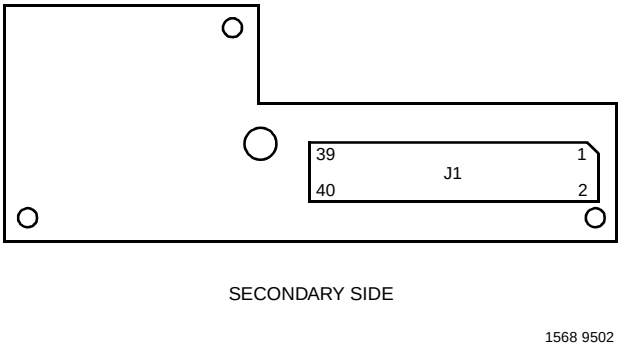



**Figure 2-2. Serial Interface Module, Connector Side**

For the part numbers of the serial interface modules, refer to Table 2-2. The part numbers are ordinarily printed on the primary side (top) of the SIMMs, but may be found on the secondary side in some versions.

If you need to replace an existing serial interface module with a SIMM of another type, go to *Removal of Existing SIMM* below. If there is no SIMM on the main board, skip to *Installation of New SIMM*.

**Table 2-2. Serial Interface Module Part Numbers**

| EIA Standard | Configuration | Part Number | Model Number |
|---|---|---|---|
| EIA-232-D | DTE | 01-W3846B | SIMM05 |
| | DCE | 01-W3865B | SIMM06 |
| EIA-530 | DTE | 01-W3868B | SIMM07 |
| | DCE | 01-W3867B | SIMM08 |
| EIA-485 | -- | 01-W3002F | SIMM09 |
| or EIA-422 | DTE or DCE | | |

## Removal of Existing SIMM

1. Each serial interface module is retained by two 4-40 x 3/16-in. Phillips-head screws in opposite corners. (Exception: SIMM09 is

retained by one Phillips-head screw in the center of the module.) Remove the screw(s) and store them in a safe place for later use.

2. Grasp opposite sides of the SIMM and gently lift straight up.



**Caution**

Avoid lifting the SIMM by one side only, as the connector can be damaged on the SIMM or the main board.

3. Place the SIMM in a static-safe container for possible reuse.

### Installation of New SIMM

1. Observe the orientation of the connector keys on SIMM connector J1 and MVME172 connector J15. Turn the SIMM so that the keys line up and place it gently on connector J15, aligning the mounting hole(s) at the SIMM corners (or center) with the matching standoff(s) on the MVME172.
2. Gently press the top of the SIMM to seat it on the connector. If the SIMM does not seat with gentle pressure, recheck the orientation. If the SIMM connector is oriented incorrectly, the mounting hole(s) will not line up with the standoff(s).



**Caution**

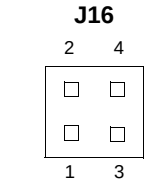
Do not attempt to force the SIMM on if it is oriented incorrectly.

3. Place the one or two 4-40 x 3/16" Phillips-head screw(s) that you previously removed (or that were supplied with the new SIMM) into the one center or two opposite-corner mounting hole(s). Screw it or them into the standoff(s) but do not overtighten it or them.

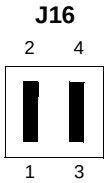
The signal relationships and signal connections in the various serial configurations available for ports A and B are illustrated in Figures 2-3, 2-4, 2-5, and 2-6.

## Clock Select Header (J16) for Serial Port 1/Console

The MVME172 is shipped from the factory with the SERIAL PORT 1/CONSOLE header configured for asynchronous communications (i.e., jumpers removed). To select synchronous communications for the SERIAL PORT 1/CONSOLE connection, install jumpers across pins 1 and 2 and pins 3 and 4.



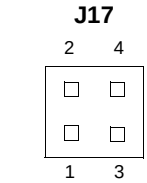

Internal Clock
(Factory configuration)



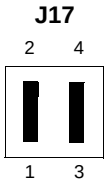

External Clock

## Clock Select Header (J17) for Serial Port 2

The MVME172 is shipped from the factory with the SERIAL PORT 2 header configured for asynchronous communications (i.e., jumpers removed). To select synchronous communications for the SERIAL PORT 2 connection, install jumpers across pins 1 and 2 and pins 3 and 4.




Internal Clock
(Factory configuration)




External Clock

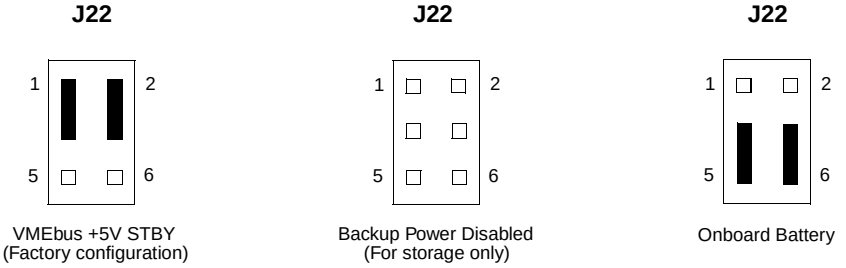
## SRAM Battery Backup Source Select Header (J22)

The MVME172 is factory-configured to use VMEbus +5V Standby power as a backup power source for the SRAM (i.e., jumpers are installed across pins 1 and 3 and 2 and 4). To select the onboard battery as the backup power source, install the jumpers across pins 3 and 5 and 4 and 6.

**Note** For MVME172s without optional VMEbus interface (i.e., without VMEchip2 ASIC), you must select the onboard battery for the backup power source.

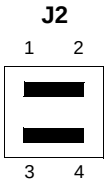
**Caution**

Removing all jumpers may temporarily disable the SRAM. Do not remove all jumpers from J22, except for storage.



## IP DMA Snoop Control Header (J2)

J2 defines the state of the snoop control bus when an IP DMA controller is local bus master. J2 pins 1 and 2 control the Snoop Control signal. Pins 3 and 4 are not used for the MC68xx060.




Snoop Inhibited
(Factory configuration)

The following table lists the snoop operations represented by the setting of J2.

**Table 2-3. J2 Snoop Control Encoding**

| Pins 1-2 | Pins 3-4 | Snoop Operation |
|---|---|---|
| 0 | X | Snoop enabled |
| 1 | X | Snoop disabled |

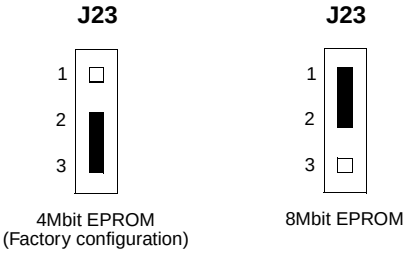
X = don’t care

Jumper enabled = logic 0

Jumper removed = logic 1

## EPROM Size Select Header (J23)

Header J23 selects the EPROM size. The MVME172 is factory-configured for a 4Mbit EPROM.



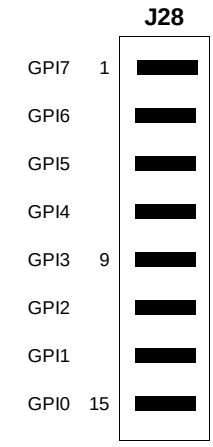

## General Purpose Readable Jumpers Header (J28)

Header J28 provides eight readable jumpers. These jumpers are read as a register (at $FFF4202D) in the MC2chip LCSR (local control/status register). The bit values are read as a zero when the jumper is installed and as a one when the jumper is removed.

With the factory-installed 172Bug firmware in place, four jumpers are user-definable (pins 1-2, 3-4, 5-6, 7-8). If the 172Bug firmware is removed, seven jumpers are user-definable (pins 1-2, 3- 4, 5-6, 7-8, 11-12, 13-14, 15-16).

**Note** Pins 9-10 (GPI3) are reserved to select either the Flash memory map (jumper installed) or the EPROM memory map (jumper removed). They are not user-definable. See Chapter 3 for more information.

In most cases, the MVME172 is shipped from the factory with J28 set to all zeros (jumpers on all pins). *On boards built with the no-VMEbus option*, however, no jumper is installed across pins 9-10.




| | 172BUG INSTALLED | USER CODE INSTALLED |
|---|---|---|
| GPI7 | USER-DEFINABLE | USER-DEFINABLE |
| GPI6 | USER-DEFINABLE | USER-DEFINABLE |
| GPI5 | USER-DEFINABLE | USER-DEFINABLE |
| GPI4 | USER-DEFINABLE | USER-DEFINABLE |
| GPI3 | IN=FLASH; OUT=EPROM | IN=FLASH; OUT=EPROM |
| GPI2 | REFER TO 172BUG MANUAL | USER-DEFINABLE |
| GPI1 | REFER TO 172BUG MANUAL | USER-DEFINABLE |
| GPI0 | REFER TO 172BUG MANUAL | USER-DEFINABLE |

Flash Selected (factory configuration except on non-VMEbus models)

## Flash Write Protect Header (J24)

When the Flash write protect jumper is installed (factory configuration), Flash memory can be written to via the normal software routines. When the jumper is removed, Flash memory cannot be written to.




Flash Write Protect
(Factory configuration)

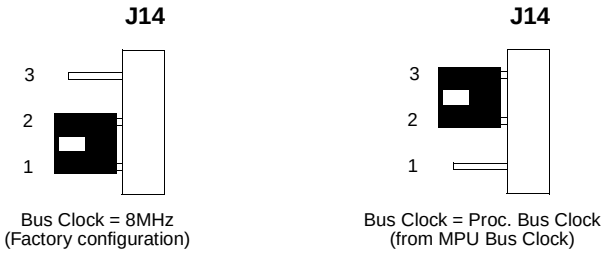
## IP Bus Clock Header (J14)

This header selects the speed of the IP bus clock. The IP bus clock speed may be 8MHz or it may be set synchronous to the processor bus clock.

If the jumper is installed between J14 pins 2 and 3, the IP bus clock is the same as the MC68060 bus clock, that is 32MHz (MC68LC060) or 30MHz (MC68060) allowing the IP module to run with a 32MHz MPU. All IP ports operate at the same speed.

**Caution**

The IP32 CSR bit (IP2 chip, register at offset $1D, bit 0) must be set to correspond to the jumper setting. This is cleared (0) for 8MHz, or set (1) for 30 or 32MHz. If the jumper and the bit are not configured the same, the board may not run properly.

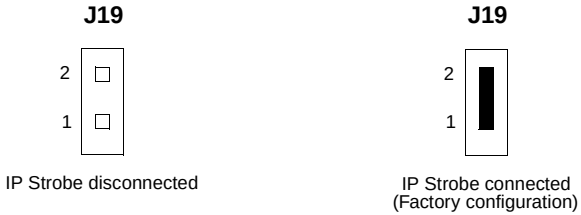


## IP Bus Strobe Select Header (J19)

Some IP bus implementations make use of the Strobe∗ signal (pin 46) as an input to the IP modules from the IP2 chip. Other IP interfaces require that the strobe be disconnected.

With a jumper installed between J19 pins 1 and 2, a programmable frequency source is connected to the Strobe∗ signal on the IP bus (for details, refer to the IP2 chip programming model in the MVME172 *Programmer's Reference Guide*).

If the jumper is removed from J19, the strobe line is available for a sideband type of messaging between IP modules. The Strobe∗ signal is not connected to any active devices on the board, but it may be connected to a pull-up resistor.



2

# Installation Instructions

The following sections discuss the installation of IndustryPacks (IPs) on the MVME172, the installation of the MVME172 into a VME chassis, and the system considerations relevant to the installation. Before installing IndustryPacks, ensure that the serial ports and all header jumpers are configured as desired.

## IP Installation on the MVME172

Up to four IndustryPack (IP) modules may be installed on the MVME172. Install the IPs on the MVME172 as follows:

1. Each IP has two 50-pin connectors that plug into two corresponding 50-pin connectors on the MVME172: J4/J5, J10/J11, J20/J21, J29/J30. See Figure 2-1 on page 2-4 for the MVME172 connector locations.
   - Orient the IP(s) so that the tapered connector shells mate properly. Plug IP_a into connectors J4 and J5; plug IP_b into J10 and J11. Plug IP_c into J20 and J21; plug IP_d into J29 and J30. If a double-sized IP is used, plug IP_ab into J4, J5, J10, and J11; plug IP_cd into J20, J21, J29, and J30.
2. Four additional 50-pin connectors (J7, J8, J26, and J27) are provided behind the MVME172 front panel for external cabling connections to the IP modules. There is a one-to-one correspondence between the signals on the cabling connectors and the signals on the associated IP connectors (i.e., J8 has the same IP_a signals as J4; J7 has the same IP_b signals as J10; J27 has the same IP_c signals as J20; and J26 has the same IP_d signals as J29.
   - Connect user-supplied 50-pin cables to J7, J8, J26, and J27 as needed. Because of the varying requirements for each different kind of IP, Motorola does not supply these cables.
   - Bring the IP cables out the narrow slots in the MVME172 front panel and attach them to the appropriate external equipment, depending on the nature of the particular IP(s).

## MVME172 Module Installation

With EPROM, IndustryPack, and SIMMs installed and headers properly configured, proceed as follows to install the MVME172 in the VME chassis:

1. Turn all equipment power OFF and disconnect the power cable from the AC power source.



**Caution**

Inserting or removing modules while power is applied could result in damage to module components.



**Warning**

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

2. Remove the chassis cover as instructed in the user's manual for the equipment.
3. Remove the filler panel from the card slot where you are going to install the MVME172.
   - If you intend to use the MVME172 as system controller, it must occupy the leftmost card slot (slot 1). The system controller must be in slot 1 to correctly initiate the bus-grant daisy-chain and to ensure proper operation of the IACK daisy-chain driver.
   - If you do not intend to use the MVME172 as system controller, it can occupy any unused double-height card slot.
4. Slide the MVME172 into the selected card slot. Be sure the module is seated properly in the P1 and P2 connectors on the backplane. Do not damage or bend connector pins.
5. Secure the MVME172 in the chassis with the screws provided, making good contact with the transverse mounting rails to minimize RF emissions.
6. Install the MVME712 series transition module in the front or the rear of the VME chassis. (To install an MVME712M, which has a

double-wide front panel, you may need to shift other modules in the chassis.)

7. On the chassis backplane, remove the INTERRUPT ACKNOWLEDGE (IACK) and BUS GRANT (BG) jumpers from the header for the card slot occupied by the MVME172.

**Note** Some VME backplanes (e.g., those used in Motorola "Modular Chassis" systems) have an autojumpering feature for automatic propagation of the IACK and BG signals. Step 7 does not apply to such backplane designs.

8. Connect the P2 Adapter Board or LCP2 Adapter Board and cable(s) to MVME172 backplane connector P2. This provides a connection point for terminals or other peripherals at the EIA-232-D serial ports, SCSI ports, and LAN Ethernet port. For information on installing the P2 or LCP2 Adapter Board and the MVME712 series transition module(s), refer to the manuals listed in *Related Documentation* (the *MVME172 VME Embedded Controller Programmer's Reference Guide* provides some connection diagrams.)

**Note** If you intend to use the MVME172 with Port B in an EIA-530 configuration or an EIA-485/EIA-422 configuration, do not install the P2 or LCP2 Adapter Board and the MVME712 series transition module. They are incompatible with the EIA-530 interface and the EIA-485/EIA-422 interface. Refer to *MVME172 Serial Port 2* for more information.

9. Connect the appropriate cable(s) to the panel connectors for the serial ports, SCSI port, and LAN Ethernet port.
   – Note that some cables are not provided with the MVME712 series module and must be made or purchased by the user. (Motorola recommends shielded cable for all peripheral connections to minimize radiation.)
10. Connect the peripheral(s) to the cable(s). Appendix A supplies detailed information on the EIA-232-D, EIA-530, and EIA-485/EIA-422 signals supported. Appendix B describes the SCSI (*Small Computer System Interface*) I/O bus connections. Appendix C describes the Ethernet LAN (*Local Area Network*) port connections.
11. Install any other required VMEmodules in the system.
12. Replace the chassis cover.
13. Connect the power cable to the AC power source and turn the equipment power ON.
14. Startup information is located in Chapter 3.

## System Considerations

The MVME172 draws power from VMEbus backplane connectors P1 and P2. P2 is also used for the upper 16 bits of data in 32-bit transfers, and for the upper 8 address lines used in extended addressing mode. The MVME172 may not function properly without its main board connected to VMEbus backplane connectors P1 and P2.

Whether the MVME172 operates as VMEbus master or VMEbus slave, it is configured for 32 bits of address and 32 bits of data (A32/D32). However, it handles A16 or A24 devices in the address ranges indicated in Chapter 1. D8 and/or D16 devices in the system must be handled by the MC68060/MC68LC060 software. Refer to the memory maps in the *MVME172 VME Embedded Controller Programmer's Reference Guide*.)

The MVME172 contains shared onboard DRAM whose base address is software-selectable. Both the onboard processor and offboard VMEbus devices see this local DRAM at base physical address $00000000, as programmed by the 172Bug firmware. This may be changed via software to any other base address. Refer to the *MVME172 VME Embedded Controller Programmer's Reference Guide* for more information.

If the MVME172 tries to access offboard resources in a nonexistent location and is not system controller, and if the system does not have a global bus timeout, the MVME172 waits forever for the VMEbus cycle to complete. This will cause the system to lock up. There is only one situation in which the system might lack this global bus timeout: when the MVME172 is not the system controller and there is no global bus timeout elsewhere in the system.

Multiple MVME172s may be installed in a single VME chassis. In general, hardware multiprocessor features are supported.

**Note** If you are installing multiple MVME172s in an MVME945 chassis, do not install an MVME172 in slot 12. The height of the IP modules may cause clearance difficulties in that slot position.

Other MPUs on the VMEbus can interrupt, disable, communicate with, and determine the operational status of the processor(s). One register of the GCSR (global control/status register) set includes four bits that function as location monitors to allow one MVME172 processor to broadcast a signal to any other MVME172 processors. All eight registers are accessible from any local processor as well as from the VMEbus.

The following circuits are protected by solid-state fuses that reset during overload conditions: remote reset connector, IndustryPack +5V, and IndustryPack ±12V.

The FUSE LED illuminates to indicate that all fuses are functioning correctly. If a fuse opens, power must be removed for several minutes to allow the fuse to reset.

The MVME172 provides +12 Vdc power to the Ethernet transceiver interface through a solid-state fuse (F2). The FUSE LED lights to indicate that +12 Vdc is available. When the MVME712M module is used, the yellow DS1 LED on the MVME712M illuminates when LAN power is available, which indicates that the fuse is good. If the Ethernet transceiver fails to operate, check fuse F2.

The MVME172 provides SCSI terminator power through a 1A fuse (F1) located on the P2 Adapter Board or LCP2 Adapter Board. If the fuse is blown, the SCSI device(s) may function erratically or not at all. When the P2 Adapter Board is used with an MVME712M and the SCSI bus is connected to the MVME712M, the green DS2 LED on the MVME712M front panel illuminates when SCSI terminator power is available. If the green DS2 LED flickers during SCSI bus operation, check P2 Adapter Board fuse F1.

The following figures illustrate the signal relationships and signal connections in the various serial configurations available for ports A and B.

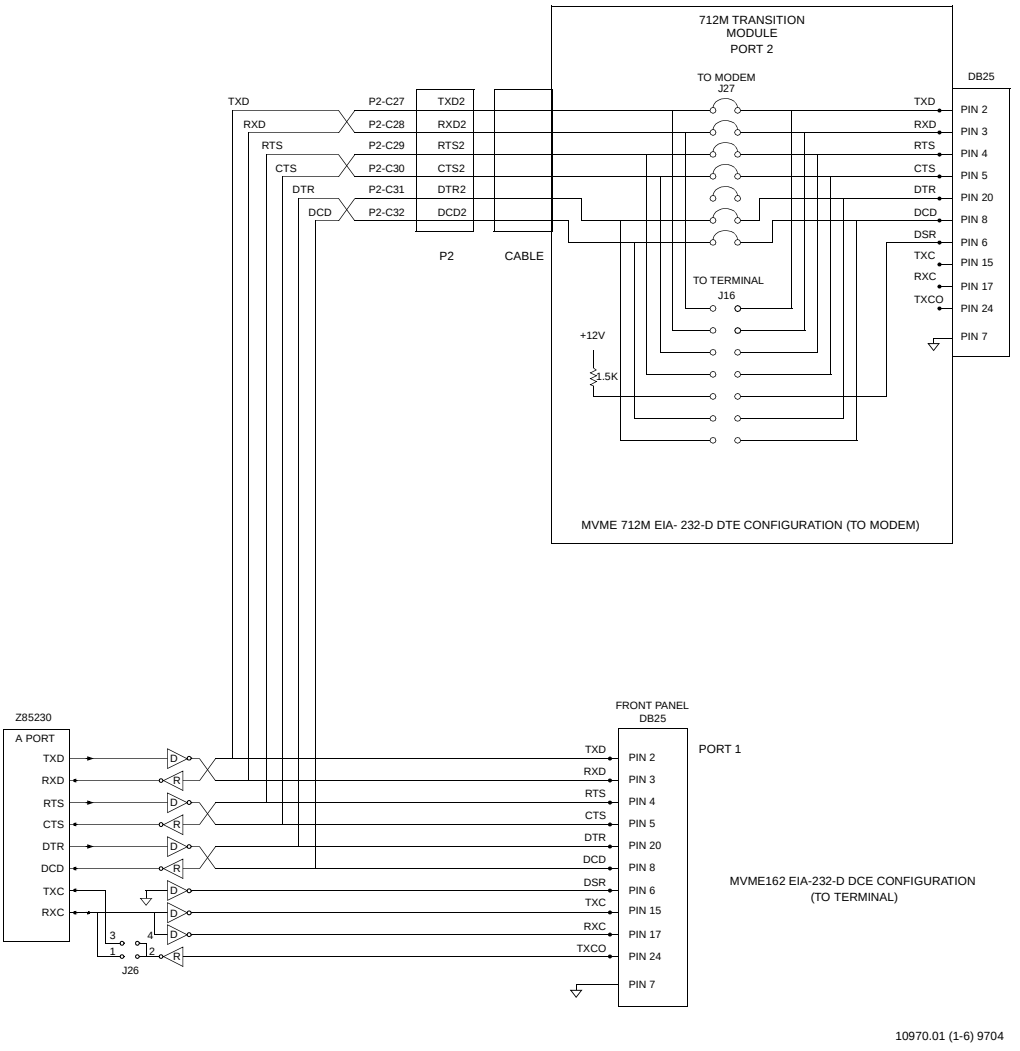



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 1 of 6)**

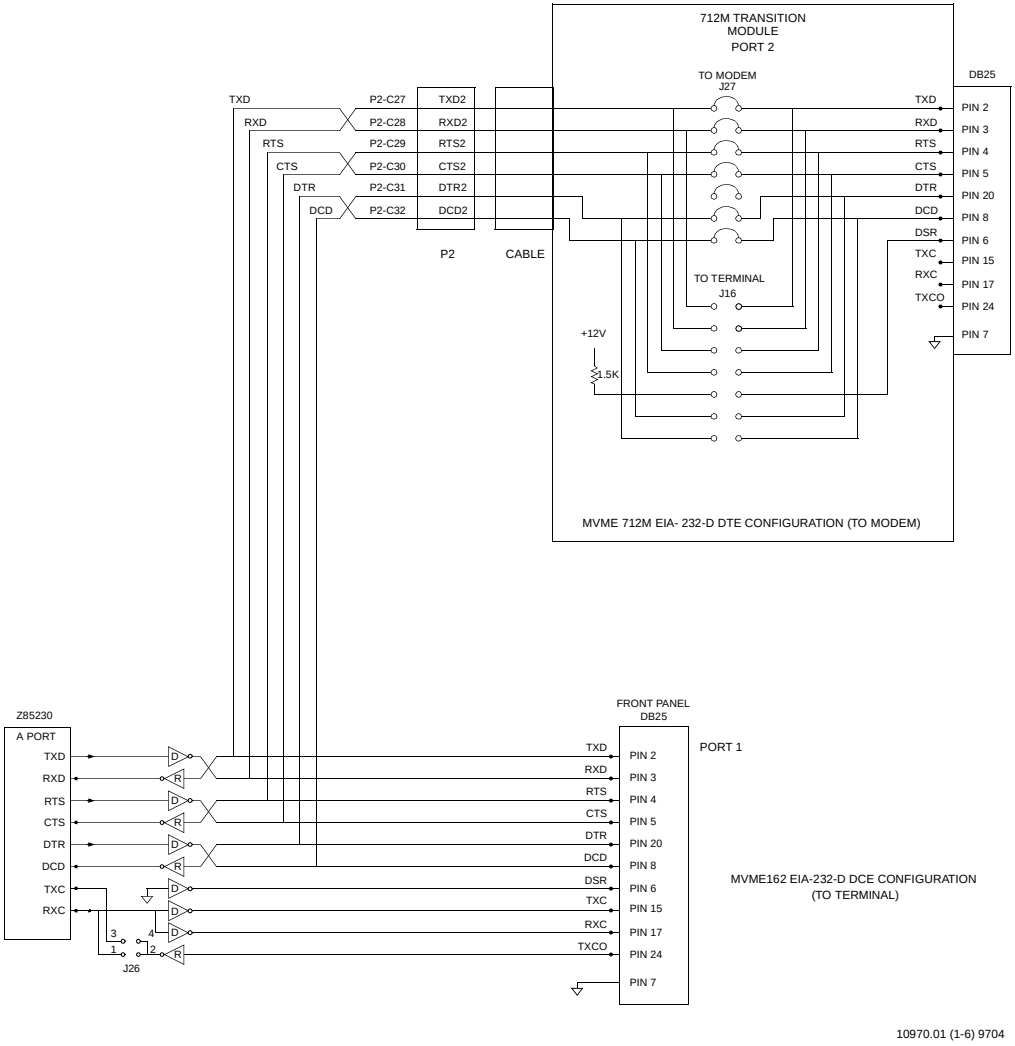



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 2 of 6)**

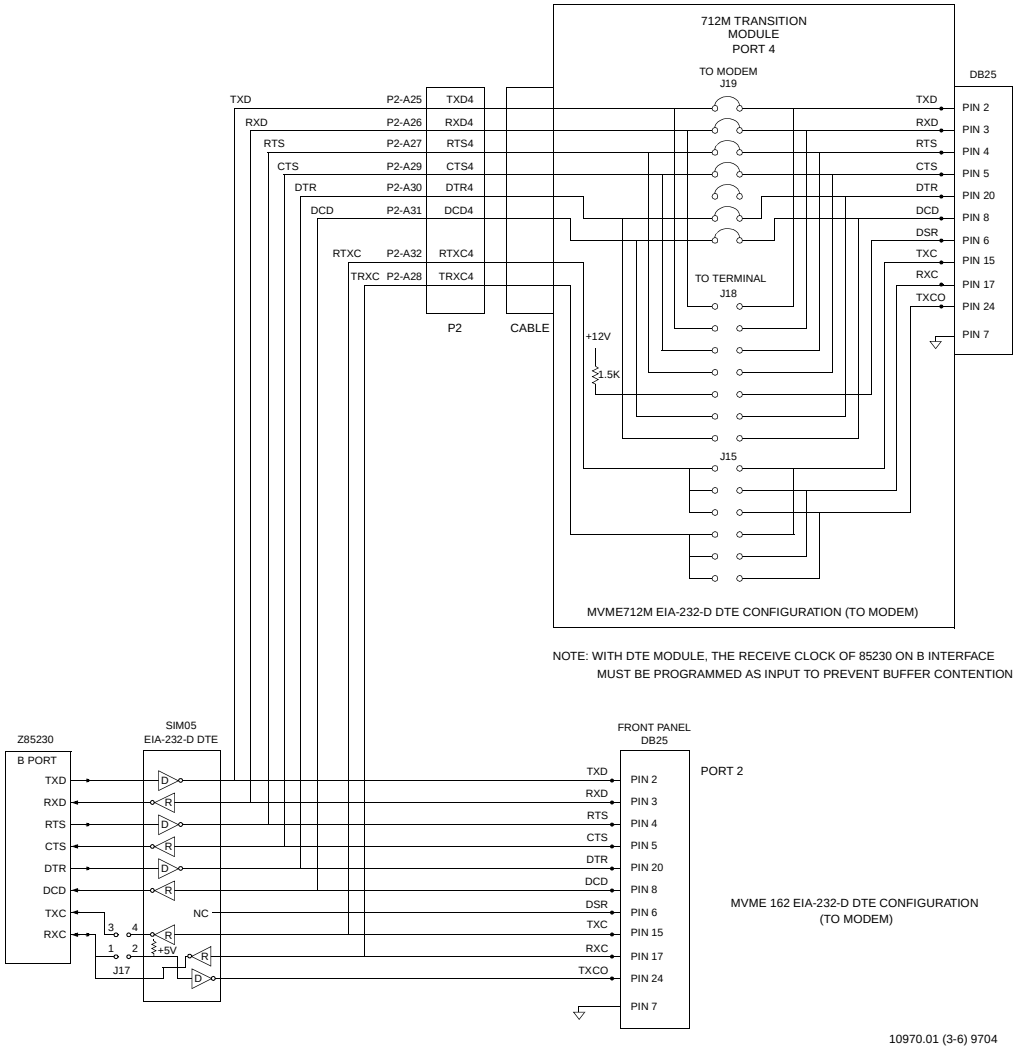



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 3 of 6)**

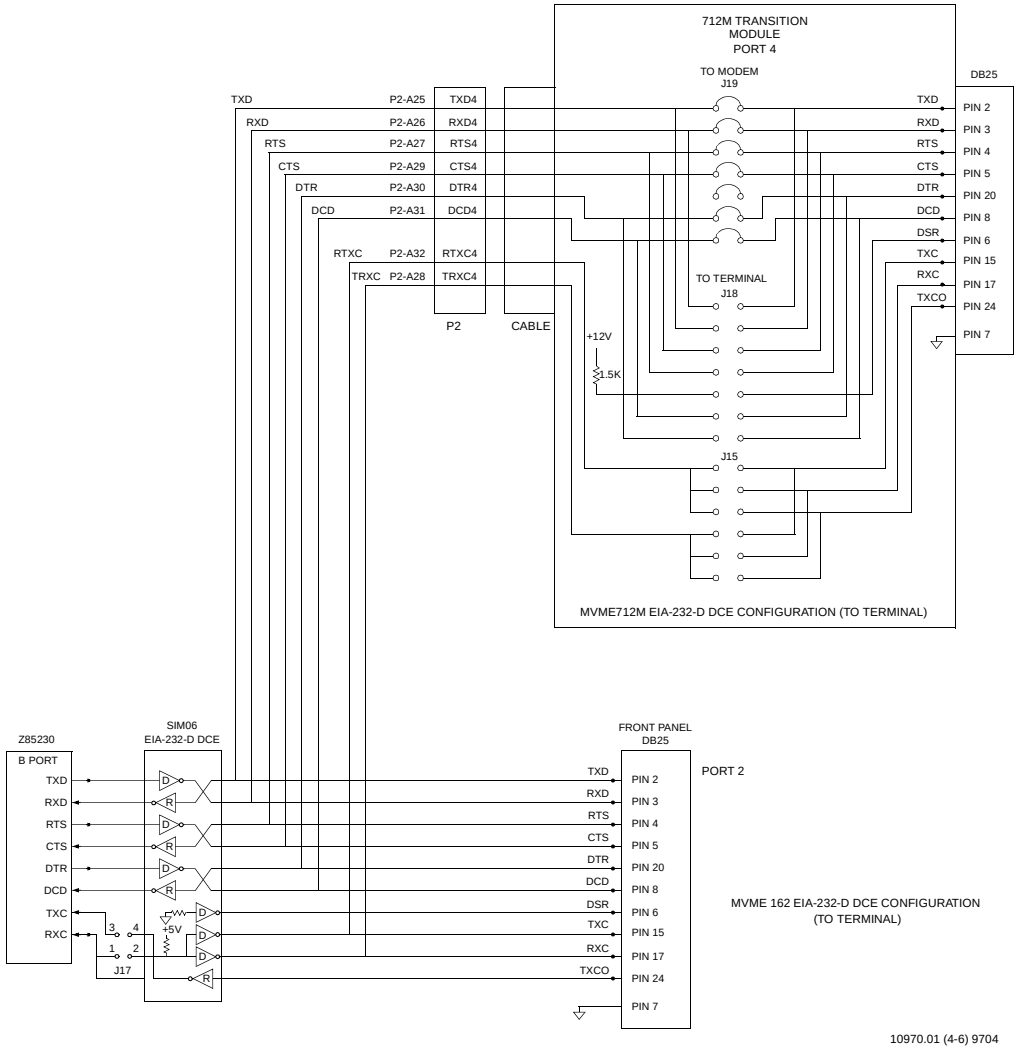



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 4 of 6)**

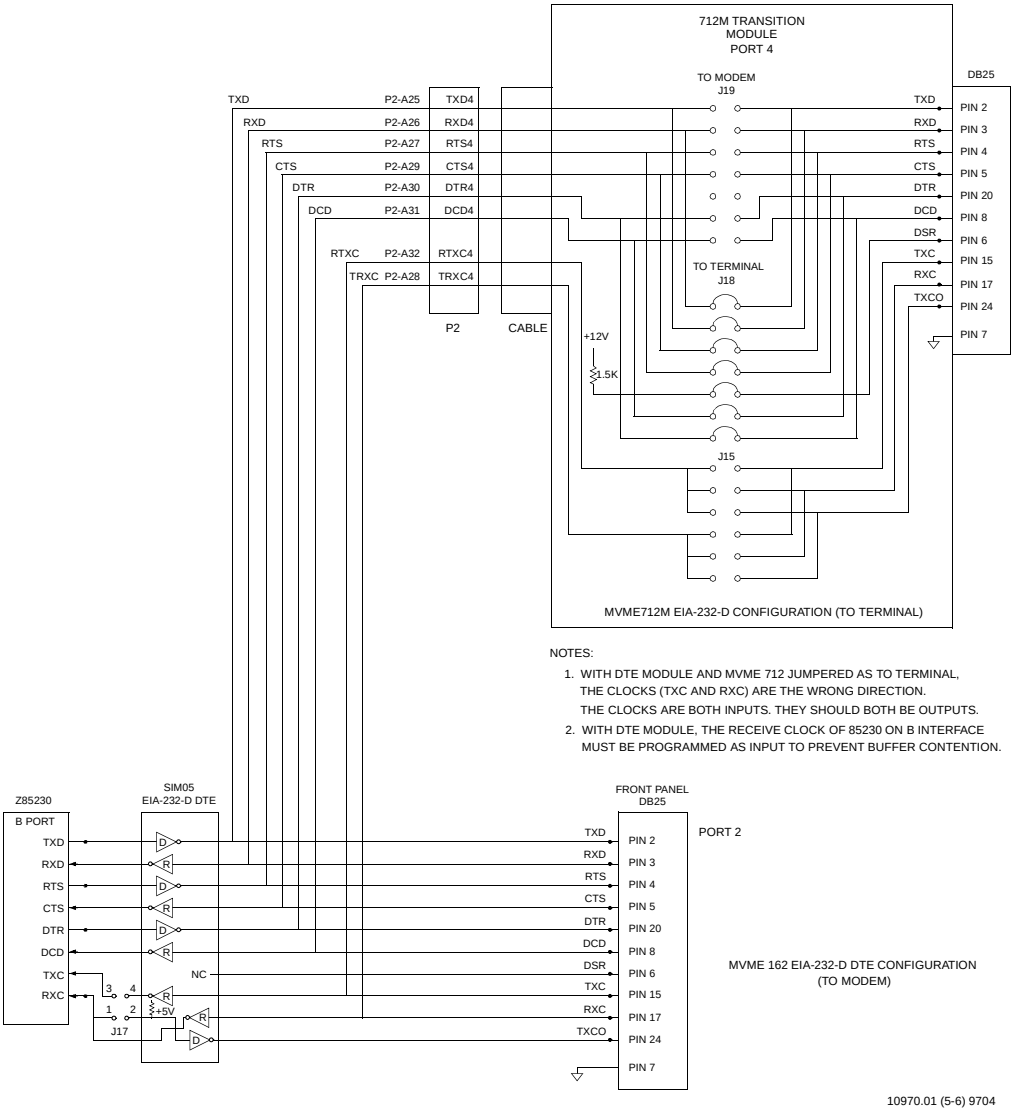



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 5 of 6)**

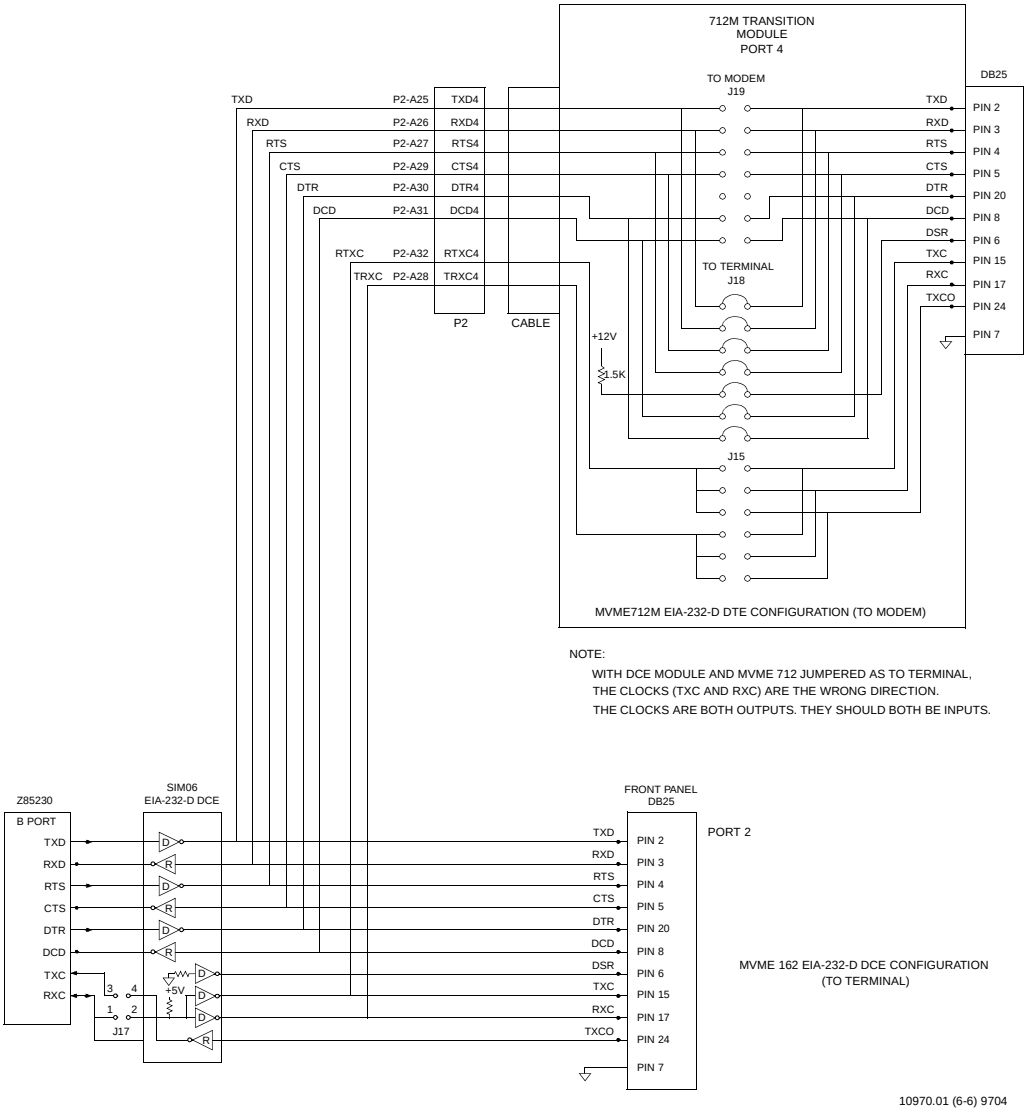



**Figure 2-3. MVME172 EIA-232-D Connections, MVME712M (Sheet 6 of 6)**

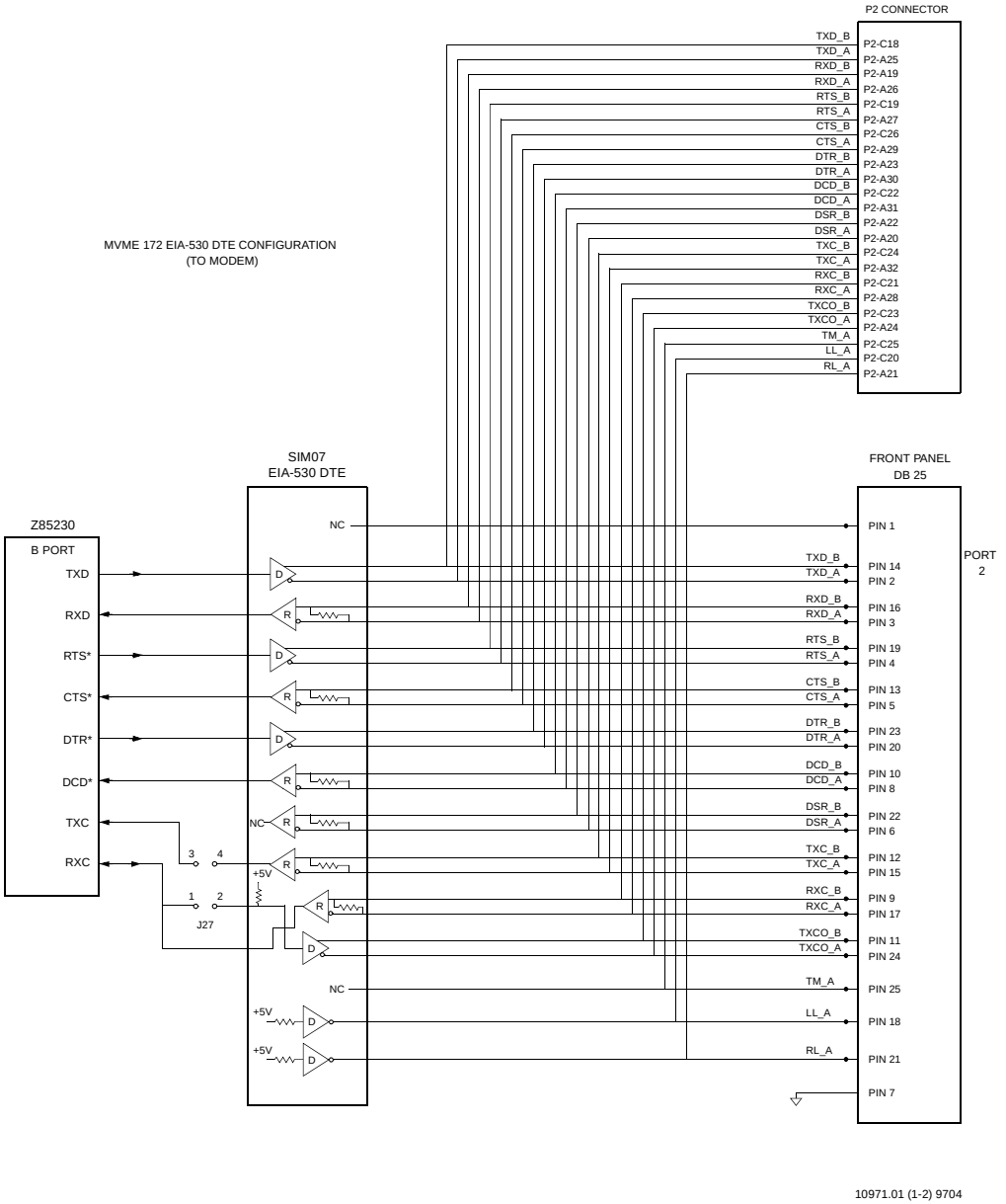



**Figure 2-4. MVME172 EIA-530 Connections (Sheet 1 of 2)**

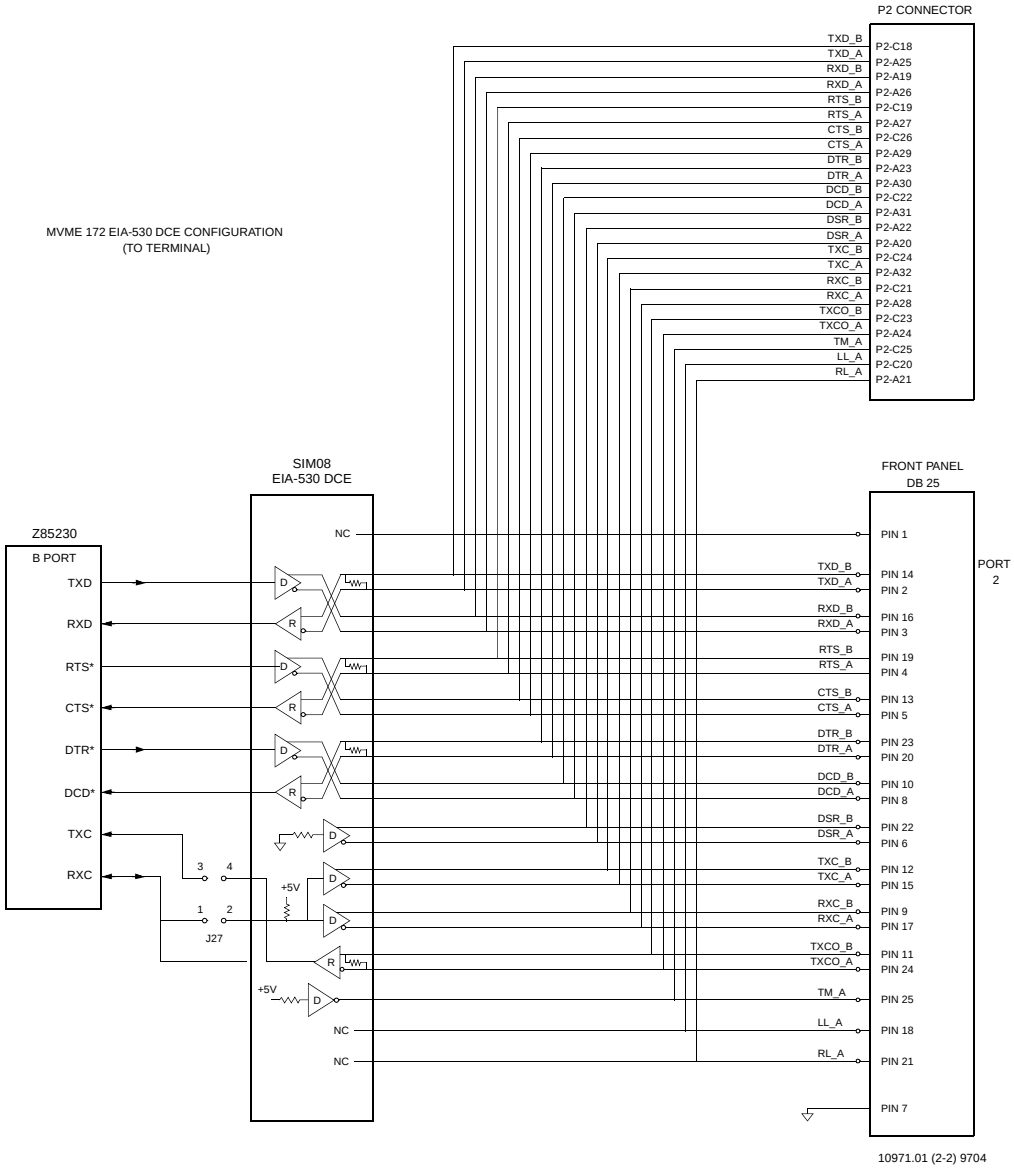



**Figure 2-4. MVME172 EIA-530 Connections (Sheet 2 of 2)**

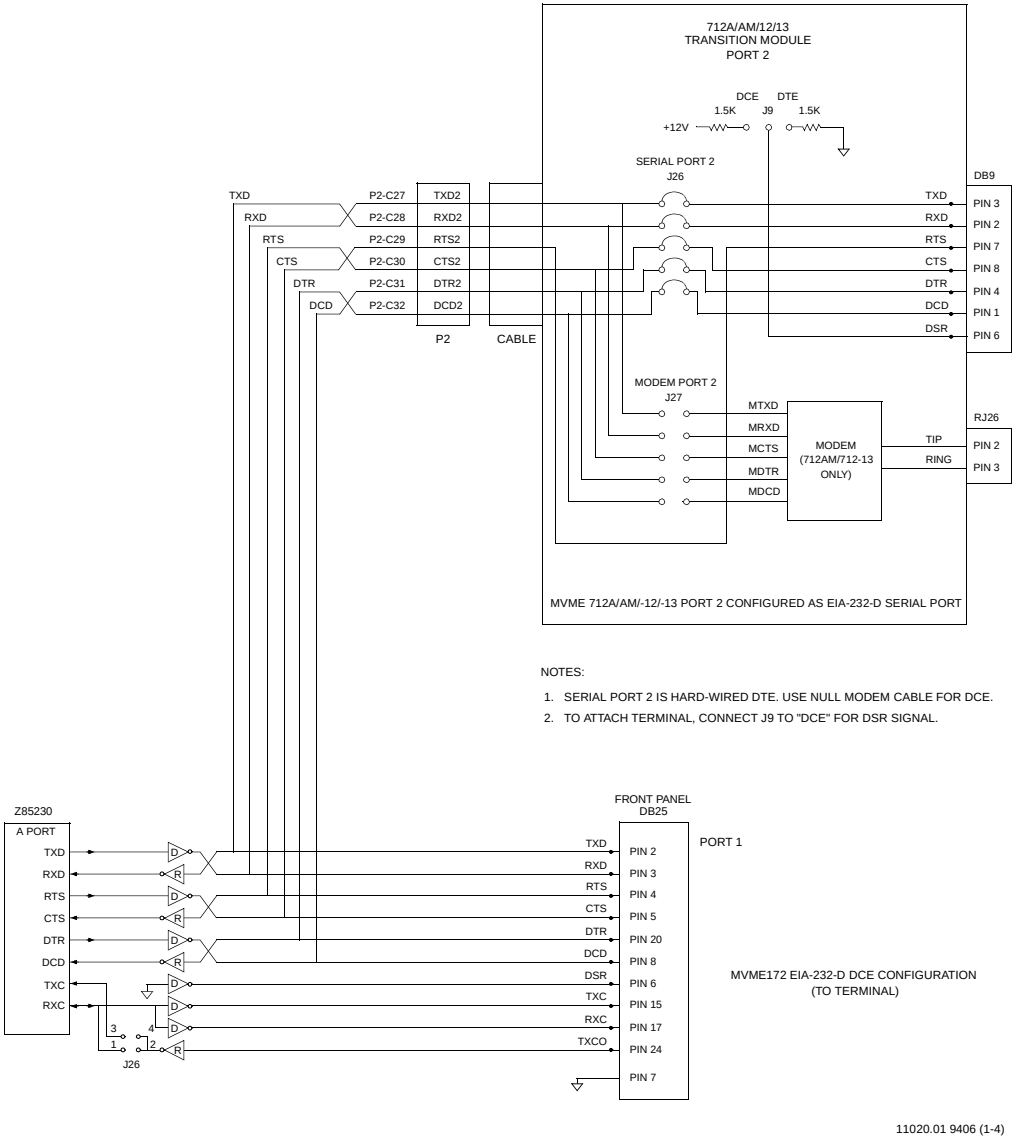



**Figure 2-5. MVME172 EIA-232-D Connections, MVME712A/AM/-12/-13 (Sheet 1 of 4)**

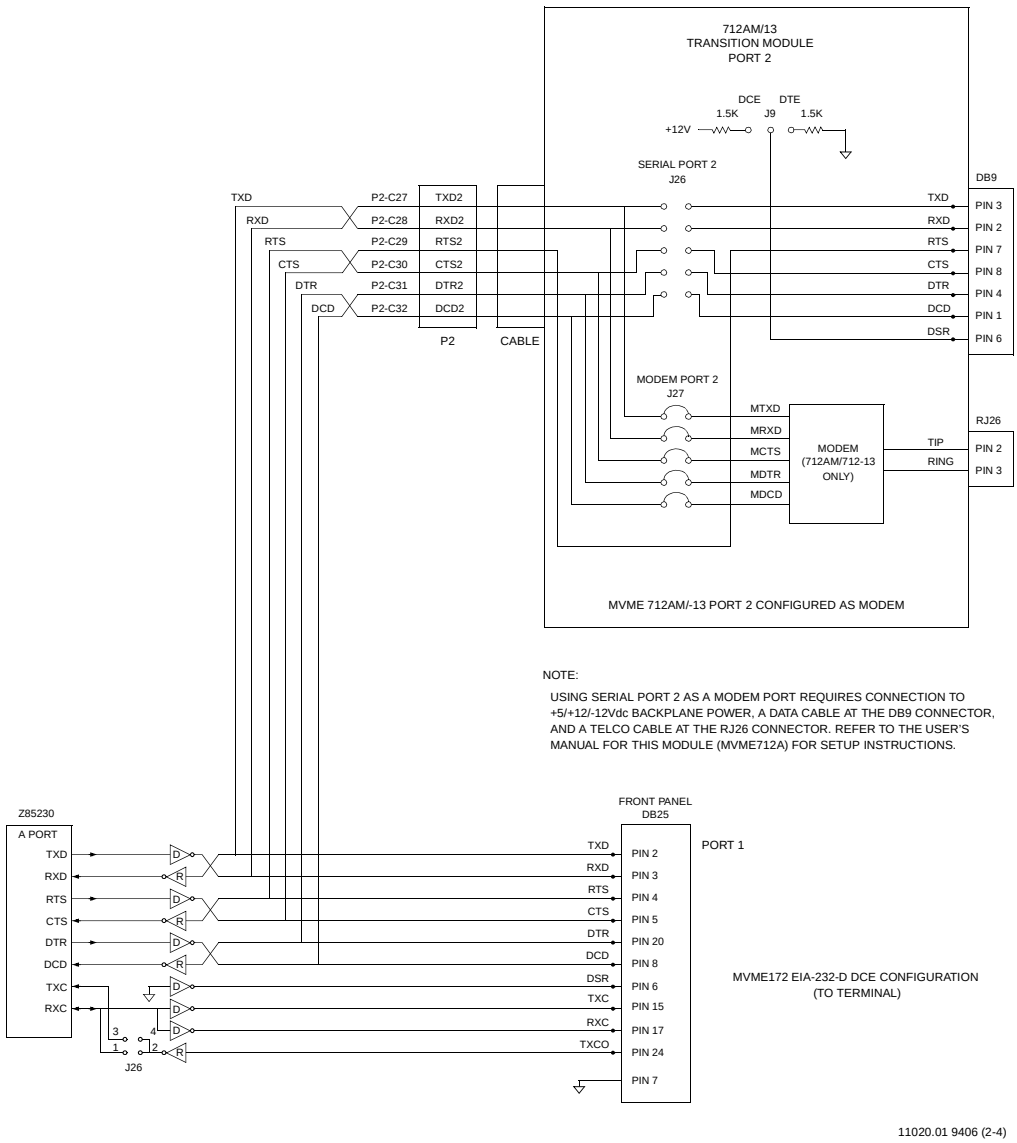



**Figure 2-5. MVME172 EIA-232-D Connections, MVME712A/AM/-12/-13 (Sheet 2 of 4)**

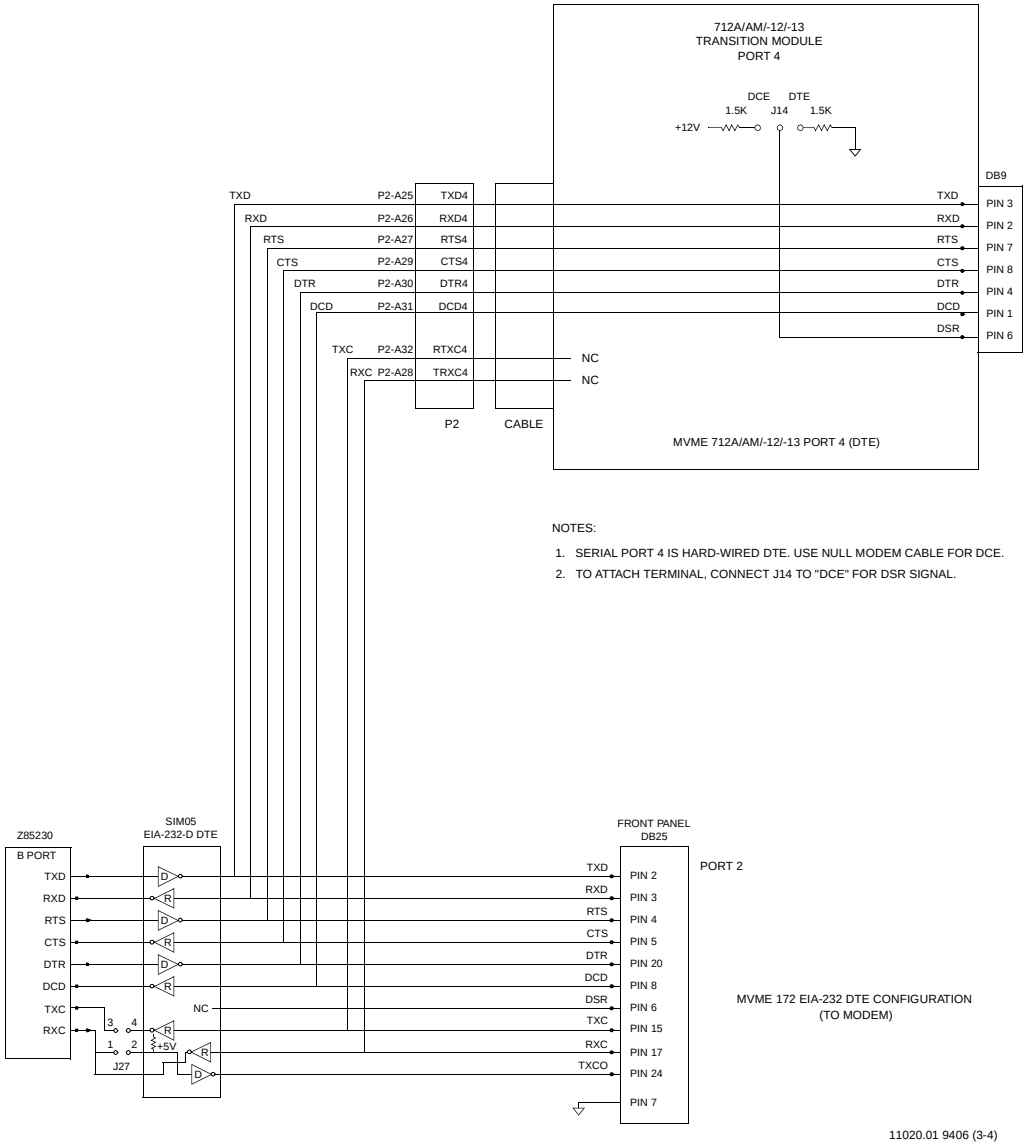



**Figure 2-5. MVME172 EIA-232-D Connections, MVME712A/AM/-12/-13 (Sheet 3 of 4)**

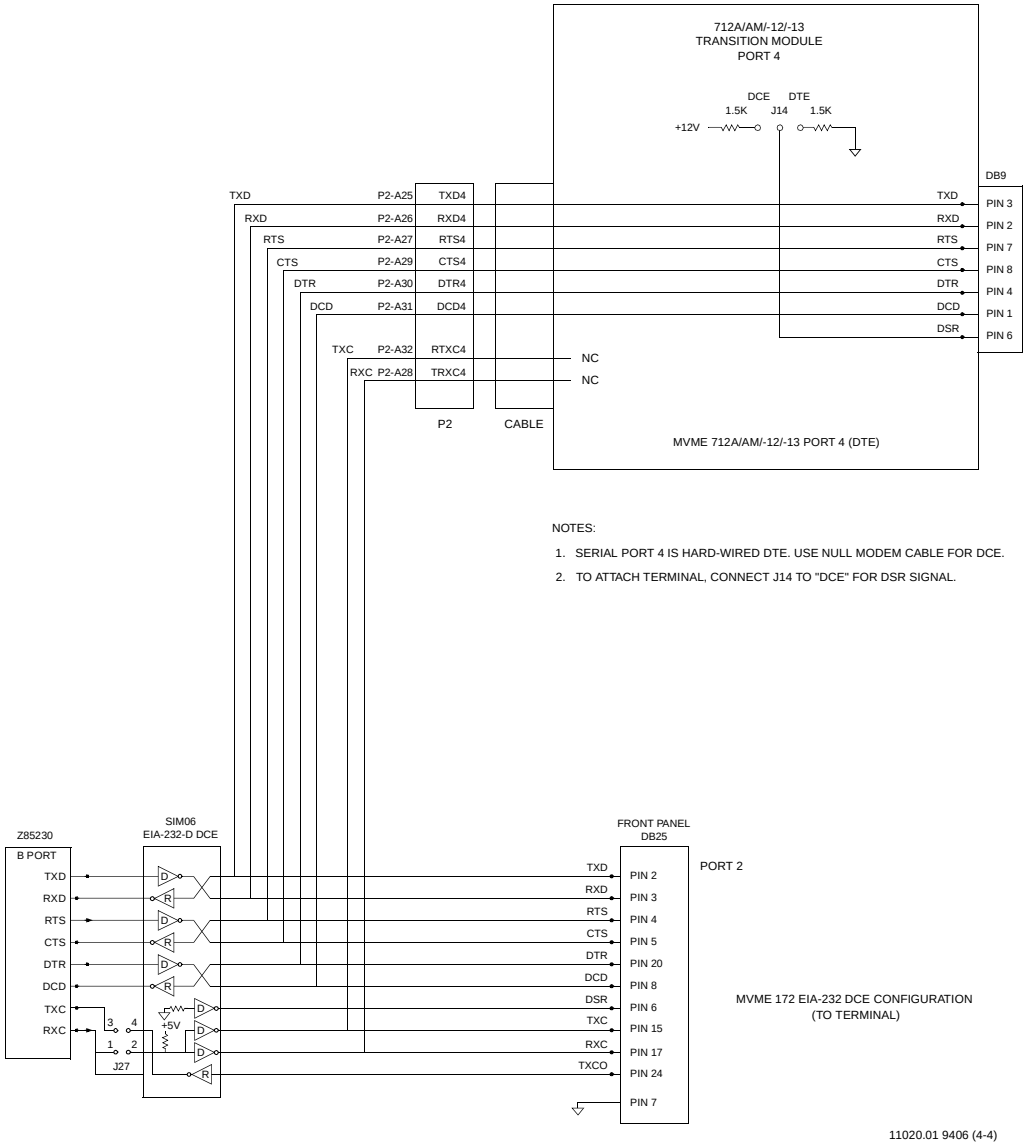



**Figure 2-5. MVME172 EIA-232-D Connections, MVME712A/AM/-12/-13 (Sheet 4 of 4)**

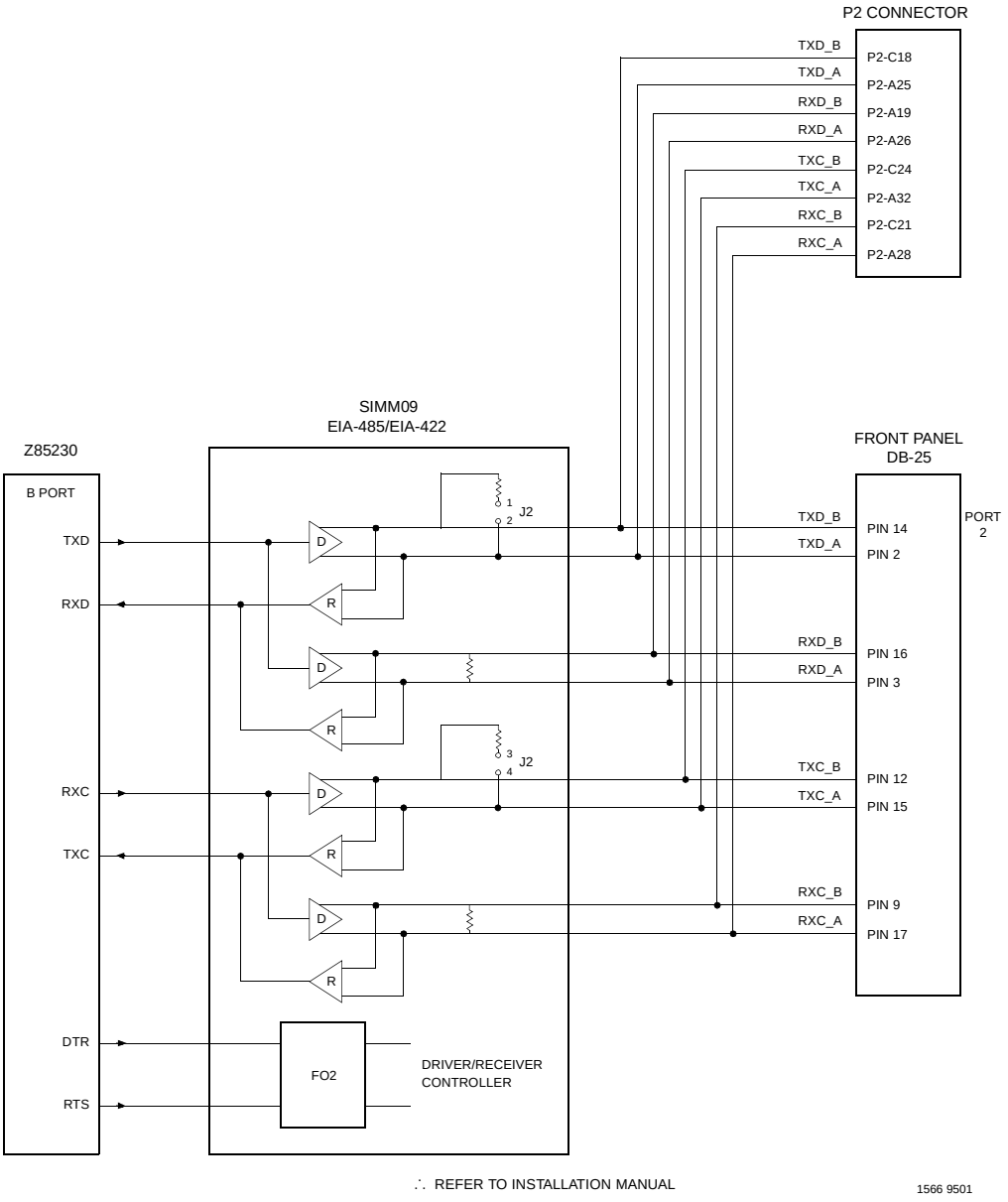



**Figure 2-6. MVME172 EIA-485/EIA-422 Connections**

# Debugger General Information

# 3

## Overview of M68000 Firmware

The firmware for the M68000-based (68K) series of board and system level products has a common genealogy, deriving from the debugger firmware currently used on all Motorola M68000-based CPU modules. The M68000 firmware family provides a high degree of functionality and user friendliness, and yet stresses portability and ease of maintenance. The M68000 firmware implementation on the MVME172 MC68060-based or MC68LC060-based Embedded Controller is known as the MVME172Bug, or 172Bug. It includes diagnostics for testing and configuring IndustryPack modules.

## Description of 172Bug

The 172Bug package, MVME172Bug, is a powerful evaluation and debugging tool for systems built around the MVME172 CISC-based microcomputers. Facilities are available for loading and executing user programs under complete operator control for system evaluation. 172Bug includes commands for display and modification of memory, breakpoint and tracing capabilities, a powerful assembler/disassembler useful for patching programs, and a power-up self test which verifies the integrity of the system. Various 172Bug routines that handle I/O, data conversion, and string functions are available to user programs through the TRAP #15 system calls.

172Bug consists of three parts:

- A command-driven user-interactive software debugger, described in *Chapter 4* and hereafter referred to as "the debugger" or "172Bug".
- A command-driven diagnostic package for the MVME172 hardware, described in the *MVME172Bug Diagnostics Manual*, and hereafter referred to as "the diagnostics".
- A user interface which accepts commands from the system console terminal.

When using 172Bug, you operate out of either the debugger directory or the diagnostic directory. If you are in the debugger directory, the debugger prompt "`172-Bug>`" is displayed and you have all of the debugger commands at your disposal. If you are in the diagnostic directory, the diagnostic prompt "`172-Diag>`" is displayed and you have all of the diagnostic commands at your disposal as well as all of the debugger commands. You may switch between directories by using the Switch Directories (**SD**) command, or may examine the commands in the particular directory that you are currently in by using the Help (**HE**) command.

Because 172Bug is command-driven, it performs its various operations in response to user commands entered at the keyboard. When you enter a command, 172Bug executes the command and the prompt reappears. However, if you enter a command that causes execution of user target code (e.g., **"GO"**), then control may or may not return to 172Bug, depending on the outcome of the user program.

If you have used one or more of Motorola's other debugging packages, you will find the CISC 172Bug very similar. Some effort has also been made to make the interactive commands more consistent. For example, delimiters between commands and arguments may now be commas or spaces interchangeably.

# 172Bug Implementation



MVME172Bug is written largely in the "C" programming language, providing benefits of portability and maintainability. Where necessary, assembler has been used in the form of separately compiled modules containing only assembler code — no mixed language modules are used.

Physically, 172Bug is contained in the 28F016SA Flash memory, providing 512KB (128K longwords) of storage. Optionally, the 172Bug can be loaded and executed in a single 27C040 PROM. (128K longwords) of storage. The executable code is checksummed at every power-on or reset firmware entry, and the result (which includes a pre-calculated checksum contained in the memory devices), is tested for an expected zero. Thus, users are cautioned against modification of the memory devices unless re-checksum precautions are taken.

**Note** MVME172 models ordered without the VMEbus interface are shipped with Flash memory blank (the factory uses the VMEbus to program the Flash memory with debugger code). To use the 172Bug package, be sure that jumper header J28 is configured to select the EPROM memory map.

If you subsequently wish to run the debugger from Flash memory, you must first initialize Flash memory with the PFLASH command, then reconfigure J28. Refer to Step 12 (Note) on page 3-8 for further details.

# Installation and Startup



Follow the steps below to operate 172Bug with the MVME172 module. 172Bug is factory-installed in the Flash memory of the MVME172, except in the no-VMEbus case.



**Caution**

Inserting or removing modules while power is applied could damage module components.

1. Turn all equipment power OFF. Refer to the *Hardware Preparation* section to install/remove jumpers on headers as required for your particular application.

   Jumpers on header J28 affect 172Bug operation as listed below. The default condition is with all eight jumpers installed, between pins 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, and 15-16. (Models with no VMEbus interface have no jumper between pins 9-10.)

   These readable jumpers can be read as a register (at $FFF4202D) on the Memory Controller (MC2 chip) ASIC. The bit values are read as a one when the jumper is off, and as a zero when the jumper is on. This jumper block (header J28) contains eight bits. Refer also to the *MVME172 VME Embedded Controller Programmer's Reference Guide* for more information on the MC2 chip.

   The MVME172Bug reserves/defines the four lower order bits (GPI3 to GPI0). The following is the description for the bits reserved/defined by the debugger:

| Bit | J28 Pins | Description |
|---|---|---|
| Bit #0 (GPI0) | 15-16 | When set to 1 (high), instructs the debugger to use local Static RAM for its work page (i.e., variables, stack, vector tables, etc.). |

| Bit | J28 Pins | Description |
|---|---|---|
| Bit #1 (GPI1) | 13-14 | When set to 1 (high), instructs the debugger to use the default setup/operation parameters in Flash or PROM versus the user setup/operation parameters in NVRAM. This is the same as depressing the RESET and ABORT switches at the same time. This feature can be used in the event the user setup is corrupted or does not meet a sanity check. Refer to the **ENV** command (*Appendix A*) for the Flash/PROM defaults. |
| Bit #2 (GPI2) | 11-12 | Reserved for future use. |
| Bit #3 (GPI3) | 9-10 | When set to 0 (low), informs the debugger that it is executing out of the Flash memory. When set to 1 (high), as set in non-VMEbus models, informs the debugger that it is executing out of the PROM. |
| Bit #4 (GPI4) | 7-8 | Open to your application. |
| Bit #5 (GPI5) | 5-6 | Open to your application. |
| Bit #6 (GPI6) | 3-4 | Open to your application. |
| Bit #7 (GPI7) | 1-2 | Open to your application. |

Note that when the MVME172 comes up in a cold reset, 172Bug runs in Board Mode. Using the Environment (**ENV**) or **MENU** commands can make 172Bug run in System Mode. Refer to *Appendix A* for details.

2. Configure header J1 by installing/removing a jumper between pins 1 and 2. A jumper installed/removed enables/disables the system controller function of the MVME172.

3. You may configure Port B of the Z85230 serial communications controller via a serial interface module (SIMM) which is installed at connector J15 on the MVME172 board. Five serial interface modules are available:
   - EIA-232-D DTE (SIMM05)
   - EIA-232-D DCE (SIMM06)
   - EIA-530 DTE (SIMM07)
   - EIA-530 DCE (SIMM08)
   - EIA-485, or EIA-422 DTE or DCE (all with SIMM09)

   For information on removing and/or installing a SIMM, refer to *Chapter 2*.

4. Jumpers on headers J16 and J17 configure serial ports 1 and 2 to drive or receive clock signals provided by the TXC and RXC signal lines. The factory configures the module for asynchronous communication, that is, installs no jumpers. Refer to *Chapter 2* if your application requires configuring ports 1 and 2 for synchronous communication.

5. The jumper on header J14 configures the IP bus clock for either 8MHz (on both MVME172-4*xx* and -5*xx* boards) or 32MHz (on MVME172-5xx boards only). The factory configuration puts a jumper between J14 pins 1 and 2 for an 8MHz clock. Verify that this setting is appropriate for your application.

6. The jumper on header J19 enables/disables the IP bus strobe function on the MVME172. The factory configuration puts a jumper between J19 pins 1 and 2 to connect the Strobe∗ signal to the IP2 chip. Verify that the strobe line should be connected in your application.

7. Header J2 defines the state of the snoop control bus when an IP DMA controller is local bus master. The factory configuration has both jumpers in place for snoop inhibition. (Pins 3 and 4 are not used for the MC68xx060.) Verify that this setting is appropriate for your application.

8. Refer to the setup procedure for your particular chassis or system for details concerning the installation of the MVME172.

9. Connect the terminal that is to be used as the 172Bug system console to the default debug EIA-232-D port at serial port 1 on the front panel of the MVME172 module. Refer to *Chapter 2* for other connection options. Set up the terminal as follows:
   - eight bits per character
   - one stop bit per character
   - parity disabled (no parity)
   - baud rate 9600 baud (default baud rate of MVME172 ports at power-up)

   After power-up, you can reconfigure the baud rate of the debug port if necessary by using the Port Format (**PF**) command of the 172Bug debugger.

**Note** In order for high-baud rate serial communication between 172Bug and the terminal to work, the terminal must do some form of handshaking. If the terminal being used does not do hardware handshaking via the CTS line, then it must do XON/XOFF handshaking. If you get garbled messages and missing characters, then you should check the terminal to make sure XON/XOFF handshaking is enabled.

10. If you want to connect devices (such as a host computer system and/or a serial printer) to the other EIA-232-D port connectors (marked SERIAL PORT on the MVME712*x* transition module), connect the appropriate cables and configure the port(s) as detailed in Step 3 above. After power-up, you can reconfigure the port(s) by programming the MVME172 Z85230 Serial Communications Controller (SCC), or by using the 172Bug **PF** command.

11. Power up the system. 172Bug executes some self-checks and displays the debugger prompt "`172-Bug>`" (if in Board Mode). However, if the **ENV** command (Appendix A) has put 172Bug in System Mode, the system performs a selftest and tries to autoboot. Refer to the **ENV** and **MENU** commands. They are listed in Table 4-3.

    If the confidence test fails, the test is aborted when the first fault is encountered. If possible, an appropriate message is displayed, and control then returns to the menu.

12. Before using the MVME172 after the initial installation, set the date and time using the following command line structure:

    `172-Bug>` **SET** [*mmddyyhhmm*]|[<+/-*CAL*>;**C** ]

    For example, the following command line starts the real-time clock and sets the date and time to 10:37 AM, June 6, 1998:

    `172-Bug>` **SET 0606981037**

    The board's self-tests and operating systems require that the real-time clock be running.

**Note** If you wish to execute the debugger out of Flash and Flash does not contain 172Bug, you may copy the PROM version of 172Bug to Flash memory. To copy the PROM version of 172Bug to Flash memory, first remove the jumper at J28 pins 9 and 10, make sure that 172Bug is in Board Mode, and copy the PROM contents to Flash memory with the **PFLASH** command as follows:

```
172-Bug> PFLASH FF800000:80000 FFA00000
```

Then reinstall the jumper at J28 pins 9 and 10. (172Bug always executes from memory location FF800000; the state of J28 determines whether that location is in PROM or Flash.)

# Autoboot



Autoboot is a software routine that is contained in the 172Bug Flash/PROM to provide an independent mechanism for booting an operating system. This autoboot routine automatically scans for controllers and devices in a specified sequence until a valid bootable device containing a boot media is found or the list is exhausted. If a valid bootable device is found, a boot from that device is started. The controller scanning sequence goes from the lowest controller Logical Unit Number (LUN) detected to the highest LUN detected. Controllers, devices, and their LUNs are listed in Appendix E.

At power-up, Autoboot is enabled, and providing the drive and controller numbers encountered are valid, the following message is displayed upon the system console:

```
Autoboot in progress... To abort hit <BREAK>
```

Following this message there is a delay to allow you an opportunity to abort the Autoboot process if you wish. Then the actual I/O is begun: the program pointed to within the volume ID of the media specified is loaded into RAM and control passed to it. If, however, during this time you want to gain control without Autoboot, you can press the <BREAK> key or the software ABORT or RESET switches.

Autoboot is controlled by parameters contained in the **ENV** command. These parameters allow the selection of specific boot devices and files, and allow programming of the Boot delay. Refer to the **ENV** command in *Appendix A* for more details.



**Caution**

Although streaming tape can be used to autoboot, the same power supply must be connected to the streaming tape drive, controller, and the MVME172. At power-up, the tape controller will position the streaming tape to load point where the volume ID can correctly be read and used.

If, however, the MVME172 loses power but the controller does not, and the tape happens to be at load point, the sequences of commands required (attach and rewind) cannot be given to the controller and autoboot will not be successful.

# ROMboot



As shipped from the factory, 172Bug occupies the first quarter of the Flash memory. This leaves the remainder of the Flash memory and the PROM socket (XU2) available for your use. This function is configured/enabled by the Environment (**ENV**) command (refer to *Appendix A*) and executed at power-up (optionally also at reset) or by the **RB** command assuming there is valid code in the memory devices (or optionally elsewhere on the module or VMEbus) to support it. If ROMboot code is installed, a user-written routine is given control (if the routine meets the format requirements). One use of ROMboot might be resetting SYSFAIL* on an unintelligent controller module. The **NORB** command disables the function.

For a user's ROMboot module to gain control through the ROMboot linkage, four requirements must be met:

- Power must have just been applied (but the **ENV** command can change this to also respond to any reset).
- Your routine must be located within the MVME172 Flash/PROM memory map (but the **ENV** command can change this to any other portion of the onboard memory, or even offboard VMEbus memory).
- The ASCII string "BOOT" must be located within the specified memory range.
- Your routine must pass a checksum test, which ensures that this routine was really intended to receive control at powerup.

For complete details on how to use ROMboot, refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual.*

# Network Boot

Network Auto Boot is a software routine contained in the 172Bug Flash/PROM that provides a mechanism for booting an operating system using a network (local Ethernet interface) as the boot device. The Network

Auto Boot routine automatically scans for controllers and devices in a specified sequence until a valid bootable device containing a boot media is found or the list is exhausted. If a valid bootable device is found, a boot from that device is started. The controller scanning sequence goes from the lowest controller Logical Unit Number (LUN) detected to the highest LUN detected. (Refer to *Appendix E* for default LUNs.)

At power-up, Network Boot is enabled, and providing the drive and controller numbers encountered are valid, the following message is displayed upon the system console:

```
"Network Boot in progress... To abort hit <BREAK>"
```

Following this message there is a delay to allow you to abort the Auto Boot process if you wish. Then the actual I/O is begun: the program pointed to within the volume ID of the media specified is loaded into RAM and control passed to it. If, however, during this time you want to gain control without Network Boot, you can press the <BREAK> key or the software ABORT or RESET switches.

Network Auto Boot is controlled by parameters contained in the **NIOT** and **ENV** commands. These parameters allow the selection of specific boot devices, systems, and files, and allow programming of the Boot delay. Refer to the **ENV** command in *Appendix A* for more details.

# Restarting the System

You can initialize the system to a known state in three different ways: reset, abort, and break. Each has characteristics which make it more appropriate than the others in certain situations.

The debugger has a special feature upon a reset condition. This feature is activated by depressing the RESET and ABORT switches at the same time. This feature instructs the debugger to use the default setup/operation parameters in ROM versus your setup/operation parameters in NVRAM. This feature can be used in the event your setup/operation parameters are corrupted or do not meet a sanity check. Refer to the **ENV** command in *Appendix A* for the ROM defaults.

## Reset



Pressing and quickly releasing the MVME172 front panel RESET button initiates a system reset. COLD and WARM reset modes are available. By default, 172Bug is in COLD mode. During COLD reset, a total system initialization takes place, as if the MVME172 had just been powered up. All static variables (including disk device and controller parameters) are restored to their default states. The breakpoint table and offset registers are cleared. The target registers are invalidated. Input and output character queues are cleared. Onboard devices (timer, serial ports, etc.) are reset, and the two serial ports are reconfigured to their default state.

During WARM reset, the 172Bug variables and tables are preserved, as well as the target state registers and breakpoints.

Reset must be used if the processor ever halts, or if the 172Bug environment is ever lost (vector table is destroyed, stack corrupted, etc.).

## Abort

Abort is invoked by pressing and releasing the ABORT switch on the MVME172 front panel. Whenever abort is invoked when executing a user program (running target code), a "snapshot" of the processor state is captured and stored in the target registers. For this reason, abort is most appropriate when terminating a user program that is being debugged. Abort should be used to regain control if the program gets caught in a loop, etc. The target PC, register contents, etc., help to pinpoint the malfunction.

Pressing and releasing the ABORT switch generates a local board condition which may interrupt the processor if enabled. The target registers, reflecting the machine state at the time the ABORT switch was pressed, are displayed on the screen. Any breakpoints installed in your code are removed and the breakpoint table remains intact. Control is returned to the debugger.

## Break

A "Break" is generated by pressing and releasing the BREAK key on the terminal keyboard. Break does not generate an interrupt. The only time break is recognized is when characters are sent or received by the console port. Break removes any breakpoints in your code and keeps the breakpoint table intact. Break also takes a snapshot of the machine state if the function was entered using SYSCALL. This machine state is then accessible to you for diagnostic purposes.

Many times it may be desirable to terminate a debugger command prior to its completion; for example, during the display of a large block of memory. Break allows you to terminate the command.

## SYSFAIL* Assertion/Negation

Upon a reset/powerup condition the debugger asserts the VMEbus SYSFAIL* line (refer to the VMEbus specification). SYSFAIL* stays asserted if any of the following has occurred:

- confidence test failure
- NVRAM checksum error
- NVRAM low battery condition
- local memory configuration status
- self test (if system mode) has completed with error
- MPU clock speed calculation failure

After debugger initialization is done and none of the above situations have occurred, the SYSFAIL* line is negated. This indicates to the user or VMEbus masters the state of the debugger. In a multi-computer configuration, other VMEbus masters could view the pertinent control and status registers to determine which CPU is asserting SYSFAIL*. SYSFAIL* assertion/negation is also affected by the **ENV** command. Refer to *Appendix A*.

## MPU Clock Speed Calculation

The clock speed of the microprocessor is calculated and checked against a user definable parameter housed in NVRAM (refer to the **CNFG** command in *Appendix A*). If the check fails, a warning message is displayed. The calculated clock speed is also checked against known clock speeds and tolerances.

# Memory Requirements



The program portion of 172Bug is approximately 512KB of code, consisting of download, debugger, and diagnostic packages and contained entirely in Flash or PROM.

The 172Bug executes from $FF800000 whether in Flash or PROM. With the jumper at J28 pins 9-10 installed (the factory ship configuration *except* in the no-VMEbus case), the Flash memory appears at address $FF800000 and is the part executed during reset. The PROM socket is mapped to address $FFA00000 with this configuration. If you remove the jumper at J28 pins 9 and 10, the address spaces of the Flash and PROM are swapped.

The 172Bug initial stack completely changes all 8KB of memory at addresses $FFE0C000 through $FFE0DFFF at power-up or reset.

| Type of Memory Present | Default DRAM Base Address | Default SRAM Base Address |
|---|---|---|
| Single DRAM mezzanine | $00000000 | $FFE00000 (onboard SRAM) |

DRAM is neither ECC nor parity type, but unprotected.

The 172Bug requires 2KB of NVRAM for storage of board configuration, communication, and booting parameters. This storage area begins at $FFFC16F8 and ends at $FFFC1EF7.

172Bug requires a minimum of 64KB of contiguous read/write memory to operate. The **ENV** command controls where this block of memory is located. Regardless of where the onboard RAM is located, the first 64KB is used for 172Bug stack and static variable space and the rest is reserved as user space. Whenever the MVME172 is reset, the target PC is initialized to the address cor- responding to the beginning of the user space, and the target stack pointers are initialized to addresses within the user space, with the target Interrupt Stack Pointer (ISP) set to the top of the user space.

# Disk I/O Support



172Bug can initiate disk input/output by communicating with intelligent disk controller modules over the VMEbus. Disk support facilities built into 172Bug consist of command-level disk operations, disk I/O system calls (only via one of the TRAP #15 instructions) for use by user programs, and defined data structures for disk parameters.

Parameters such as the address where the module is mapped and the type and number of devices attached to the controller module are kept in tables by 172Bug. Default values for these parameters are assigned at power-up and cold-start reset, but may be altered as described in the section on default parameters, later in this chapter.

Appendix E contains a list of the controllers presently supported, as well as a list of the default configurations for each controller.

## Blocks Versus Sectors

The logical block defines the unit of information for disk devices. A disk is viewed by 172Bug as a storage area divided into logical blocks. By default, the logical block size is set to 256 bytes for every block device in the system. The block size can be changed on a per device basis with the **IOT** command.

The sector defines the unit of information for the media itself, as viewed by the controller. The sector size varies for different controllers, and the value for a specific device can be displayed and changed with the **IOT** command.

When a disk transfer is requested, the start and size of the transfer is specified in blocks. 172Bug translates this into an equivalent sector specification, which is then passed on to the controller to initiate the transfer. If the conversion from blocks to sectors yields a fractional sector count, an error is returned and no data is transferred.

## Device Probe Function

A device probe with entry into the device descriptor table is done whenever a specified device is accessed; i.e., when system calls .DSKRD, .DSKWR, .DSKCFIG, .DSKFMT, and .DSKCTRL, and debugger commands **BH**, **BO**, **IOC**, **IOP**, **IOT**, **MAR**, and **MAW** are used.

The device probe mechanism utilizes the SCSI commands "Inquiry" and "Mode Sense". If the specified controller is non-SCSI, the probe simply returns a status of "device present and unknown". The device probe makes an entry into the device descriptor table with the pertinent data. After an entry has been made, the next time a probe is done it simply returns with "device present" status (pointer to the device descriptor).

## Disk I/O via 172Bug Commands

These following 172Bug commands are provided for disk I/O. Detailed instructions for their use are found in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*. When a command is issued to a particular controller LUN and device LUN, these LUNs are remembered by 172Bug so that the next disk command defaults to use the same controller and device.

### IOI (Input/Output Inquiry)

This command is used to probe the system for all possible CLUN/DLUN combinations and display inquiry data for devices which support it. The device descriptor table only has space for 16 device descriptors; with the **IOI** command, you can view the table and clear it if necessary.

### IOP (Physical I/O to Disk)

**IOP** allows you to read or write blocks of data, or to format the specified device in a certain way. **IOP** creates a command packet from the arguments you have specified, and then invokes the proper system call function to carry out the operation.

### IOT (I/O Teach)

**IOT** allows you to change any configurable parameters and attributes of the device. In addition, it allows you to see the controllers available in the system.

### IOC (I/O Control)

**IOC** allows you to send command packets as defined by the particular controller directly. **IOC** can also be used to look at the resultant device packet after using the **IOP** command.

### BO (Bootstrap Operating System)

**BO** reads an operating system or control program from the specified device into memory, and then transfers control to it.

### BH (Bootstrap and Halt)

**BH** reads an operating system or control program from a specified device into memory, and then returns control to 172Bug. It is used as a debugging tool.

## Disk I/O via 172Bug System Calls

All operations that actually access the disk are done directly or indirectly by 172Bug TRAP #15 system calls. (The command-level disk operations provide a convenient way of using these system calls without writing and executing a program.)

The following system calls are provided to allow user programs to do disk I/O:

| | |
|---|---|
| .DSKRD | Disk read. System call to read blocks from a disk into memory. |
| .DSKWR | Disk write. System call to write blocks from memory onto a disk. |
| .DSKCFIG | Disk configure. This function allows you to change the configuration of the specified device. |
| .DSKFMT | Disk format. This function allows you to send a format command to the specified device. |
| .DSKCTRL | Disk control. This function is used to implement any special device control functions that cannot be accommodated easily with any of the other disk functions. |

Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for information on using these and other system calls.

To perform a disk operation, 172Bug must eventually present a particular disk controller module with a controller command packet which has been especially prepared for that type of controller module. (This is accomplished in the respective controller driver module.) A command packet for one type of controller module usually does not have the same format as a command packet for a different type of module. The system call facilities which do disk I/O accept a generalized (controller-independent) packet format as an argument, and translate it into a controller-specific packet, which is then sent to the specified device. Refer to the system call descriptions in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for details on the format and construction of these standardized "user" packets.

The packets which a controller module expects to be given vary from controller to controller. The disk driver module for the particular hardware module (board) must take the standardized packet given to a trap function and create a new packet. This new packet is specifically tailored for the disk drive controller for which it is sent. Refer to documentation on the particular controller module for the format of its packets, and for using the **IOC** command.

## Default 172Bug Controller and Device Parameters

172Bug initializes the parameter tables for a default configuration of controllers and devices (refer to *Appendix E*). If the system needs to be configured differently than this default configuration (for example, to use a 70MB Winchester drive where the default is a 40MB Winchester drive), then these tables must be changed.

There are three ways to change the parameter tables:

- Using **BO** or **BH**. When you invoke one of these commands, the configuration area of the disk is read and the parameters corresponding to that device are rewritten according to the parameter information contained in the configuration area. This is a temporary change. If a cold-start reset occurs, then the default parameter information is written back into the tables.
- Using the **IOT**. You can use this command to reconfigure the parameter table manually for any controller and/or device that is different from the default. This is also a temporary change and is overwritten if a cold-start reset occurs.

## Disk I/O Error Codes

172Bug returns an error code if an attempted disk operation is unsuccessful.

# Network I/O Support



The Network Boot Firmware provides the capability to boot the CPU through the Flash/PROM debugger using a network (local Ethernet interface) as the boot device.

The booting process is executed in two distinct phases.

- The first phase allows the diskless remote node to discover its network identify and the name of the file to be booted.
- The second phase has the diskless remote node reading the boot file across the network into its memory.

The various modules (capabilities) and the dependencies of these modules that support the overall network boot function are described in the following paragraphs.

## Intel 82596 LAN Coprocessor Ethernet Driver

This driver manages/surrounds the Intel 82596 LAN Coprocessor. Management is in the scope of the reception of packets, the transmission of packets, receive buffer flushing, and interface initialization.

This module ensures that the packaging and unpackaging of Ethernet packets is done correctly in the Boot PROM.

## UDP/IP Protocol Modules

The Internet Protocol (IP) is designed for use in interconnected systems of packet-switched computer communication networks. The Internet Protocol provides for transmitting of blocks of data called datagrams (hence User Datagram Protocol, or UDP) from sources to destinations, where sources and destinations are hosts identified by fixed length addresses.

The UDP/IP protocols are necessary for the TFTP and BOOTP protocols; TFTP and BOOTP require a UDP/IP connection.

## RARP/ARP Protocol Modules

The Reverse Address Resolution Protocol (RARP) basically consists of an identity-less node broadcasting a "whoami" packet onto the Ethernet, and waiting for an answer. The RARP server fills an Ethernet reply packet up with the target's Internet Address and sends it.

The Address Resolution Protocol (ARP) basically provides a method of converting protocol addresses (e.g., IP addresses) to local area network addresses (e.g., Ethernet addresses). The RARP protocol module supports systems which do not support the BOOTP protocol (next paragraph).

## BOOTP Protocol Module

The Bootstrap Protocol (BOOTP) basically allows a diskless client machine to discover its own IP address, the address of a server host, and the name of a file to be loaded into memory and executed.

## TFTP Protocol Module

The Trivial File Transfer Protocol (TFTP) is a simple protocol to transfer files. It is implemented on top of the Internet User Datagram Protocol (UDP or Datagram) so it may be used to move files between machines on different networks implementing UDP. The only thing it can do is read and write files from/to a remote server.

## Network Boot Control Module

The "control" capability of the Network Boot Control Module is needed to tie together all the necessary modules (capabilities) and to sequence the booting process. The booting sequence consists of two phases: the first phase is labeled "address determination and bootfile selection" and the second phase is labeled "file transfer". The first phase will utilize the RARP/BOOTP capability and the second phase will utilize the TFTP capability.

## Network I/O Error Codes

172Bug returns an error code if an attempted network operation is unsuccessful.



# Multiprocessor Support

The MVME172 dual-port RAM feature makes the shared RAM available to remote processors as well as to the local processor. This can be done by either of the following two methods. Either method can be enabled/disabled by the **ENV** command as its Remote Start Switch Method (refer to *Appendix A*).

## Multiprocessor Control Register (MPCR) Method

A remote processor can initiate program execution in the local MVME172 dual-port RAM by issuing a remote **GO** command using the Multiprocessor Control Register (MPCR). The MPCR, located at shared RAM location of $800 offset from the base address the debugger loads it at, contains one of two longwords used to control communication between processors. The MPCR contents are organized as follows:

| | | | | | |
|---|---|---|---|---|---|
| $800 | * | N/A | N/A | N/A | (MPCR) |

The status codes stored in the MPCR are of two types:

- Status returned (from the monitor)
- Status set (by the bus master)

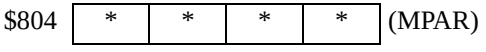
The status codes that may be returned from the monitor are:

| | | | | |
|---|---|---|---|---|
| HEX | 0 | (HEX 00) | -- | Wait. Initialization not yet complete. |
| ASCII | E | (HEX 45) | -- | Code pointed to by the MPAR address is executing. |
| ASCII | P | (HEX 50) | -- | Program Flash Memory. The MPAR is set to the address of the Flash memory program control packet. |
| ASCII | R | (HEX 52) | -- | Ready. The firmware monitor is watching for a change. |

You can only program Flash memory by the MPCR method. Refer to the .PFLASH system call in the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for a description of the Flash memory program control packet structure. The status codes that may be set by the bus master are:

| | | | | |
|---|---|---|---|---|
| ASCII | G | (HEX 47) | -- | Use Go Direct (**GD**) logic specifying the MPAR address. |
| ASCII | B | (HEX 42) | -- | Install breakpoints using the Go (**G**) logic. |

The Multiprocessor Address Register (MPAR), located in shared RAM location of $804 offset from the base address the debugger loads it at, contains the second of two longwords used to control communication between processors. The MPAR contents specify the address at which execution for the remote processor is to begin if the MPCR contains a G or B. The MPAR is organized as follows:




At power-up, the debug monitor self-test routines initialize RAM, including the memory locations used for multi-processor support ($800 through $807).

The MPCR contains $00 at power-up, indicating that initialization is not yet complete. As the initialization proceeds, the execution path comes to the "prompt" routine. Before sending the prompt, this routine places an R in the MPCR to indicate that initialization is complete. Then the prompt is sent.

If no terminal is connected to the port, the MPCR is still polled to see whether an external processor requires control to be passed to the dual-port RAM. If a terminal does respond, the MPCR is polled for the same purpose while the serial port is being polled for user input.

An ASCII G placed in the MPCR by a remote processor indicates that the Go Direct type of transfer is requested. An ASCII B in the MPCR indicates that breakpoints are to be armed before control is transferred (as with the **GO** command).

In either sequence, an E is placed in the MPCR to indicate that execution is underway just before control is passed to RAM. (Any remote processor could examine the MPCR contents.)

If the code being executed in dual-port RAM is to reenter the debug monitor, a TRAP #15 call using function $0063 (SYSCALL .RETURN) returns control to the monitor with a new display prompt. Note that every time the debug monitor returns to the prompt, an R is moved into the MPCR to indicate that control can be transferred once again to a specified RAM location.

## GCSR Method

A remote processor can initiate program execution in the local MVME172 dual-port RAM by issuing a remote **GO** command using the VMEchip2 Global Control and Status Registers (GCSR). The remote processor places the MVME172 execution address in general purpose registers 0 and 1 (GPCSR0 and GPCSR1). The remote processor then sets bit 8 (SIG0) of the VMEchip2 LM/SIG register. This causes the MVME172 to install breakpoints and begin execution. The result is identical to the MPCR method (with status code B) described in the previous section.

The GCSR registers are accessed in the VMEbus short I/O space. Each general purpose register is two bytes wide, occurring at an even address. The general purpose register number 0 is at an offset of $8 (local bus) or $4 (VMEbus) from the start of the GCSR registers. The local bus base address for the GCSR is $FFF40100. The VMEbus base address for the GCSR depends on the group select value and the board select value programmed in the Local Control and Status Registers (LCSR) of the MVME172. The execution address is formed by reading the GCSR general purpose registers in the following manner:

GPCSR0 used as the upper 16 bits of the address
GPCSR1 used as the lower 16 bits of the address

The address appears as:

| GPCSR0 | GPCSR1 |
|---|---|

# Diagnostic Facilities

The 172Bug package includes a set of hardware diagnostics for testing and troubleshooting the MVME172. To use the diagnostics, switch directories to the diagnostic directory. If you are in the debugger directory, you can switch to the diagnostic directory with the debugger command Switch Directories (**SD**). The diagnostic prompt ("`172-Diag>`") appears. Refer to the *MVME172Bug Diagnostics Manual* for complete descriptions of the diagnostic routines available and instructions on how to invoke them. Note that some diagnostics depend on restart defaults that are set up only in a particular restart mode. The documentation for such diagnostics includes restart information.

# Manufacturing Test Process

During the manufacturing process for MVME172 modules, the manufacturing test parameters and testing state flags are stored in NVRAM. These strings are installed during the manufacturing process and

result in the product performing manufacturing tests. None of these tests harm the product or system into which a module is installed. Entering an ASCII break on the console port from a terminal terminates these tests.



The two state flags that start the test processes are:

```
FLASH EMPTY$00122984
```

and

```
Burnin test$00000000
```

If either string is in the first location of NVRAM ($FFFC0000), the test process starts.

# Using the 172Bug Debugger 4

## This Chapter Covers

- Entering debugger command lines
- Entering and debugging programs
- Calling system utilities from user programs
- Preserving the debugger operating environment
- Floating point support
- The 172Bug debugger command set

## Entering Debugger Command Lines

172Bug is command-driven and performs its various operations in response to user commands entered at the keyboard. When the debugger prompt

```
172-Bug>
```

appears on the terminal screen, then the debugger is ready to accept commands.

### Terminal Input/Output Control

As the command line is entered, it is stored in an internal buffer. Execution begins only after the carriage return is entered, so that you can correct entry errors, if necessary, using the control characters described below.

**Note** The presence of the upward caret ( ^ ) before a character indicates that the Control (CTRL) key must be held down while striking the character key.



| | | |
|---|---|---|
| ^X | (cancel line) | The cursor is backspaced to the beginning of the line. |
| ^H | (backspace) | The cursor is moved back one position. |
| Delete key | (delete) | Performs the same function as **^H**. |
| ^D | (redisplay) | The entire command line as entered so far is redisplayed on the following line. |
| ^A | (repeat) | Repeats the previous line. This happens only at the command line. The last line entered is redisplayed but not executed. The cursor is positioned at the end of the line. You may enter the line as is or you can add more characters to it. You can edit the line by backspacing and typing over old characters. |

When observing output from any 172Bug command, the XON and XOFF characters which are in effect for the terminal port may be entered to control the output, if the XON/XOFF protocol is enabled (default). These characters are initialized to **^S** and **^Q** respectively by 172Bug, but you may change them with the **PF** command. In the initialized (default) mode, operation is as follows:

| | | |
|---|---|---|
| ^S | (wait) | Console output is halted. |
| ^Q | (resume) | Console output is resumed. |

When a command is entered, the debugger executes the command and the prompt reappears. However, if the command entered causes execution of user target code, for example **GO**, then control may or may not return to the debugger, depending on what the user program does.

For example, if a breakpoint has been specified, then control returns to the debugger when the breakpoint is encountered during execution of the user program. Alternately, the user program could return to the debugger by means of the TRAP #15 function ".RETURN".

## Debugger Command Syntax

In general, a debugger command is made up of the following parts:

- The command identifier (i.e., **MD** or **md** for the Memory Display command). Note that either upper- or lowercase is allowed.
- A port number, if the command is set up to work with more than one port.
- At least one intervening space before the first argument.
- Any required arguments, as specified by the command.
- An option field, set off by a semicolon (**;**) to specify conditions other than the default conditions of the command.

The commands are shown using a modified Backus-Naur form syntax. The metasymbols used are:



## Syntactic Variables

The following syntactic variables are encountered in the command descriptions which follow. In addition, other syntactic variables may be used and are defined in the particular command description in which they occur.

| | |
|---|---|
| *exp* | Expression (described in detail in a following section). |
| *addr* | Address (described in detail in a following section). |
| *count* | Count; the syntax is the same as for *exp*. |
| *range* | A range of memory addresses which may be specified either by *addr addr* or by *addr*: *count*. |
| *text* | An ASCII string of up to 255 characters, delimited at each end by the single quote mark (’). |

### Expression as a Parameter

An expression can be one or more numeric values separated by the arithmetic operators: plus (+), minus (-), multiplied by (*****), divided by (/), logical AND (**&**), shift left (<<), or shift right (>>).

Numeric values may be expressed in either hexadecimal, decimal, octal, or binary by immediately preceding them with the proper base identifier.

| Base | Identifier | Examples |
|---|---|---|
| Hexadecimal | $ | $FFFFFFFF |
| Decimal | & | &1974, &10-&4 |
| Octal | @ | @456 |
| Binary | % | %1000110 |

If no base identifier is specified, then the numeric value is assumed to be hexadecimal.

A numeric value may also be expressed as a string literal of up to four characters. The string literal must begin and end with the single quote mark (’). The numeric value is interpreted as the concatenation of the ASCII values of the characters. This value is right-justified, as any other numeric value would be.

| String Literal | Numeric Value (In Hexadecimal) |
|---|---|
| ’A’ | 41 |
| ’ABC’ | 414243 |
| ’TEST’ | 54455354 |

Evaluation of an expression is always from left to right unless parentheses are used to group part of the expression. There is no operator precedence. Subexpressions within parentheses are evaluated first. Nested parenthetical subexpressions are evaluated from the inside out.

Valid expression examples:

| Expression | Result (In Hex) | Notes |
|---|---|---|
| FF0011 | FF0011 | |
| 45+99 | DE | |
| &45+&99 | 90 | |
| @35+@67+@10 | 5C | |
| %10011110+%1001 | A7 | |
| 88<<4 | 880 | shift left |

| Expression | Result (In Hex) | Notes |
|---|---|---|
| AA&F0 | A0 | logical AND |

The total value of the expression must be between 0 and $FFFFFFFF.

## Address as a Parameter



Many commands use *addr* as a parameter. The syntax accepted by 172Bug is similar to the one accepted by the MC68060 one-line assembler. All control addressing modes are allowed. An "address + offset register" mode is also provided.

## Address Formats

Table 4-1 summarizes the address formats that are acceptable for address parameters in debugger command lines.

**Table 4-1. Debugger Address Parameter Formats**

| Format | Example | Description |
|---|---|---|
| *N* | 140 | Absolute address+contents of automatic offset register. |
| *N*+R*n* | 130+R5 | Absolute address+contents of the specified offset register (not an assembler-accepted syntax). |
| (A*n*) | (A1) | Address register indirect. (Also post-increment, pre-decrement) |
| (*d*,A*n*) or *d*(A*n*) | (120,A1)<br>120(A1) | Address register indirect with displacement (two formats accepted). |
| (*d*,A*n*,*Xn*) or *d*(A*n*,*Xn*) | (&120,A1,D2)<br>&120(A1,D2) | Address register indirect with index and displacement (two formats accepted). |
| ([*bd*,A*n*,*Xn*],*od*) | ([C,A2,A3],&100) | Memory indirect preindexed. |
| ([*bd*,A*n*],X*n*,*od*) | ([12,A3],D2,&10) | Memory indirect postindexed. |
| For the memory indirect modes, fields can be omitted.<br>For example, three of many permutations are as follows: | | |

**Table 4-1. Debugger Address Parameter Formats**

| Format | Example | Description |
|---|---|---|
| ([,A*n*],*od*) | ([,A1],4) | |
| ([*bd*]) | ([FC1E]) | |
| ([*bd*,,*Xn*]) | ([8,,D2]) | |



Notes

*N* — Absolute address (any valid expression).
A*n* — Address register *n*.
*Xn* — Index register *n* (A*n* or D*n*).
*d* — Displacement (any valid expression).
*bd* — Base displacement (any valid expression).
*od* — Outer displacement (any valid expression).
*n* — Register number (0 to 7).
R*n* — Offset register n.

**Note** In commands with *range* specified as *addr addr*, and with size option W or L chosen, data at the second (ending) address is acted on only if the second address is a proper boundary for a word or longword, respectively.

## Offset Registers

Eight pseudo-registers (R0 through R7) called offset registers are used to simplify the debugging of relocatable and position-independent modules. The listing files in these types of programs usually start at an address (normally 0) that is not the one at which they are loaded, so it is harder to correlate addresses in the listing with addresses in the loaded program. The offset registers solve this problem by taking into account this difference and forcing the display of addresses in a relative address+offset format. Offset registers have adjustable ranges and may even have overlapping ranges. The range for each offset register is set by two addresses: base and top. Specifying the base and top addresses for an offset register sets its range. In the event that an address falls in two or more offset registers' ranges, the one that yields the least offset is chosen.

**Note** Relative addresses are limited to 1MB (5 digits), regardless of the range of the closest offset register.

Example: A portion of the listing file of an assembled, relocatable module is shown below:

```
1
2                       *
3                       *  MOVE STRING SUBROUTINE
4                       *
5   0 00000000 48E78080    MOVESTR MOVEM.L   D0/A0,-(A7)
6   0 00000004 4280                CLR.L     D0
7   0 00000006 1018                MOVE.B    (A0)+,D0
8   0 00000008 5340                SUBQ.W    #1,D0
9   0 0000000A 12D8        LOOP    MOVE.B    (A0)+,(A1)+
10  0 0000000C 51C8FFFC    MOVS    DBRA      D0,LOOP
11  0 00000010 4CDF0101            MOVEM.L   (A7)+,D0/A0
12  0 00000014 4E75                RTS
13
14                      END     END
     ******  TOTAL ERRORS     0--
     ******  TOTAL WARNINGS   0--
```

The above program was loaded at address $0001327C.

The disassembled code is shown next:

```
172Bug>MD 1327C;DI
0001327C 48E78080                MOVEM.L  D0/A0,-(A7)
00013280 4280                    CLR.L    D0
00013282 1018                    MOVE.B   (A0)+,D0
00013284 5340                    SUBQ.W   #1,D0
00013286 12D8                    MOVE.B   (A0)+,(A1)+
00013288 51C8FFFC                DBF      D0,$13286
0001328C 4CDF0101                MOVEM.L  (A7)+,D0/A0
00013290 4E75                    RTS
172Bug>
```

By using one of the offset registers, the disassembled code addresses can be made to match the listing file addresses as follows:

```
172Bug>OF R0
R0 =00000000 00000000? 1327C. <CR>
172Bug>MD 0+R0;DI <CR>
00000+R0 48E78080                MOVEM.L  D0/A0,–(A7)
00004+R0 4280                    CLR.L    D0
00006+R0 1018                    MOVE.B   (A0)+,D0
00008+R0 5340                    SUBQ.W   #1,D0
0000A+R0 12D8                    MOVE.B   (A0)+,(A1)+
0000C+R0 51C8FFFC                DBF      D0,$A+R0
00010+R0 4CDF0101                MOVEM.L  (A7)+,D0/A0
00014+R0 4E75                    RTS
172Bug>
```

For additional information about the offset registers, refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual*.

## Port Numbers

Some 172Bug commands give you the option to choose the port to be used to input or output. Valid port numbers which may be used for these commands are as follows:

- MVME172 EIA-232-D Debug (Terminal Port 0 or 00) (PORT 1 on the MVME172 P2 connector). Sometimes known as the "console port", it is used for interactive user input/output by default.
- MVME172 EIA-232-D (Terminal Port 1 or 01) (PORT 2 on the MVME172 P2 connector). Sometimes known as the "host port", this is the default for downloading, uploading, concurrent mode, and transparent modes.

**Note** These logical port numbers (0 and 1) are shown in the pinouts of the MVME172 module as "SERIAL PORT 1" and "SERIAL PORT 2", respectively. Physically, they are all part of connector P2. They are also available at the front panel DB-25 connectors J25 (for PORT 1 or A) and J18 (for PORT 2 or B).

# Entering and Debugging Programs



There are various ways to enter a user program into system memory for execution:

- Create the program with the assembler/disassembler
- Download an S-record object file
- Read the program from disk

## Creating a Program with the Assembler/Disassembler

You can create a program using the Memory Modify (**MM**) command with the assembler/disassembler option.

1. Enter the program one source line at a time.
2. After each source line is entered, it is assembled and the object code is loaded to memory.

Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for complete details on the 172Bug Assembler/Disassembler.

## Downloading an S-Record Object File

Another way to enter a program is to download an object file from a host system.

The program must be in S-record format (described in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*) and may have been assembled or compiled on the host system.

Alternately, the program may have been previously created using the 172Bug **MM** command as outlined above and stored to the host using the Dump (**DU**) command.

A communication link must exist between the host system and the MVME172 port 1. (Hardware configuration details are provided in *Installation and Startup*.) The file is downloaded from the host to MVME172 memory by the Load (**LO**) command.

## Read the Program from Disk

Another way to enter a program is by reading the program from disk, using one of the disk commands (**BO**, **BH**, **IOP**). Once the object code has been loaded into memory, you can set breakpoints if desired and run the code or trace through it.



# Calling System Utilities from User Programs

A convenient way of doing character input/output and many other useful operations has been provided so that you do not have to write these routines into the target code. You can access various 172Bug routines via one of the MC68060 TRAP instructions, using vector #15. Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for details on the various TRAP #15 utilities available and how to invoke them from within a user program.

# Preserving the Debugger Operating Environment



This section explains how to avoid contaminating the operating environment of the debugger. Topics covered include:

- 172Bug Vector Table and workspace
- Hardware functions
- Exception vectors used by 172Bug

172Bug uses certain of the MVME172 onboard resources and may also use offboard system memory to contain temporary variables, exception vectors, etc. If you disturb resources upon which 172Bug depends, then the debugger may function unreliably or not at all.

If your application enables translation through the Memory Management Units (MMUs), and if your application utilizes resources of the debugger (e.g., system calls), your application must create the necessary translation tables for the debugger to have access to its various resources. The debugger honors the enabling of the MMUs; it does not disable translation.

## 172Bug Vector Table and Workspace

As described in *Memory Requirements*, 172Bug needs 64KB of read/write memory to operate.

| 172Bug reserves ... | For ... |
|---|---|
| 1024-byte area | A user program vector table area |
| 1024-byte area | An exception vector table for the debugger itself to use |
| Space for static variables and initializes these static variables to predefined default values. | |
| Space for the system stack then initializes the system stack pointer to the top of this area. | |

With the exception of the first 1024-byte vector table area, you must be extremely careful not to use the above-mentioned memory areas for other purposes.

Refer to *Memory Requirements* on page 3-15 to determine how to dictate the location of the reserved memory areas.



### Examples

- If, for example, your program inadvertently wrote over the static variable area containing the serial communication parameters, these parameters would be lost, resulting in a loss of communication with the system console terminal.
- If your program corrupts the system stack, then an incorrect value may be loaded into the processor Program Counter (PC), causing a system crash.

## Hardware Functions

The only hardware resources used by the debugger are the EIA-232-D ports, which are initialized to interface to the debug terminal and a host. If these ports are reprogrammed, the terminal characteristics must be modified to suit, or the ports should be restored to the debugger-set characteristics prior to reinvoking the debugger.

## Exception Vectors Used by 172Bug

The exception vectors used by the debugger are listed below. These vectors must reside at the specified offsets in the target program's vector table for the associated debugger facilities (breakpoints, trace mode, etc.) to operate.



**Table 4-2. Exception Vectors Used by 172Bug**

| Vector Offset | Exception | 172Bug Facility |
|---|---|---|
| $10 | Illegal instruction | Breakpoints (used by **GO**, **GN**, **GT**) |
| $24 | Trace | Trace operations (such as **T**, **TC**, **TT**) |
| $80-$B8 | TRAP #0 - #14 | Used internally |
| $BC | TRAP #15 | System calls |
| $ Note 1 | Level 7 interrupt | ABORT pushbutton |
| $ Note 2 | Level 7 interrupt | AC Fail |
| $DC | FP Unimplemented Data Type | Software emulation and data type conversion of floating point data. |

**Notes**

1. This depends on what the Vector Base Register (VBR) is set to in the MC2chip.
2. This depends on what the Vector Base Register (VBR) is set to in the VMEchip2.

When the debugger handles one of the exceptions listed in Table 4-2, the target stack pointer is left pointing past the bottom of the exception stack frame created; that is, it reflects the system stack pointer values just before the exception occurred. In this way, the operation of the debugger facility (through an exception) is transparent to users.

Example: Trace one instruction using debugger.

```
172-Bug>rd
PC    =00010000   SR    =2708=TR:OFF_S._7_N..          VBR   =00000000
SSP   =0000FFFC   USP   =00010000   SFC    =1=UD       DFC   =1=UD
CACR  =00000000=D: ....._B:..._I:...                   PCR   =04310402
D0    =FFFFFFFF   D1    =00000000   D2     =00000000   D3    =00000000
D4    =00000000   D5    =00000000   D6     =00000000   D7    =00000000
A0    =00000000   A1    =00000000   A2     =00000000   A3    =00000000
A4    =00000000   A5    =00000000   A6     =00000000   A7    =0000FFFC
IPLR  =00000007   IMLR  =00000000   MMIEN  =00000003   VIEN  =C0000000
VIST  =00000000   PIEN  =00002000   PIST   =00000000
00010000 203C0000 0001   MOVE.L   #$1,D0
172-Bug>t
PC    =00010006   SR    =2700=TR:OFF_S._7_.....       VBR   =00000000
SSP*  =0000FFFC   USP   =00010000   SFC    =1=UD       DFC   =1=UD
CACR  =00000000=D: ....._B:..._I:...                   PCR   =04310402
D0    =00000001   D1    =00000000   D2     =00000000   D3    =00000000
D4    =00000000   D5    =00000000   D6     =00000000   D7    =00000000
A0    =00000000   A1    =00000000   A2     =00000000   A3    =00000000
A4    =00000000   A5    =00000000   A6     =00000000   A7    =0000FFFC
IPLR  =00000007   IMLR  =00000000   MMIEN  =00000003   VIEN  =C0000000
VIST  =00000000   PIEN  =00002000   PIST   =00000000
00010006 D280 0001       ADD.L      D0,D1
172-Bug>
```

## Exception Vector Tables

Notice in the preceding example that the value of the target stack pointer register (A7) has not changed even though a trace exception has taken place. Your program may either use the exception vector table provided by 172Bug or it may create a separate exception vector table of its own. The two following sections detail these two methods.

## Using 172Bug Target Vector Table

The 172Bug initializes and maintains a vector table area for target programs. A target program is any program started by the bug:

- Manually with **GO** command
- Manually with trace commands (**T**, **TC**, or **TT**)
- Automatically with the **BO** command.

The start address of this target vector table area is the base address ($00) of the debugger memory. This address is loaded into the target-state VBR at powerup and cold-start reset and can be observed by using the **RD** command to display the target-state registers immediately after powerup.

The 172Bug initializes the target vector table with the debugger vectors listed in Table 4-2 on page 4-14 and fills the other vector locations with the address of a generalized exception handler. The target program may take over as many vectors as desired by simply writing its own exception vectors into the table. If the vector locations listed in Table 4-2 are overwritten then the accompanying debugger functions are lost.

The 172Bug maintains a separate vector table for its own use. In general, you do not have to be aware of the existence of the debugger vector table. It is completely transparent and you should never make any modifications to the vectors contained in it.

### Creating a New Vector Table

Your program may create a separate vector table in memory to contain its exception vectors. If this is done, the program must change the value of the VBR to point at the new vector table. In order to use the debugger facilities you can copy the proper vectors from the 172Bug vector table into the corresponding vector locations in your program vector table.

The vector for the 172Bug generalized exception handler may be copied from offset $08 (bus error vector) in the target vector table to all locations in your program vector table where a separate exception handler is not used. This provides diagnostic support in the event that your program is stopped by an unexpected exception. The generalized exception handler gives a formatted display of the target registers and identifies the type of the exception.

The following is an example of a routine which builds a separate vector table and then moves the VBR to point at it:

```
*
***  BUILDX - Build exception vector table ****
*
BUILDX  MOVEC.L  VBR,A0            Get copy of VBR.
        LEA      $10000,A1         New vectors at $10000.
        MOVE.L   $80(A0),D0        Get generalized exception vector.
        MOVE.W   $3FC,D1           Load count (all vectors).
LOOP    MOVE.L   D0,(A1,D1)       Store generalized exception vector.
        SUBQ.W   #4,D1
        BNE.B    LOOP              Initialize entire vector table.
        MOVE.L   $10(A0),$10(A1)   Copy breakpoints vector.
        MOVE.L   $24(A0),$24(A1)   Copy trace vector.
        MOVE.L   $BC(A0),$BC(A1)   Copy system call vector.
        LEA.L    COPROCC(PC),A2    Get your exception vector.
        MOVE.L   A2,$2C(A1)        Install as F-Line handler.
        MOVEC.L  A1,VBR            Change VBR to new table.
        RTS
        END
```

It may turn out that your program uses one or more of the exception vectors that are required for debugger operation. Debugger facilities may still be used, however, if your exception handler can determine when to handle the exception itself and when to pass the exception to the debugger.

When an exception occurs which you want to pass on to the debugger; i.e., **ABORT**, your exception handler must read the vector offset from the format word of the exception stack frame. This offset is added to the address of the 172Bug target program vector table (which your program saved), yielding the address of the 172Bug exception vector. The program then jumps to the address stored at this vector location, which is the address of the 172Bug exception handler.

Your program must make sure that there is an exception stack frame in the stack and that it is exactly the same as the processor would have created for the particular exception before jumping to the address of the exception handler.

The following is an example of an exception handler which can pass an exception along to the debugger:

```
*
***  EXCEPT - Exception handler  ****
*
EXCEPT  SUBQ.L   #4,A7              Save space in stack for a PC value.
        LINK     A6,#0              Frame pointer for accessing PC space.
        MOVEM.L  A0-A5/D0-D7,-(SP)  Save registers.
        :
        : decide here if your code handles exception, if so, branch...
        :
        MOVE.L   BUFVBR,A0          Pass exception to debugger; Get saved VBR.
        MOVE.W   14(A6),D0          Get the vector offset from stack frame.
        AND.W    #$0FFF,D0          Mask off the format information.
        MOVE.L   (A0,D0.W),4(A6)    Store address of debugger exc handler.
        MOVEM.L  (SP)+,A0-A5/D0-D7  Restore registers.
        UNLK     A6
        RTS                         Put addr of exc handler into PC and go.
```



# Floating Point Support

The floating point unit (FPU) of the MC68060 microprocessor chip is supported in 172Bug. The **MD**, **MM**, **RM**, and **RS** commands have been extended to allow display and modification of floating point data in registers and in memory. Floating point instructions can be assembled and disassembled with the **DI** option of the **MD** and **MM** commands.

**RM** and **RS** for floating point registers accept the floating point value in Double Precision Real Format or Scientific Notation.

Valid data types that can be used when modifying a floating point data register or a floating point memory location:

| Integer Data Types | |
|---|---|
| 12 | Byte |
| 1234 | Word |
| 12345678 | Longword |



| Floating Point Data Types | |
|---|---|
| 1_FF_7FFFFF | Single Precision Real Format |
| 1_7FF_FFFFFFFFFFFF | Double Precision Real Format |
| -3.12345678901234501_E+123 | Scientific Notation Format (decimal) |

When entering data in single or double precision, the following rules must be observed:

1. The sign field is the first field and is a binary field.
2. The exponent field is the second field and is a hexadecimal field.
3. The mantissa field is the last field and is a hexadecimal field.
4. The sign field, the exponent field, and at least the first digit of the mantissa field must be present (any unspecified digits in the mantissa field are set to zero).
5. Each field must be separated from adjacent fields by an underscore.
6. All the digit positions in the sign and exponent fields must be present.

## Single Precision Real

This format would appear in memory as:

| | |
|---|---|
| 1-bit sign field | (1 binary digit) |

| | |
|---|---|
| 8-bit biased exponent field | (2 hex digits. Bias = $7F) |
| 23-bit fraction field | (6 hex digits) |

A single precision number takes 4 bytes in memory.



## Double Precision Real

This format would appear in memory as:

| | |
|---|---|
| 1-bit sign field | (1 binary digit) |
| 11-bit biased exponent field | (3 hex digits. Bias = $3FF) |
| 52-bit fraction field | (13 hex digits) |

A double precision number takes 8 bytes in memory.

**Note** The single and double precision formats have an implied integer bit (always 1).

## Scientific Notation

This format provides a convenient way to enter and display a floating point decimal number. Internally, the number is assembled into a packed decimal number and then converted into a number of the specified data type.



Entering data in this format requires the following fields:

An optional sign bit (+ or -).
One decimal digit followed by a decimal point.
Up to 17 decimal digits (at least one must be entered).
An optional Exponent field that consists of:

- An optional underscore.
- The Exponent field identifier, letter “E”.
- An optional Exponent sign (+, -).
- From 1 to 3 decimal digits.

For more information about the MC68060 floating point unit, refer to the *MC68060 Microprocessor User’s Manual.*

# The 172Bug Debugger Command Set



The 172Bug debugger commands are summarized in Table 4-3. The command syntax is shown using the symbols explained earlier in this chapter. The **CNFG** and **ENV** commands are explained in Appendix A. Controllers, devices, and their LUNs are listed in Appendix D or Appendix E. All other command details are explained in the MVME*172Bug Debugging Package User's Manual*.

**Table 4-3. Debugger Commands**

| Command Mnemonic | Title | Command Line Syntax |
|---|---|---|
| AB | Automatic Bootstrap Operating System | **AB** [**;V**] |
| NOAB | No Autoboot | **NOAB** |
| AS | One Line Assembler | **AS** *addr* |
| BC | Block of Memory Compare | **BC** *range addr* [**; B**\|**W**\|**L**] |
| BF | Block of Memory Fill | **BF** *range data* [*increment*] [**; B**\|**W**\|**L**] |
| BH | Bootstrap Operating System and Halt | **BH** [*controller LUN*] [*device LUN*] [*string*] |
| BI | Block of Memory Initialize | **BI** *range* [**; B**\|**W**\|**L**] |
| BM | Block of Memory Move | **BM** *range addr* [**; B**\|**W**\|**L**] |
| BO | Bootstrap Operating System | **BO** [*controller LUN*] [*device LUN*] [*string*] |
| BR | Breakpoint Insert | **BR** [*addr* [:*count*]] |
| NOBR | Breakpoint Delete | **NOBR** [*addr*] |
| BS | Block of Memory Search | **BS** *range text* [**; B**\|**W**\|**L**] or **BS** *range data* [*mask*] [**; B**\|**W**\|**L** [**,N**] [**,V**]] |
| BV | Block of Memory Verify | **BV** *range data* [*increment*] [**; B**\|**W**\|**L**] |
| CM | Concurrent Mode | **CM** [[*port*] [*ID-string] [baud*] [*phone-number*]]\|[**; A**]\|[**; H**] |
| NOCM | No Concurrent Mode | **NOCM** |
| CNFG | Configure Board Information Block | **CNFG** [**;** [**I**][**M**]] |
| CS | Checksum | **CS** *range* [**; B**\|**W**\|**L**] |
| DC | Data Conversion | **DC** *exp* \| *addr* [**;** [**B**][**O**][**A**]] |

**Table 4-3. Debugger Commands (Continued)**

| Command Mnemonic | Title | Command Line Syntax |
|---|---|---|
| DMA | DMA Block of Memory Move | **DMA** *range addr vdir am blk* [**; B**\|**W**\|**L**] |
| DS | One Line Disassembler | **DS** *addr* [**:***count* \| *addr*] |
| DU | Dump S-records | **DU** [*port*] *range* [*text*] [*addr*] [*offset*] [**; B**\|**W**\|**L**] |
| ECHO | Echo String | **ECHO** [*port*] {*hexadecimal number*} {**'***string***'**} |
| ENV | Set Environment to Bug/Operating System | **ENV** [**;** [**D**]] |
| GD | Go Direct (Ignore Breakpoints) | **GD** [*addr*] |
| GN | Go to Next Instruction | **GN** |
| GO | Go Execute User Program | **GO** [*addr*] |
| GT | Go to Temporary Breakpoint | **GT** *addr* |
| HE | Help | **HE** [*command*] |
| IOC | I/O Control for Disk | **IOC** |
| IOI | I/O Inquiry | **IOI** [**;** [**C**\|**L**]] |
| IOP | I/O Physical (Direct Disk Access) | **IOP** |
| IOT | I/O "TEACH" for Configuring Disk Controller | **IOT** [**;** [**A**\|**F**\|**H**\|**T**]] |
| IRQM | Interrupt Request Mask | **IRQM** [*mask*] |
| LO | Load S-records from Host | **LO** [*port*] [*addr*] [**; [X] [C] [T]**] [**=***text*] |
| MA | Macro Define/Display | **MA** [*name*\|**; L**] |
| NOMA | Macro Delete | **NOMA** [*name*] |
| MAE | Macro Edit | **MAE** *name line#* [*string*] |
| MAL | Enable Macro Expansion Listing | **MAL** |
| NOMAL | Disable Macro Expansion Listing | **NOMAL** |
| MAW | Save Macros | **MAW** [*controller LUN*] [*device LUN*] [*del block #*] |
| MAR | Load Macros | **MAR** [*controller LUN*] [*device LUN*] [*del block #*] |

**Table 4-3. Debugger Commands (Continued)**

| Command Mnemonic | Title | Command Line Syntax |
|---|---|---|
| MD | Memory Display | **MD** [**S**] *addr* [**:***count* \| *addr*] [**;** [**B**\|**W**\|**L**\|**S**\|**D**\|**DI**]] |
| MENU | Menu | **MENU** |
| MM | Memory Modify | **MM** *addr* [**;** [[**B**\|**W**\|**L**\|**S**\|**D**] [**A**] [**N**]]\|[**DI**]] |
| MMD | Memory Map Diagnostic | **MMD** *range increment* [**; B**\|**W**\|**L**] |
| MS | Memory Set | **MS** *addr* {*hexadecimal number*} {**'***string***'**} |
| MW | Memory Write | **MW** *addr data* [**; B**\|**W**\|**L**] |
| NAB | Automatic Network Boot Operating System | **NAB** |
| NBH | Network Boot Operating System and Halt | **NBH** [*controller LUN*] [*device LUN*] [*client IP Address*] [*server IP Address*] [*string*] |
| NBO | Network Boot Operating System | **NBO** [*controller LUN*] [*device LUN*] [*client IP Address*] [*server IP Address*] [*string*] |
| NIOC | Network I/O Control | **NIOC** |
| NIOP | Network I/O Physical | **NIOP** |
| NIOT | Network I/O Teach | **NIOT** [**;** [**H**]\|[**A**]] |
| NPING | Network Ping | **NPING** *controller-LUN device-LUN source-IP destination-IP* [*n-packets*] |
| OF | Offset Registers Display/Modify | **OF** [**R***n* [**; A**]] |
| PA | Printer Attach | **PA** [*port*] |
| NOPA | Printer Detach | **NOPA** [*port*] |
| PF | Port Format | **PF** [*port*] |
| NOPF | Port Detach | **NOPF** [*port*] |
| PFLASH | Program FLASH Memory | **PFLASH** *SSADDR SEADDR DSADDR* [*IEADDR*] [**;**[**A**\|**R**][**X**]] or **PFLASH** *SSADDR***:***COUNT DSADDR* [*IEADDR*] [**;**[**B**\|**W**\|**L**] [**A**\|**R**] [**X**]] |
| PS | Put RTC Into Power Save Mode for Storage | **PS** |
| RB | ROMboot Enable | **RB** [**; V**] |
| NORB | ROMboot Disable | **NORB** |

**Table 4-3. Debugger Commands (Continued)**

| Command Mnemonic | Title | Command Line Syntax |
|---|---|---|
| RD | Register Display | **RD** {[+\|-\|=] [*dname*] [/]} {[+\|-\|=] [*reg1*[-*reg2*]] [/]} [**; E**] |
| REMOTE | Connect the Remote Modem to CSO | **REMOTE** |
| RESET | Cold/Warm Reset | **RESET** |
| RL | Read Loop | **RL** *addr***;** [**B**\|**W**\|**L**] |
| RM | Register Modify | **RM** [*reg*] |
| RS | Register Set | **RS** *reg* [*exp*\|*addr*] |
| SD | Switch Directories | **SD** |
| SET | Set Time and Date | **SET** *mmddyyhhmm* \| *n***; C** |
| SYM | Symbol Table Attach | **SYM** [*addr*] |
| NOSYM | Symbol Table Detach | **NOSYM** |
| SYMS | Symbol Table Display/Search | **SYMS** [*symbol-name*]\|[**; S**] |
| T | Trace | **T** [*count*] |
| TA | Terminal Attach | **TA** [*port*] |
| TC | Trace on Change of Control Flow | **TC** [*count*] |
| TIME | Display Time and Date | **TIME** [**;** [**C**\|**L**\|**O**]] |
| TM | Transparent Mode | **TM** [*port*] [ESCAPE] |
| TT | Trace to Temporary Breakpoint | **TT** *addr* |
| VE | Verify S-Records Against Memory | **VE** [*port*] [*addr*] [**;** [**X**][**C**]] [=*text*] |
| VER | Display Revision/Version | **VER** [**; E**] |
| WL | Write Loop | **WL** *addr data* [**; B**\|**W**\|**L**] |

# Configure Board Information Block

**CNFG** [;[**I**][**M**]]

This command is used to display and configure the board information block. This block is resident within the Non-Volatile RAM (NVRAM). Refer to the *Debugging Package for Motorola 68K CISC CPUs User's Manual* for the actual location. The information block contains various elements detailing specific operation parameters of the hardware. The *Debugging Package for Motorola 68K CISC CPUs User's Manual* describes the elements within the board information block, and lists the size and logical offset of each element. The **CNFG** command does *not* describe the elements and their use. The board information block contents are checksummed for validation purposes. This checksum is the last element of the block.

Although the factory fills all fields except the IndustryPack fields, only these fields MUST contain correct information:

- MPU clock speed
- Ethernet address
- Local SCSI identifier

Example: to display the current contents of the board information block.

```
172-Bug>cnfg
Board (PWA) Serial Number = "000000061050"
Board Identifier          = "MVME172-513A    "
Artwork (PWA) Identifier  = "01-W3960B01A    "
MPU Clock Speed           = "3200"
Ethernet Address          = 08003E20A867
Local SCSI Identifier     = "07"
Parity Memory Mezzanine Artwork (PWA) Identifier  = "       "
Parity Memory Mezzanine (PWA) Serial Number      = "       "
Static Memory Mezzanine Artwork (PWA) Identifier  = "       "
```

```
Static Memory Mezzanine (PWA) Serial Number      = "       "
ECC Memory Mezzanine #1 Artwork (PWA) Identifier  = "       "
ECC Memory Mezzanine #1 (PWA) Serial Number      = "       "
ECC Memory Mezzanine #2 Artwork (PWA) Identifier  = "       "
ECC Memory Mezzanine #2 (PWA) Serial Number      = "       "
Serial Port 2 Personality Artwork (PWA) Identifier  = "     "
Serial Port 2 Personality Module (PWA) Serial Number = "     "
IndustryPack A Board Identifier         = "        "
IndustryPack A (PWA) Serial Number      = "        "
IndustryPack A Artwork (PWA) Identifier = "        "
IndustryPack B Board Identifier         = "        "
IndustryPack B (PWA) Serial Number      = "        "
IndustryPack B Artwork (PWA) Identifier = "        "
IndustryPack C Board Identifier         = "        "
IndustryPack C (PWA) Serial Number      = "        "
IndustryPack C Artwork (PWA) Identifier = "        "
IndustryPack D Board Identifier         = "        "
IndustryPack D (PWA) Serial Number      = "        "
IndustryPack D Artwork (PWA) Identifier = "        "
172-Bug>
```

Note that the parameters that are quoted are left-justified character (ASCII) strings padded with space characters, and the quotes (") are displayed to indicate the size of the string. Parameters that are not quoted are considered data strings, and data strings are right-justified. The data strings are padded with zeroes if the length is not met.

In the event of corruption of the board information block, the command displays a question mark "?" for nondisplayable characters. A warning message (`WARNING: Board Information Block Checksum Error`) is also displayed in the event of a checksum failure.

Using the **I** option initializes the unused area of the board information block to zero.

Modification is permitted by using the **M** option of the command. At the end of the modification session, you are prompted for the update to Non-Volatile RAM (NVRAM). A `Y` response must be made for the update to occur; any other response terminates the update (disregards all changes). The update also recalculates the checksum.

Be cautious when modifying parameters. Some of these parameters are set up by the factory, and correct board operation relies upon these parameters.

Once modification/update is complete, you can now display the current contents as described earlier.

# Set Environment to Bug/Operating System

**ENV** [**;**[**D**]]

The **ENV** command allows you to interactively view/configure all Bug operational parameters that are kept in Battery Backed Up RAM (BBRAM), also known as Non-Volatile RAM (NVRAM). The operational parameters are saved in NVRAM and used whenever power is lost.

Any time the Bug uses a parameter from NVRAM, the NVRAM contents are first tested by checksum to insure the integrity of the NVRAM contents. In the instance of BBRAM checksum failure, certain default values are assumed as stated below.

The bug operational parameters (which are kept in NVRAM) are not initialized automatically on power up/warm reset. It is up to the Bug user to invoke the **ENV** command. Once the **ENV** command is invoked and executed without error, Bug default and/or user parameters are loaded into NVRAM along with checksum data.

If any of the operational parameters have been modified, the new parameters do not go into effect until a reset/powerup condition occurs. Should you determine that the NVRAM contents have been corrupted, use a double-button reset (described under *Restarting the System* in Chapter 3) to reinitialize the system.

If the **ENV** command is invoked with no options on the command line, you are prompted to configure all operational parameters. If the **ENV** command is invoked with the option **D**, ROM defaults will be loaded into NVRAM.

The parameters to be configured are listed in the following table:

## Table A-1. ENV Command Parameters

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Bug or System environment [B/S] | B | Bug mode |
| Field Service Menu Enable [Y/N] | N | Do not display field service menu. |
| Remote Start Method Switch [G/M/B/N] | B | Use both methods (Global Control and Status Register (GCSR) in the VMEchip2, and the Multiprocessor Control Register (MPCR) in shared RAM) to pass and start execution of cross-loaded programs. |
| Probe System for Supported I/O Controllers [Y/N] | Y | Accesses will be made to the appropriate system busses (e.g., VMEbus, local bus) to determine presence of supported controllers. |
| Negate VMEbus SYSFAIL* Always [Y/N] | N | Negate VMEbus SYSFAIL after successful completion or entrance into the bug command monitor. |
| Local SCSI Bus Reset on Debugger Startup [Y/N] | N | Local SCSI bus is not reset on debugger startup. |
| Local SCSI Bus Negotiations Type [A/S/N] | A | Asynchronous |
| Industry Pack Reset on Debugger Startup [Y/N] | Y | Industry Pack(s) is/are reset on debugger startup. |
| Ignore CFGA Block on a Hard Disk Boot [Y/N] | Y | Enable the ignorance of the Configuration Area (CFGA) Block (hard disk only). |
| Auto Boot Enable [Y/N] | N | Auto Boot function is disabled. |
| Auto Boot at power-up only [Y/N] | Y | Auto Boot is attempted at power-up reset only. |
| Auto Boot Controller LUN | 00 | LUN of a disk/tape controller module currently supported by the Bug. Default is $0. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Auto Boot Device LUN | 00 | LUN of a disk/tape device currently supported by the Bug. Default is $0. |
| Auto Boot Abort Delay | 15 | The time in seconds that the Auto Boot sequence will delay before starting the boot. The purpose for the delay is to allow you the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds. |
| Auto Boot Default String [Y(NULL String)/(String)] | | You may specify a string (filename) which is passed on to the code being booted. Maximum length is 16 characters. Default is the null string. |
| ROM Boot Enable [Y/N] | N | ROMboot function is disabled. |
| ROM Boot at power-up only [Y/N] | Y | ROMboot is attempted at power up only. |
| ROM Boot Enable search of VMEbus [Y/N] | N | VMEbus address space will not be accessed by ROMboot. |
| ROM Boot Abort Delay | 00 | The time in seconds that the ROMboot sequence will delay before starting the boot. The purpose for the delay is to allow you the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds. |
| ROM Boot Direct Starting Address | FF800000 | First location tested when the Bug searches for a ROMboot Module. |

## Table A-1. ENV Command Parameters (Continued)

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| ROM Boot Direct Ending Address | FFDFFFFC | Last location tested when the Bug searches for a ROMboot Module. |
| Network Auto Boot Enable [Y/N] | N | Network Auto Boot function is disabled. |
| Network Auto Boot at power-up only [Y/N] | Y | Network Auto Boot is attempted at power up reset only. |
| Network Auto Boot Controller LUN | 00 | LUN of a disk/tape controller module currently supported by the Bug. Default is $0. |
| Network Auto Boot Device LUN | 00 | LUN of a disk/tape device currently supported by the Bug. Default is $0. |
| Network Auto Boot Abort Delay | 5 | This is the time in seconds that the Network Boot sequence will delay before starting the boot. The purpose for the delay is to allow you the option of stopping the boot by use of the Break key. The time value is from 0 through 255 seconds. |
| Network Autoboot Configuration Parameters Pointer (NVRAM) | 00000000 | The address where the network interface configuration parameters are to be saved/retained in NVRAM; these parameters are the necessary parameters to perform an unattended network boot.<br>If you are using NVRAM space for your own program information or commands, change the default pointer value to the value necessary to clear your data. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Memory Search Starting Address | 00000000 | Where the Bug begins to search for a work page (a 64KB block of memory) to use for vector table, stack, and variables. This must be a multiple of the debugger work page, modulo $10000 (64KB). In a multi-172 environment, each MVME172 board could be set to start its work page at a unique address to allow multiple debuggers to operate simultaneously. |
| Memory Search Ending Address | 00100000 | Top limit of the Bug's search for a work page. If a contiguous block of memory, 64KB in size, is not found in the range specified by Memory Search Starting Address and Memory Search Ending Address parameters, then the bug will place its work page in the onboard static RAM on the MVME172. Default Memory Search Ending Address is the calculated size of local memory. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Memory Search Increment Size | 00010000 | A multi-CPU feature used to offset the location of the Bug work page. This must be a multiple of the debugger work page, modulo $10000 (64KB). Typically, Memory Search Increment Size is the product of CPU number and size of the Bug work page. Example: first CPU $0 (0 x $10000), second CPU $10000 (1 x $10000), etc. |
| Memory Search Delay Enable [Y/N] | N | No delay before the Bug begins its search for a work page. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Memory Search Delay Address | FFFFD20F | Default address is $FFFFD20F. This is the MVME172 GCSR GPCSR0 as accessed through VMEbus A16 space and assumes the MVME172 GRPAD (group address) and BDAD (board address within group) switches are set to "on". This byte-wide value is initialized to $FF by MVME172 hardware after a System or Power-on Reset. In a multi-172 environment, where the work pages of several Bugs are to reside in the memory of the primary (first) MVME172, the non-primary CPUs will wait for the data at the Memory Search Delay Address to be set to $00, $01, or $02 (refer to the *Memory Requirements* section in *Chapter 3* for the definition of these values) before attempting to locate their work page in the memory of the primary CPU. |
| Memory Size Enable [Y/N] | Y | Memory will be sized for Self Test diagnostics. |
| Memory Size Starting Address | 00000000 | Default Starting Address is $0. |
| Memory Size Ending Address | 00100000 | Default Ending Address is the calculated size of local memory. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Base Address of Dynamic Memory | 00000000 | Beginning address of Dynamic Memory (Parity and/or ECC type memory). It must be a multiple of the Dynamic Memory board size, starting with 0. Default is $0. |
| Size of Parity Memory | 00100000 | Size of the Parity type dynamic RAM mezzanine, if any. The default is the calculated size of the Dynamic memory mezzanine board. |
| Size of ECC Memory Board #0 | 00000000 | Size of the first ECC type memory mezzanine. The default is the calculated size of the memory mezzanine. |
| Size of ECC Memory Board #1 | 00000000 | Size of the second ECC type memory mezzanine. The default is the calculated size of the memory mezzanine. |
| Base Address of Static Memory | FFE00000 | The beginning address of SRAM. The default for this parameter is FFE00000 for the onboard 512KB. |
| Size of Static Memory | 00080000 | Size of the SRAM type memory present. The default is the calculated size of the onboard SRAM. |
| **ENV** asks the following series of questions to set up the VMEbus interface for the MVME172 series modules. You should have a working knowledge of the VMEchip2 as given in the *MVME172 VME Embedded Controller Programmer's Reference Guide* in order to perform this configuration. Also included in this series are questions for setting ROM and Flash access time.<br>The slave address decoders are used to allow another VMEbus master to access a local resource of the MVME172. There are two slave address decoders set. They are set up as follows: | | |
| Slave Enable #1 [Y/N] | Y | Yes, set up and enable the Slave Address Decoder #1. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Slave Starting Address #1 | 00000000 | Base address of the local resource that is accessible by the VMEbus. Default is the base of local memory, $0. |
| Slave Ending Address #1 | 000FFFFF | Ending address of the local resource that is accessible by the VMEbus. Default is the end of calculated memory. |
| Slave Address Translation Address #1 | 00000000 | Register that allows the VMEbus address and the local address to be different. The value in this register is the base address of local resource that is associated with the starting and ending address selection from the previous questions. Default is 0. |
| Slave Address Translation Select #1 | 00000000 | Register that defines which bits of the address are significant. A logical one "1" indicates significant address bits, logical zero "0" is non-significant. Default is 0. |
| Slave Control #1 | 03FF | Defines the access restriction for the address space defined with this slave address decoder. Default is $03FF. |
| Slave Enable #2 [Y/N] | N | Do not set up and enable the Slave Address Decoder #2. |
| Slave Starting Address #2 | 00000000 | Base address of the local resource that is accessible by the VMEbus. Default is 0. |
| Slave Ending Address #2 | 00000000 | Ending address of the local resource that is accessible by the VMEbus. Default is 0. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Slave Address Translation Address #2 | 00000000 | Works the same as Slave Address Translation Address #1. Default is 0. |
| Slave Address Translation Select #2 | 00000000 | Works the same as Slave Address Translation Select #1. Default is 0. |
| Slave Control #2 | 0000 | Defines the access restriction for the address space defined with this slave address decoder. Default is $0000. |
| Master Enable #1 [Y/N] | Y | Yes, set up and enable the Master Address Decoder #1. |
| Master Starting Address #1 | 02000000 | Base address of the VMEbus resource that is accessible from the local bus. Default is the end of calculated local memory, unless memory is less than 16MB, then this register will always be set to 01000000. |
| Master Ending Address #1 | EFFFFFFF | Ending address of the VMEbus resource that is accessible from the local bus. Default is the end of calculated memory. |
| Master Control #1 | 0D | Defines the access characteristics for the address space defined with this master address decoder. Default is $0D. |
| Master Enable #2 [Y/N] | N | Do not set up and enable the Master Address Decoder #2. |
| Master Starting Address #2 | 00000000 | Base address of the VMEbus resource that is accessible from the local bus. Default is $00000000. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Master Ending Address #2 | 00000000 | Ending address of the VMEbus resource that is accessible from the local bus. Default is $00000000. |
| Master Control #2 | 00 | Defines the access characteristics for the address space defined with this master address decoder. Default is $00. |
| Master Enable #3 [Y/N] | Depends on calculated size of local RAM | Yes, set up and enable the Master Address Decoder #3. This is the default if the board contains less than 16MB of calculated RAM.<br>Do not set up and enable the Master Address Decoder #3. This is the default for boards containing at least 16MB of calculated RAM. |
| Master Starting Address #3 | 00000000 | Base address of the VMEbus resource that is accessible from the local bus. If enabled, the value is calculated as one more than the calculated size of memory. If not enabled, the default is $00000000. |
| Master Ending Address #3 | 00000000 | Ending address of the VMEbus resource that is accessible from the local bus. If enabled, the default is $00FFFFFF, otherwise $00000000. |

## Table A-1. ENV Command Parameters (Continued)

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Master Control #3 | 00 | Defines the access characteristics for the address space defined with this master address decoder. If enabled, the default is $3D, otherwise $00. |
| Master Enable #4 [Y/N] | N | Do not set up and enable the Master Address Decoder #4. |
| Master Starting Address #4 | 00000000 | Base address of the VMEbus resource that is accessible from the local bus. Default is $0. |
| Master Ending Address #4 | 00000000 | Ending address of the VMEbus resource that is accessible from the local bus. Default is $0. |
| Master Address Translation Address #4 | 00000000 | Register that allows the VMEbus address and the local address to be different. The value in this register is the base address of VMEbus resource that is associated with the starting and ending address selection from the previous questions. Default is 0. |
| Master Address Translation Select #4 | 00000000 | Register that defines which bits of the address are significant. A logical one "1" indicates significant address bits, logical zero "0" is non-significant. Default is 0. |
| Master Control #4 | 00 | Defines the access characteristics for the address space defined with this master address decoder. Default is $00. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| Short I/O (VMEbus A16) Enable [Y/N] | Y | Yes, Enable the Short I/O Address Decoder. |
| Short I/O (VMEbus A16) Control | 01 | Defines the access characteristics for the address space defined with the Short I/O address decoder. Default is $01. |
| F-Page (VMEbus A24) Enable [Y/N] | Y | Yes, Enable the F-Page Address Decoder. |
| F-Page (VMEbus A24) Control | 02 | Defines the access characteristics for the address space defined with the F-Page address decoder. Default is $02. |
| ROM Access Time Code | 04 (50 MHz MPU), or 03 (64 MHz MPU) | Defines the ROM access time. The default is: $04, which sets an access time of220 ns (25MHz bus frequency), or $03, which sets an access time of 130 ns (32MHz bus frequency). |
| Flash Access Time Code | 01 | Defines the Flash access time. The default is $01, which sets an access time of 100 ns (25MHz bus frequency) or 70 ns (32MHz bus frequency). |
| MC2 chip Vector Base<br>VMEC2 Vector Base #1<br>VMEC2 Vector Base #2 | 05<br>06<br>07 | Base interrupt vector for the component specified. Default: MC2 chip = $05, VMEchip2 Vector 1 = $06, VMEchip2 Vector 2 = $07. |

**Table A-1. ENV Command Parameters (Continued)**

| ENV Parameter and Options | Default | Meaning of Default |
|---|---|---|
| VMEC2 GCSR Group Base Address | D2 | Specifies the group address ($FFF*x*00) in Short I/O for this board. Default = $D2. |
| VMEC2 GCSR Board Base Address | 00 | Specifies the base address ($FFFFD2*xx*) in Short I/O for this board. Default = $00. |
| VMEbus Global Time Out Code | 01 | Controls the VMEbus timeout when systems controller. Default $01 = 64 µs. |
| Local Bus Time Out Code | 02 | This controls the local bus timeout. Default $02 = 256 µs. |
| VMEbus Access Time Out Code | 02 | This controls the local bus to VMEbus access timeout. Default $02 = 32 ms. |

## Configuring the IndustryPacks

**ENV** asks the following series of questions to set up IndustryPacks (IP) on MVME172 modules.

The *MVME172 VME Embedded Controller Programmer's Reference Guide* describes the base addresses and the IP register settings. Refer to that manual for information on setting base addresses and register bits.

IP A Base Address = 00000000?
IP B Base Address = 00000000?
IP C Base Address = 00000000?
IP D Base Address = 00000000?

Base address for mapping IP modules. Only the upper 16 bits are significant.

IP D/C/B/A Memory Size = 00000000?

Define the memory size requirements for the IP modules:

| Bits | IP | Register Address |
|---|---|---|
| 31-24 | D | FFFBC00F |
| 23-16 | C | FFFBC00E |
| 15-08 | B | FFFBC00D |
| 07-00 | A | FFFBC00C |

IP D/C/B/A General Control = 00000000?

Define the general control requirements for the IP modules:

| Bits | IP | Register Address |
|---|---|---|
| 31-24 | D | FFFBC01B |
| 23-16 | C | FFFBC01A |
| 15-08 | B | FFFBC019 |
| 07-00 | A | FFFBC018 |

IP D/C/B/A Interrupt 0 Control = 00000000?

Define the interrupt control requirements for the IP modules channel 0:

| Bits | IP | Register Address |
|---|---|---|
| 31-24 | D | FFFBC016 |
| 23-16 | C | FFFBC014 |
| 15-08 | B | FFFBC012 |
| 07-00 | A | FFFBC010 |

IP D/C/B/A Interrupt 1 Control = 00000000?

Define the interrupt control requirements for the IP modules channel 1:

| Bits | IP | Register Address |
|---|---|---|
| 31-24 | D | FFFBC017 |
| 23-16 | C | FFFBC015 |
| 15-08 | B | FFFBC013 |
| 07-00 | A | FFFBC011 |



**Caution**

**Before environment parameters are saved in the NVRAM, a warning message will appear if the user has specified environment parameters which will cause an overlap condition. The important information about each configurable element in the memory map is displayed, showing where any overlap conditions exist. This will allow the user to quickly identify and correct an undesirable configuration before it is saved.**

ENV warning example:

```
WARNING: Memory MAP Overlap Condition Exists

S-Address   E-Address   Enable Overlap  M-Type   Memory-MAP-Name
$00000000   $FFFFFFFF   Yes    Yes      Master   Local Memory (Dynamic RAM)
$FFE00000   $FFE7FFFF   Yes    Yes      Master   Static RAM
$01000000   $EFFFFFFF   Yes    Yes      Master   VMEbus Master #1
$00000000   $00000000   No     No       Master   VMEbus Master #2
$00000000   $00FFFFFF   Yes    Yes      Master   VMEbus Master #3
$00000000   $00000000   No     No       Master   VMEbus Master #4
$F0000000   $FF7FFFFF   Yes    Yes      Master   VMEbus F Pages (A24/A32)
$FFFF0000   $FFFFFFFF   Yes    Yes      Master   VMEbus Short I/O (A16)
$FF800000   $FFBFFFFF   Yes    Yes      Master   Flash/PROM
$FFF00000   $FFFEFFFF   Yes    Yes      Master   Local I/O
$00000000   $00000000   No     No       Master   Industry Pack A
$00000000   $00000000   No     No       Master   Industry Pack B
$00000000   $00000000   No     No       Master   Industry Pack C
$00000000   $00000000   No     No       Master   Industry Pack D
$00000000   $00000000   No     No       Slave    VMEbus Slave #1
$00000000   $00000000   No     No       Slave    VMEbus Slave #2
```

# Serial Interconnections

B

## Introduction

As described in previous chapters of this manual, one of the MVME172's two serial ports (port A internally, SERIAL PORT 1/CONSOLE on the front panel) is an EIA-232-D DCE port exclusively. The second port (port B internally, SERIAL PORT 2 on the front panel) can be configured via serial interface modules as an EIA-232-D DCE/DTE, an EIA-530 DCE/DTE port, an EIA-485 port, or an EIA-422 DCE/DTE port.

The MVME172 uses a Zilog Z85230 serial port controller to implement the two serial communications interfaces. Each interface supports CTS, DCD, RTS, and DTR control signals as well as the TxD and RxD transmit/receive data signals, and TxC/RxC synchronous clock signals. The Z85230 supports synchronous (SDLC/HDLC) and asynchronous protocols. The MVME172 hardware supports asynchronous serial baud rates of 110B/sec to 38.4KB/sec.

## EIA-232-D Connections

The EIA-232-D standard defines the electrical and mechanical aspects of this serial interface. The interface employs unbalanced (single-ended) signaling and is generally used with DB-25 connectors, although other connector styles (e.g., DB-9 and RJ-45) are sometimes used as well.

Table A-1 lists the standard EIA-232-D interconnections. Not all pins listed in the table are necessary in every application.

To interpret the information correctly, remember that the EIA-232-D serial interface was developed to connect a terminal to a modem. Serial data leaves the sending device on a Transmit Data (TxD) line and arrives at the receiving device on a Receive Data (RxD) line. When computing equipment is interconnected without modems, one of the units must be configured as a terminal (data terminal equipment: DTE) and the other as

a modem (data circuit-terminating equipment: DCE). Since computers are normally configured to work with terminals, they are said to be configured as a modem in most cases.

## Table B-1. EIA-232-D Interconnections

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 1 | | Not used. |
| 2 | TxD | **Transmit Data**. Data to be transmitted; input to modem from terminal. |
| 3 | RxD | **Receive Data**. Data which is demodulated from the receive line; output from modem to terminal. |
| 4 | RTS | **Request To Send**. Input to modem from terminal when required to transmit a message. With RTS off, the modem carrier remains off. When RTS is turned on, the modem immediately turns on the carrier. |
| 5 | CTS | **Clear To Send**. Output from modem to terminal to indicate that message transmission can begin. When a modem is used, CTS follows the off-to-on transition of RTS after a time delay. |
| 6 | DSR | **Data Set Ready**. Output from modem to terminal to indicate that the modem is ready to send or receive data. |
| 7 | SG | **Signal Ground**. Common return line for all signals at the modem interface. |
| 8 | DCD | **Data Carrier Detect**. Output from modem to terminal to indicate that a valid carrier is being received. |
| 9-14 | | Not used. |
| 15 | TxC | **Transmit Clock** (DCE). Output from modem to terminal; clocks data from the terminal to the modem. |
| 16 | | Not used. |
| 17 | RxC | **Receive Clock**. Output from terminal to modem; clocks input data from the terminal to the modem. |
| 18, 19 | | Not used. |
| 20 | DTR | **Data Terminal Ready**. Input to modem from terminal; indicates that the terminal is ready to send or receive data. |
| 21 | | Not used. |

## Table B-1. EIA-232-D Interconnections (Continued)

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 22 | RI | **Ring Indicator**. Output from modem to terminal; indicates that an incoming call is present. The terminal causes the modem to answer the phone by carrying DTR true while RI is active. |
| 23 | | Not used. |
| 24 | TxC | **Transmit Clock** (DTE). Input to modem from terminal; same function as TxC on pin 15. |
| 25 | BSY | **Busy**. Input to modem from terminal; a positive EIA signal applied to this pin causes the modem to go off-hook and make the associated phone busy. |

**Notes**

1. A high EIA-232-D signal level is +3 to +15 volts. A low level is –3 to –15 volts. Connecting units in parallel may produce out-of-range voltages and is contrary to specifications.
2. The EIA-232-D interface is intended to connect a terminal to a modem. When computers are connected without modems, one computer must be configured as a modem and the other as a terminal.

B

## Interface Characteristics

The EIA-232-D interface standard specifies all parameters for serial binary data interchange between DTE and DCE devices using unbalanced lines. EIA-232-D transmitter and receiver parameters applicable to the MVME172 are listed in Tables A-2 and A-3.

### Table B-2. EIA-232-D Interface Transmitter Characteristics

| Parameter | Value | | Unit |
|---|---|---|---|
| | Minimum | Maximum | |
| Output voltage (with load resistance of 3000Ω to 7000Ω) | ±8.5 | | V |
| Open circuit output voltage | | ±12 | V |
| Short circuit output current (to ground or any other interconnection cable conductor) | | ±100 | mA |
| Power off output resistance | 300 | | W |
| Output transition time (for a transition region of –3V to +3V and with total load capacitance, including connection cable, of less than 2500pF) | | 2 | µs |
| Open circuit slew rate | | 30 | V/µs |

### Table B-3. EIA-232-D Interface Receiver Characteristics

| Parameter | Value | | Unit |
|---|---|---|---|
| | Minimum | Maximum | |
| Input signal voltage | | ±25 | V |
| Input high threshold voltage | | 2.25 | V |
| Input low threshold voltage | 0.75 | | V |
| Input hysteresis | 1.0 | | V |
| Input impedance ($-15V < V_{in} < +15V$) | 3000 | 7000 | W |

The MVME172 conforms to EIA-232-D specifications. Note that although the EIA-232-D standard recommends the use of short interconnection cables not more than 50 feet (15m) in length, longer cables are permissible provided the total load capacitance measured at the interface point and including signal terminator does not exceed 2500pF.

# EIA-530 Connections

The EIA-530 interface complements the EIA-232-D interface in function. The EIA-530 standard defines the mechanical aspects of this interface, which is used for transmission of serial binary data, both synchronous and asynchronous. It is adaptable to balanced (double-ended) as well as unbalanced (single-ended) signaling and offers the possibility of higher data rates than EIA-232-D with the same DB-25 connector.

Table A-4 lists the EIA-530 interconnections that are available at serial port B (SERIAL PORT 2 on the front panel) when the port is configured via serial interface modules as an EIA-530 DCE or DTE port.

**Table B-4. Serial Port B EIA-530 Interconnect Signals**

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 1 | | Not used. |
| 2 | TxD_A | **Transmit Data (A).** Data to be transmitted; output from DTE to DCE. |
| 3 | RxD_A | **Receive Data (A).** Data which is demodulated from the receive line; input from DCE to DTE. |
| 4 | RTS_A | **Request to Send (A).** Output from DTE to DCE when required to transmit a message. |
| 5 | CTS_A | **Clear to Send (A).** Input to DTE from DCE to indicate that message transmission can begin. |
| 6 | DSR_A | **Data Set Ready (A).** Input to DTE from DCE to indicate that the DCE is ready to send or receive data. In DCE configuration, always true. |
| 7 | SIG GND | **Signal Ground.** Common return line for all signals. |
| 8 | DCD_A | **Data Carrier Detect (A)**. Receive line signal detector output from DCE to DTE to indicate that valid data is being transferred to the DTE on the RxD line. |
| 9 | RxC_B | **Receive Signal Element Timing—DCE (B).** Control signal that clocks input data. |

## Table B-4. Serial Port B EIA-530 Interconnect Signals (Continued)

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 10 | DCD_B | **Data Carrier Detect (B).** Receive line signal detector output from DCE to DTE to indicate that valid data is being transferred to the DTE on the RxD line. |
| 11 | TxCO_B | **Transmit Signal Element Timing—DTE (B).** Control signal that clocks output data. |
| 12 | TxC_B | **Transmit Signal Element Timing—DCE (B).** Control signal that clocks input data. |
| 13 | CTS_B | **Clear to Send (B).** Input to DTE from DCE to indicate that message transmission can begin. |
| 14 | TxD_B | **Transmit Data (B).** Data to be transmitted; output from DTE to DCE. |
| 15 | TxC_A | **Transmit Signal Element Timing—DCE (A).** Control signal that clocks input data. |
| 16 | RxD_B | **Receive Data (B).** Data which is demodulated from the receive line; input from DCE to DTE. |
| 17 | RxC_A | **Receive Signal Element Timing—DCE (A).** Control signal that clocks input data. |
| 18 | LL_A | **Local Loopback (A).** Reroutes signal within local DCE. In DTE configuration, always tied inactive and driven false. In DCE configuration, ignored. |
| 19 | RTS_B | **Request to Send (B).** Output from DTE to DCE when required to transmit a message. |
| 20 | DTR_A | **Data Terminal Ready (A).** Output from DTE to DCE indicating that the DTE is ready to send or receive data. |
| 21 | RL_A | **Remote Loopback (A).** Reroutes signal within remote DCE. In DTE configuration, always tied inactive and driven false. In DCE configuration, ignored. |
| 22 | DSR_B | **Data Set Ready (B).** Input to DTE from DCE to indicate that the DCE is ready to send or receive data. In DCE configuration, always true. |
| 23 | DTR_B | **Data Terminal Ready (B).** Output from DTE to DCE indicating that the DTE is ready to send or receive data. |
| 24 | TxCO_A | **Transmit Signal Element Timing—DTE (A).** Control signal that clocks output data. |
| 25 | TM_A | **Test Mode (A).** Indicates whether the local DCE is under test. In DTE configuration, ignored. In DCE configuration, always tied inactive and driven false. |

## Interface Characteristics

In specifying parameters for serial binary data interchange between DTE and DCE devices, the EIA-530 standard assumes the use of balanced lines, except for the Remote Loopback, Local Loopback, and Test Mode lines, which are single-ended. Balanced-line data interchange is generally employed in preference to unbalanced-line data interchange where any of the following conditions prevail:

- The interconnection cable is too long for effective unbalanced operation.
- The interconnection cable is exposed to extraneous noise sources that may cause an unwanted voltage in excess of ±1V measured differentially between the signal conductor and circuit ground at the load end of the cable, with a 50Ω resistor substituted for the transmitter.
- It is necessary to minimize interference with other signals.
- Inversion of signals may be required (e.g., plus polarity MARK to minus polarity MARK may be achieved by inverting the cable pair).

EIA-530 interface transmitter and receiver parameters applicable to the MVME172 are listed in Tables A-5 and A-6.

**Table B-5. EIA-530 Interface Transmitter Characteristics**

| Parameter | Value | | Unit |
|---|---|---|---|
| | Minimum | Maximum | |
| Differential output voltage (absolute, with 100Ω load) | 2.0 | | V |
| Open circuit differential voltage output (absolute) | | 6.0 | V |
| Output offset voltage (with 100Ω load) | | 3.0 | V |
| Short circuit output current (for any voltage between –7V and +7V) | | ±180 | mA |
| Power off output current (for any voltage between –7V and +7V) | | ±100 | µA |
| Output transition time (with 100Ω, 15pF load) | | 15 | ns |

**Table B-6. EIA-530 Interface Receiver Characteristics**

| Parameter | Value Minimum | Value Maximum | Unit |
|---|---|---|---|
| Differential input voltage | | ±12 | V |
| Input offset voltage | | ±12 | V |
| Differential input high threshold voltage | | 200 | mV |
| Differential input low threshold voltage | | –200 | mV |
| Differential input hysteresis | 50 | | mV |
| Input impedance (without termination resistors) | 10 | | KΩ |

# EIA-485/EIA-422 Connections

The EIA-485 and EIA-422 standards define a balanced (double-ended) electrical interface, which is used for transmission of serial binary data, both synchronous and asynchronous. As used here, they use the same DB-25 connector and pin assignments as does EIA-530. EIA-485 is a low cost differential multidrop driver and receiver standard. The quadruple differential line drivers and receivers that are used in this module are tri-state. They meet the requirements of EIA-485 and EIA-422.

Table A-7 lists the EIA-485/EIA-422 interconnections that are available at serial port B (SERIAL PORT 2 on the front panel) when the port is configured via serial interface modules as an EIA-485 port or as an EIA-422 DCE or DTE port.

## Table B-7. Serial Port B EIA-485/EIA-422 Interconnect Signals

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 1 | | Not used. |
| 2 | TxD_A | **Transmit Data (A).** Data to be transmitted; output from DTE to DCE. |
| 3 | RxD_A | **Receive Data (A).** Data which is demodulated from the receive line; input from DCE to DTE. |
| 4 | | Not used. |
| 5 | | Not used. |
| 6 | | Not used. |
| 7 | SIG GND | **Signal Ground.** Common return line for all signals. |
| 8 | | Not used. |
| 9 | RxC_B | **Receive Signal Element Timing—DCE (B).** Control signal that clocks input data. |
| 10 | | Not used. |
| 11 | | Not used. |
| 12 | TxC_B | **Transmit Signal Element Timing—DCE (B).** Control signal that clocks input data. |
| 13 | | Not used. |
| 14 | TxD_B | **Transmit Data (B).** Data to be transmitted; output from DTE to DCE. |
| 15 | TxC_A | **Transmit Signal Element Timing—DCE (A).** Control signal that clocks input data. |
| 16 | RxD_B | **Receive Data (B).** Data which is demodulated from the receive line; input from DCE to DTE. |
| 17 | RxC_A | **Receive Signal Element Timing—DCE (A).** Control signal that clocks input data. |
| 18 | | Not used. |
| 19 | | Not used. |
| 20 | | Not used. |
| 21 | | Not used. |
| 22 | | Not used. |
| 23 | | Not used. |
| 24 | | Not used. |
| 25 | | Not used. |

B

## Interface Characteristics

In specifying parameters for serial binary data interchange between DTE and DCE devices, the EIA-485/EIA-422 standard assumes the use of balanced lines. Balanced-line data interchange is generally employed in preference to unbalanced-line data interchange where any of the following conditions prevail:

- The interconnection cable is too long for effective unbalanced operation.
- The interconnection cable is exposed to extraneous noise sources that may cause an unwanted voltage in excess of ±1V measured differentially between the signal conductor and circuit common at the load end of the cable, with a 50Ω resistor substituted for the generator.
- It is necessary to minimize interference with other signals.
- Inversion of signals may be required (e.g., plus polarity MARK to minus polarity MARK may be achieved by inverting the cable pair).

EIA-485 interface transmitter (generator) and receiver (load) parameters applicable to the MVME172 are listed in Tables A-8 and A-9.

**Table B-8. EIA-485 Interface Transmitter (Generator) Characteristics**

| Parameter | Value | | Unit |
|---|---|---|---|
| | Minimum | Maximum | |
| Differential output voltage (absolute, with 100Ω ± 1% load) | 2.0 | | V |
| Open circuit differential voltage output (absolute) | | 6.0 | V |
| Output offset voltage (with 100Ω ± 1% load) | | 3.0 | V |
| Short circuit output current | | ±180 | mA |
| Power off output current (for any voltage between –0.25V and +6.0V) | | ±100 | µA |
| Output transition time (with 54Ω ± 1% load) | | 120 | ns |

**Table B-9. EIA-485 Interface Receiver (Load) Characteristics**

| Parameter | Value | | Unit |
|---|---|---|---|
| | Minimum | Maximum | |
| Differential input voltage | | ±12 | V |
| Input offset voltage | | ±12 | V |
| Differential input high threshold voltage | | +200 | mV |
| Differential input low threshold voltage | –200 | | mV |
| Differential input hysteresis | 50 (typical) | | mV |
| Input impedance (without termination resistors) | 12 | | KΩ |

# Proper Grounding

An important subject to consider is the use of ground pins. There are two pins labeled GND. Pin 7 is the *signal ground* and must be connected to the distant device to complete the circuit. Pin 1 is the *chassis ground*, but it must be used with care. The chassis is connected to the power ground through the green wire in the power cord and must be connected to be in compliance with the electrical code.

The problem is that when units are connected to different electrical outlets, there may be several volts of difference in ground potential. If pin 1 of each device is interconnected with the others via cable, several amperes of current could result. This condition may not only be dangerous for the small wires in a typical cable, but may also produce electrical noise that causes errors in data transmission. That is why Tables A-1, A-4, and A-7 show no connection for pin 1. Normally, pin 7 (*signal ground*) should only be connected to the *chassis ground* at one point; if several terminals are used with one computer, the logical place for that point is at the computer. The terminals should not have a connection between the logic ground return and the chassis.

# IndustryPack Interconnections

C

## Introduction

Up to four IndustryPack (IP) modules may be installed on the MVME172. For each IP module, there are two 50-pin plug connectors on the MVME172: J4/J5, J10/J11, J20/J21, and J29/J30. For external cabling to the IP modules, four 50-pin IDC connectors (J7, J8, J26, and J27) are provided behind the MVME172 front panel. Pin assignments are the same for both types of connectors. The following table lists the pin numbers, signal mnemonics, and signal descriptions for the IndustryPack logic interface.

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 1 | GND | **Ground**. First of four ground pins. Serves as zero-volt reference for logic signals, and as return path for the power supplies furnishing operating voltages to the IndustryPack. |
| 2 | CLK | **Clock**. An 8 MHz clock signal supplied to the IndustryPack by the MVME172. Synchronizes all data transfers to or from the IndustryPack. |
| 3 | Reset* | **Reset**. Driven by the MVME172 to the IndustryPack to halt all IP activity and reset the IP circuitry to a known state. |
| 4-19 | D0-D15 | **Data Bus** (bits 0-15). The 16 lines of the data bus used to read and write data between the MVME172 and the IndustryPack. |
| 20, 21 | BS0*, BS1* | **Byte Select**. Byte select lines; used on 16-bit IPs to support byte writes. BS0* selects the low or odd byte (D0-D7). BS1* selects the high or even byte (D8-D15). |
| 22 | –12V | **–12 Vdc Power**. Used primarily to power IndustryPack analog and communication functions. |
| 23 | +12V | **+12 Vdc Power**. Used primarily to power IndustryPack analog and communication functions. |
| 24 | +5V | **+5 Vdc Power**. First of two +5V pins. Primary supply for digital logic functions on the IndustryPack. |
| 25, 26 | GND | **Ground**. Second and third of four ground pins. Serve as zero-volt reference for logic signals, and as return path for the power supplies furnishing operating voltages to the IndustryPack. |

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 27 | +5V | **+5 Vdc Power**. Second of two +5V pins. Primary supply for digital logic functions on the IndustryPack. |
| 28 | R/W* | **Read/Write**. Indicates the direction of data movement on the data bus. High indicates a read cycle (data lines driven by the IP); low indicates a write cycle (data lines driven by the MVME172). |
| 29 | IDSel* | **IndustryPack ID**. First of four ‘‘select’’ lines driven by the MVME172 to enable the IP. This line is used to read a 32-byte ROM containing the IP identification information. IDSel* is not bussed; the signal is unique to each IndustryPack. |
| 30 | DMAReq0 | **DMA Request 0**. One of two DMA request lines; driven by the IndustryPack to indicate that the IP wishes to have a DMA cycle performed on DMA channel 0. |
| 31 | MemSel* | **Memory Select**. Second of four ‘‘select’’ lines driven by the MVME172 to enable the IP. This line is used in memory read or write cycles. MemSel* is not bussed; the signal is unique to each IndustryPack. An IP need not respond to MemSel* if it has no memory. |
| 32 | DMAReq1 | **DMA Request 1**. One of two DMA request lines; driven by the IndustryPack to indicate that the IP wishes to have a DMA cycle performed on DMA channel 1. |
| 33 | IntSel* | **Interrupt Vector Select**. Third of four ‘‘select’’ lines driven by the MVME172 to enable the IP. This line is used in reading the IP’s interrupt vector during an interrupt acknowledge cycle. IntSel* is not bussed; the signal is unique to each IndustryPack. An IP need not respond to IntSel* if it has no interrupt requests asserted. |
| 34 | DMAck* | **DMA Acknowledge**. DMA acknowledge line driven by the MVME172, used to qualify DMA cycles. DMAck* is bussed to the four IP connectors. |
| 35 | IOSel* | **I/O Select**. Fourth of four ‘‘select’’ lines driven by the MVME172 to enable the IP. This line is used in executing input or output cycles. IOSel* is not bussed; the signal is unique to each IndustryPack. I/O data width is an IP-specific function; an IP need not respond to IntSel* if it has no I/O functions. |
| 36 | DMAck1* | **DMA Acknowledge 1**. Not connected |
| 37 | A1 | **Address Line 1**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |
| 38 | DMAEnd* | **DMA Termination**. A bidirectional line which can be used to terminate DMA transfers. It can be asserted either by the DMA controller or by an IndustryPack module. |
| 39 | A2 | **Address Line 2**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |

| Pin Number | Signal Mnemonic | Signal Name and Description |
|---|---|---|
| 40 | Error* | **IP Error**. Asserted by an IndustryPack module in the event of a a non-recoverable error (e.g., component failure). Less serious errors are signaled by interrupts. |
| 41 | A3 | **Address Line 3**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |
| 42 | IntReq0* | **Interrupt Request 0**. One of two interrupt request lines; driven by an IndustryPack to indicate that the IP is requesting service from the MVME172. |
| 43 | A4 | **Address Line 4**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |
| 44 | IntReq1* | **Interrupt Request 1**. One of two interrupt request lines; driven by an IndustryPack to indicate that the IP is requesting service from the MVME172. |
| 45 | A5 | **Address Line 5**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |
| 46 | Strobe* | **Function Strobe**. Available for use as an input to the IP module by a strobe or clock signal related to the bus interface logic. Can be connected or disconnected on the MVME172 via jumper header J19. |
| 47 | A6 | **Address Line 6**. One of six address lines; driven by the MVME172 to address I/O locations on the IndustryPack module designated by the four ‘‘select’’ lines. |
| 48 | Ack* | **Data Acknowledge**. Asserted by an IndustryPack module to terminate each data transfer. |
| 49 | +5STBY | **+5 Vdc Standby**. Second of two +5V pins. Available for standby functions on the IndustryPack, such as non-volatile memory, real-time clocks, etc. |
| 50 | GND | **Ground**. Fourth of four ground pins. Serves as zero-volt reference for logic signals, and as return path for the power supplies furnishing operating voltages to the IndustryPack. |

# Disk/Tape Controller Data

D

## Controller Modules Supported

The following VMEbus disk/tape controller modules are supported by the 172Bug. The default address for each controller type is First Address and the controller can be addressed by First CLUN during commands **BH**, **BO**, or **IOP**, or during TRAP #15 calls .DSKRD or .DSKWR. Note that if another controller of the same type is used, the second one must have its address changed by its onboard jumpers and/or switches, so that it matches Second Address and can be called up by Second CLUN.

| Controller Type | First CLUN | First Address | Second CLUN | Second Address |
|---|---|---|---|---|
| CISC Embedded Controller | $00 (Note 1) | -- | -- | -- |
| MVME320 - Winchester/Floppy Controller | $11 (Note 2) | $FFFFB000 | $12 (NOTE 2) | $FFFFAC00 |
| MVME323 - ESDI Winchester Controller | $08 | $FFFFA000 | $09 | $FFFFA200 |
| MVME327A - SCSI Controller | $02 | $FFFFA600 | $03 | $FFFFA700 |
| MVME328 - SCSI Controller | $06 | $FFFF9000 | $07 | $FFFF9800 |
| MVME328 - SCSI Controller | $16 | $FFFF4800 | $17 | $FFFF5800 |
| MVME328 - SCSI Controller | $18 | $FFFF7000 | $19 | $FFFF7800 |
| MVME350 - Streaming Tape Controller | $04 | $FFFF5000 | $05 | $FFFF5100 |

**Notes**:

1. If an MVME172 with an SCSI port is used, the MVME172 module has CLUN 0.
2. For MVME172s, the first MVME320 has CLUN $11, and the second MVME320 has CLUN $12.

# Default Configurations

**Note** SCSI Common Command Set (CCS) devices are the only ones tested by Motorola Computer Group.



**CISC Embedded Controllers -- 7 Devices**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 0 | $*XXXXXXXX* | 00<br>10<br>20<br>30<br>40<br>50<br>60 | SCSI Common Command Set<br>(CCS), which may be any of these:<br>- Fixed direct access<br>- Removable flexible direct access<br>(TEAC style)<br>- CD-ROM<br>- Sequential access |

**MVME320 -- 4 Devices**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 11 | $FFFFB000 | 0 | Winchester hard drive |
| | | 1 | Winchester hard drive |
| 12 | $FFFFAC00 | 2 | 5-1/4" DS/DD 96 TPI floppy drive |
| | | 3 | 5-1/4" DS/DD 96 TPI floppy drive |

**MVME323 -- 4 Devices**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 8 | $FFFFA000 | 0 | ESDI Winchester hard drive |
| | | 1 | ESDI Winchester hard drive |
| 9 | $FFFFA200 | 2 | ESDI Winchester hard drive |
| | | 3 | ESDI Winchester hard drive |

**MVME327A -- 9 Devices**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 2 | $FFFFA600 | 00<br>10<br>20<br>30<br>40<br>50<br>60 | SCSI Common Command Set<br>(CCS), which may be any of these:<br>- Fixed direct access<br>- Removable flexible direct access<br>(TEAC style)<br>- CD-ROM<br>- Sequential access |
| 3 | $FFFFA700 | 80 | Local floppy drive |
| | | 81 | Local floppy drive |

**MVME328 -- 14 Devices**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 6 | $FFFF9000 | 00<br>08<br>10<br>18<br>20<br>28<br>30 | SCSI Common Command Set<br>(CCS), which may be any of these:<br>- Removable flexible direct access<br>(TEAC style)<br>- CD-ROM<br>- Sequential access |
| 7 | $FFFF9800 | | |
| 16 | $FFFF4800 | 40<br>48<br>50<br>58<br>60<br>68<br>70 | Same as above, but these<br>will only be available if<br>the daughter card for the<br>second SCSI channel is present. |
| 17 | $FFFF5800 | | |
| 18 | $FFFF7000 | | |
| 19 | $FFFF7800 | | |

**MVME350 -- 1 Device**

| Controller LUN | Address | Device LUN | Device Type |
|---|---|---|---|
| 4 | $FFFF5000 | 0 | QIC-02 streaming tape drive |
| 5 | $FFFF5100 | | |

# IOT Command Parameters

The following table lists the proper **IOT** command parameters for floppies used with boards such as the MVME328 and MVME172.

| IOT Parameter | Floppy Types and Formats | | | | | | |
|---|---|---|---|---|---|---|---|
| | DSDD5 | PCXT8 | PCXT9 | PCXT9_3 | PCAT | PS2 | SHD |
| Sector Size<br>0- 128 1- 256 2- 512<br>3-1024 4-2048 5-4096 = | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| Block Size:<br>0- 128 1- 256 2- 512<br>3-1024 4-2048 5-4096 = | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Sectors/Track | 10 | 8 | 9 | 9 | F | 12 | 24 |
| Number of Heads = | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Number of Cylinders = | 50 | 28 | 28 | 50 | 50 | 50 | 50 |
| Precomp. Cylinder = | 50 | 28 | 28 | 50 | 50 | 50 | 50 |
| Reduced Write Current Cylinder = | 50 | 28 | 28 | 50 | 50 | 50 | 50 |
| Step Rate Code = | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Single/Double DATA Density = | D | D | D | D | D | D | D |
| Single/Double TRACK Density = | D | D | D | D | D | D | D |
| Single/Equal_in_all Track Zero Density = | S | E | E | E | E | E | E |
| Slow/Fast Data Rate = | S | S | S | S | F | F | F |

| IOT Parameter | Floppy Types and Formats | | | | | | |
|---|---|---|---|---|---|---|---|
| | **DSDD5** | **PCXT8** | **PCXT9** | **PCXT9_3** | **PCAT** | **PS2** | **SHD** |
| **Other Characteristics** | | | | | | | |
| Number of Physical Sectors | 0A00 | 0280 | 02D0 | 05A0 | 0960 | 0B40 | 1680 |
| Number of Logical Blocks (100 in size) | 09F8 | 0500 | 05A0 | 0B40 | 12C0 | 1680 | 2D00 |
| Number of Bytes in Decimal | 653312 | 327680 | 368460 | 737280 | 1228800 | 1474560 | 2949120 |
| Media Size/Density | 5.25/DD | 5.25/DD | 5.25/DD | 3.5/DD | 5.25/HD | 3.5/HD | 3.5/ED |

**Notes**

1. All numerical parameters are in hexadecimal unless otherwise noted.
2. The DSDD5 type floppy is the default setting for the debugger.

# Network Controller Modules Supported

The following VMEbus network controller modules are supported by the 172Bug. The default address for each type and position is showed to indicate where the controller must reside to be supported by the 172Bug. The controllers are accessed via the specified CLUN and DLUNs listed here. The CLUN and DLUNs are used in conjunction with the debugger commands **NBH**, **NBO**, **NIOP**, **NIOC**, **NIOT**, **NPING**, and **NAB**, and also with the debugger system calls .NETRD, .NETWR, .NETFOPN, .NETFRD, .NETCFIG, and .NETCTRL.

| Controller Type | CLUN | DLUN | Address | Interface Type |
|---|---|---|---|---|
| MVME172 | $00 | $00 | $FFF46000 | Ethernet |
| MVME376 | $02 | $00 | $FFFF1200 | Ethernet |
| MVME376 | $03 | $00 | $FFFF1400 | Ethernet |
| MVME376 | $04 | $00 | $FFFF1600 | Ethernet |
| MVME376 | $05 | $00 | $FFFF5400 | Ethernet |
| MVME376 | $06 | $00 | $FFFF5600 | Ethernet |
| MVME376 | $07 | $00 | $FFFFA400 | Ethernet |
| MVME374 | $10 | $00 | $FF000000 | Ethernet |
| MVME374 | $11 | $00 | $FF100000 | Ethernet |
| MVME374 | $12 | $00 | $FF200000 | Ethernet |
| MVME374 | $13 | $00 | $FF300000 | Ethernet |
| MVME374 | $14 | $00 | $FF400000 | Ethernet |
| MVME374 | $15 | $00 | $FF500000 | Ethernet |

# Troubleshooting the MVME172: Solving Startup Problems

F

- Try these simple troubleshooting steps before calling for help or sending your CPU board back for repair.
- Some of the procedures will return the board to the factory debugger environment. (The board was tested under these conditions before it left the factory.)
- Selftest may not run in all user-customized environments

**Table F-1. Troubleshooting Steps**

| Condition | Possible Problem | Try This: |
|---|---|---|
| I. Nothing works. No display on the terminal. | A. If the RUN or +12V LED is not lit, the board may not be getting correct power. | 1. Make sure the system is plugged in.<br>2. Check that the board is securely installed in its backplane or chassis.<br>3. Check that all necessary cables are connected to the board, per this manual.<br>4. Check for compliance with System Considerations, in Chapter 2.<br>5. Review the Installation and Startup procedures, in Chapter 2. The step-by-step powerup routine for your board is in Chapter 3. |
| | B. If the LEDs are lit, the board may be in the wrong slot. | 1. The CPU board should be in the first (leftmost) slot if it is to be the system controller.<br>2. The “system controller” function requires that header J2 be set properly. See Chapter 2. |
| | C. The system console terminal may be configured wrong. | Configure the system console terminal according to the instructions in Chapter 3. |

## Table F-1. Troubleshooting Steps (Continued)

| Condition | Possible Problem | Try This: |
|---|---|---|
| II. There is a display on the terminal, but input from the keyboard has no effect. | A. The keyboard may be connected incorrectly. | Recheck the keyboard connections and power. |
| | B. Board jumpers may be configured incorrectly. | Check the board jumpers per this manual. |
| | C. You may have invoked flow control by pressing a HOLD or PAUSE key, or by typing<br>^**S**<br>Also, a HOLD LED may be lit on the keyboard. | Press the HOLD or PAUSE key again.<br>If this does not free up the keyboard, type in<br>^**Q**<br>(Hold down the CONTROL key and type a "Q") |
| III. Debug prompt<br>172-Bug><br>does not appear at powerup, and the board does not auto boot. | A. Debugger EPROM/Flash may be missing | 1. Disconnect *all* power from your system.<br>2. Check that the proper debugger EPROM is installed per this manual.<br>3. Remove the jumper from J28, pins 9 and 10. This enables use of the EPROM instead of the Flash memory.<br>4. Reconnect power. Restart the system.<br>5. If the debug prompt appears, go to step IV or step V, as indicated. If the debug prompt does not appear, go to step VI. |
| | B. The board may need to be reset. | |

## Table F-1. Troubleshooting Steps (Continued)

| Condition | Possible Problem | Try This: |
|---|---|---|
| IV. Debug prompt `172-Bug>` appears at powerup, but the board does not autoboot. | A. The initial debugger environment parameters may be set wrong. | 1. Start the onboard calendar clock and timer. Type in **set mmddyyhhmm <Return>** where the characters indicate the month, day, year, hour, and minute. The date and time will be displayed. |
| | B. There may be some fault in the board hardware. |  **Caution** Performing the next step will change some parameters that may affect your system operation.<br>2. Type in **env;d <Return>** This sets up the default parameters for the debugger environment.<br>3. When prompted to Update Non-Volatile RAM, type in **y <Return>**<br>4. When prompted to Reset Local System, type in **y <Return>**<br>After a cold start, the debug prompt: `172-Bug>` is displayed.<br>5. Change to the diagnostic directory by typing: **sd <Return>**<br>Now the prompt should be: `172-Diag>` |

F

## Table F-1. Troubleshooting Steps (Continued)

| Condition | Possible Problem | Try This: |
|---|---|---|
| | | 6. Run selftest by typing in<br>**st <Return>**<br>The tests take as much as 10 minutes, depending on RAM size. They are complete when the prompt returns. (The onboard selftest is a valuable tool in isolating defects.)<br>7. The system may indicate that it has passed all the selftests. Or, it may indicate a test that failed. If neither happens, enter<br>**de <Return>**<br>Any errors should now be displayed. If there are any errors, go to step VI. If there are no errors, go to step V. |
| V. The debugger is in system mode and the board autoboots, or the board has passed selftests. | A. No problems. Troubleshooting is done. | No further troubleshooting steps are required.<br>**Note** Even if the board passes all tests, it may still be bad. Selftest does not try out all functions in the board (for example, SCSI, or VMEbus tests). |
| VI. The board has failed one or more of the tests listed above, and can not be corrected using the steps given. | A. There may be some fault in the board hardware or the on-board debugging and diagnostic firmware. | 1. Document the problem and return the board for service.<br>2. Phone 1-800-222-5640. |
| YOU ARE FINISHED (DONE) WITH THIS TROUBLESHOOTING PROCEDURE. | | |

# Index


## Symbols

+12 Vdc power 1-15, 2-21

## Numerics

172Bug 4-1
- console port 4-9
- debugger command set 4-22
- debugger package 3-1
- description 3-1
- disk I/O support 3-16
- disk/tape controller data D-1
- implementation of 3-3
- installation 3-4
- network controller data E-1
- port 0 or 00 4-9
- port numbers 4-9
- prompt 3-8
- stack 3-15
- starting address 3-15
- vector table and workspace 4-13

172Bug (see debug monitor and MVME172BUG) 2-2
27C040 PROM 3-3
28F008SA Flash 3-3
5-1/4 DS/DD 96 TPI floppy drive D-2
53C710 (SCSI controller) 1-27
82596CA (see Ethernet and LAN) 1-26, 1-27, 3-21

## A

ABORT switch 1-15, 1-33, 3-13
adapter board (see P2 adapter board) 1-1
address 4-3
- as a parameter 4-5
- DRAM 2-20
- Flash/PROM 3-15
- formats 4-5

arbitration priority 1-30
arguments 4-3
arithmetic operators 4-3
ASCII string 4-3
assembler/disassembler 4-10
assertion 1-13
autoboot 3-9
autojumpering 2-18

## B

backplane connectors P1 and P2 2-19
backplane jumpers 2-18
Backus-Naur 4-3
base address of IndustryPacks A-16
base and top addresses 4-7
base identifier 4-4
Battery Backed Up RAM (BBRAM) and Clock A-3
battery backup function 1-18
battery backup select jumpers 2-10
battery care 1-19
battery-backed-up RAM (BBRAM) and clock 1-22
baud rates 1-23, 3-7
BG (bus grant) 2-18
BH (Bootstrap and Halt) 3-17
binary number 1-12
block diagram, MVME172 1-14
block size, logical 3-16

blocks versus sectors 3-16
BO (Bootstrap Operating System) 3-17
board connectors 1-32
Board Information Block (BIB) A-1
Board Mode, 172Bug 3-5
board-level hardware features 1-1
BOOTP protocol module 3-22
Bootstrap and Halt (BH) 3-17
Bootstrap Operating System (BO) 3-17
Bootstrap Protocol (BOOTP) 3-22
break function 3-13
BREAK key 3-13
burst transfers 1-16
bus grant (BG) 2-18
byte 1-13

## C

C programming language 3-3
cabling 2-19
cache 1-16
calling system utilities from user programs 4-11
checksum data A-3
CISC Embedded Controller(s) D-1
Clear To Send (CTS) 3-7
Clock 2-9
clock chip 1-22
clock select header (J16) 2-9
clock select header for serial port 2 (J17) 2-9
clock speed, MPU 3-14
CLUN (controller LUN) D-1, E-1
CNFG command A-1
command identifier 4-3
command line 4-1
configuration
  controllers/devices 3-20
  hardware 3-4
Configure (CNFG) and Environment (ENV) commands A-1
configuring
  base address of IndustryPacks A-16
  BIB (Board Information Block) A-1
  debug parameters A-3
  IndustryPacks A-16
  VMEbus interface A-10
connector P2 4-9
connectors 1-2, 1-32, 2-16
console port 4-9
console terminal F-1
control bit 1-13
controller LUN (CLUN) D-1
controller modules D-1
cooling requirements 1-10
count 4-3
creating a new vector table 4-16
creating a program with the assembler/disassembler 4-10
CTS (Clear To Send) 3-7

## D

data bus structure 1-16
data circuit-terminating equipment (DCE) 1-23, B-2
data terminal equipment (DTE) 1-23, B-1
date and time, setting 3-8
DCE (data circuit-terminating equipment) 1-23
debug monitor (see 172Bug and MVME172Bug) 2-2
debug port 4-9
debugger
  address parameter formats 4-5
  commands 4-22
  description 3-2
  general information 3-1
  prompt 4-1
decimal number 1-12
decoder, GCSR 1-38
default 172Bug controller and device parameters 3-20
default baud rate (see baud rates) 3-7
device LUN (DLUN) D-2, E-1
device probe function 3-17
diagnostics 3-2, 3-26

direct access devices D-2, D-4
direct memory access (DMA) 1-26
directories, switching 3-26
disk I/O
commands, 172Bug 3-17
error codes 3-20
support, 172Bug 3-16
via 172Bug commands 3-17
via 172Bug system calls 3-18
disk/tape controller
data D-1
modules supported D-1
DLUN (device LUN) D-2, E-1
DMA (direct memory access) 1-26
double precision real 4-20
double-button reset 3-11, A-3
download 4-10
downloading an S-record object file 4-10
DRAM
base address 2-20
mezzanines 3-15
options 1-17
performance 1-30
DTE (data terminal equipment) 1-23

## E

EIA-232-D
connection diagrams 2-30
connections B-1
interconnections B-1, B-2
port(s) 3-7, 4-9
ports 4-13
SIMM part numbers 2-6
EIA-485/EIA-422
connection diagrams 2-34
connections B-8
interconnections B-8
interface characteristics B-10
signals 1-26, B-9
EIA-530 1-26
connection diagrams 2-28
connections B-5
interconnections B-5
interface characteristics B-7
signals B-5
EIA-530/V.36 SIMM part numbers 2-6
elevated temperature operation 1-10
entering and debugging programs 4-10
entering debugger command lines 4-1
ENV command A-3
parameters A-4
setting up IPs A-16
Environment (ENV) and Configure (CNFG) commands A-1
environment commands 3-5
environment, operating 1-10
EPROM 1-21
and Flash 1-21
Size Select Header (J23) 2-12
size select header (J23) 2-12
EPROM/Flash cycle times 1-31
error codes, 172Bug 3-20, 3-23
error codes, disk I/O 3-20
ESDI Winchester hard drive D-3
Ethernet
controllers E-1
interface 1-27
packets 3-21
station address 1-27
transceiver interface 1-27
Ethernet (see 82596CA and LAN) 1-27, E-1
examples
displaying board information block A-1
exception vectors used by 172Bug 4-14
exponent field 4-19
expression 4-3
expression as a parameter 4-3
extended addressing 2-19

## F

facilities 3-26
factory debugger environment F-1
FAIL LED 1-15
false 1-13

FCC compliance 1-12
features 1-7
firmware overview 3-1
Flash 3-3
  memory 1-21
    initializing 3-8
    programming 3-24
flexible diskettes D-2
floating point unit (FPU) 4-18, 4-21
  instructions 4-18
  support 4-18
floppy diskettes D-4
floppy drive D-2, D-3
flow control F-1
FPU (floating point unit) 4-18, 4-21
front panel switches and indicators 1-15
functional description 1-15
fuse (F2) 2-21

## G

GCSR (Global Control and Status Registers) 2-20, 3-25
  board control register 1-39
general control register A-17
general information, debugger 3-1
general purpose readable jumpers header (J28) 2-12
global bus timeout 2-20
Global Control and Status Registers (GCSR) 2-20, 3-25
grounding
  proper B-11

## H

handshaking 3-7
hard disk drive D-3
hardware
  functions 4-13
  interrupts 1-29
  preparation 2-2
headers, setting 2-3, 3-4
hexadecimal character 1-12
host port 4-9
host system 4-10

## I

I/O
  commands
    IOC (I/O Control) 3-18
    IOI (Input/Output Inquiry) 3-17
    IOP (Physical I/O to Disk) 3-18
    IOT (I/O Teach) 3-18
  interfaces 1-23
  maps 1-33
IACK (interrupt acknowledge) 2-18
IndustryPack (IP)
  base address A-16
  configuration
    general control registers A-17
    interrupt control registers A-17
    memory size A-17
  configuring A-16
  interconnections C-1
  interfaces 1-26
  modules, installation 2-16
installation 3-4
  172Bug 3-4
  IP modules 2-16
  MVME172 2-17
  SIMMs 2-8
  transition modules 2-17
installation and startup 3-4
Intel 82596 LAN Coprocessor Ethernet Driver 3-21
interconnections
  IndustryPack C-1
  serial B-1, B-5, B-8
interface characteristics
  EIA-232-D B-4
Internet Protocol (IP) 3-21

INDEX

interrupt acknowledge (IACK) 2-18
interrupt control registers A-17
Interrupt Stack Pointer (ISP) 3-15
interrupts, programmable 1-29
introduction 1-1, 2-1
IOT command parameters D-5
IP (IndustryPack)
  bus clock header (J14) 2-14
  bus strobe select header (J19) 2-15, 3-6
  installation on the MVME172 2-16
  reset signal 1-27
  strobe signal C-3
IP2 chip 1-26
IP32 CSR bit 2-14
ISP (Interrupt Stack Pointer) 3-15

## J

J1 jumper 2-3
J12 jumpers 3-6
J14 jumper 2-14
J14 jumpers 3-6
J15 connector 2-5
J16 jumpers 2-9, 3-6
J17 jumpers 2-9, 3-6
J18 connector 1-32
J19 jumper 2-15, 3-6
J22 jumpers 2-10, 3-15
J23 jumper 2-12
J25 connector 1-32
J28 jumpers 1-21, 2-12
J6 connector 1-33
jumpers
  setting 2-3, 3-4
  user-definable 2-12

## L

LAN
  DMA transfers 1-31
  FIFO buffer 1-31
LAN (see 82596CA and Ethernet) 1-26, 1-27
LAN LED 1-15
layout, MVME172 2-4
LEDs 1-15, 1-33
local
  bus I/O devices memory map 1-36
Local Area Network (LAN) 1-26
local bus 1-16
  arbiter 1-30
  arbitration priority 1-30
  memory map 1-33
  timeout 1-29
local bus/VMEbus interface 1-22
local I/O devices memory map 1-35
local reset operation (LRST) 1-39
local resources 1-28
local-bus-to-DRAM cycle times 1-30
location monitors 2-20
logical unit number (LUN) (see CLUN or DLUN)
longword 1-13
LUN (logical unit number) (see CLUN or DLUN)

## M

mantissa field 4-19
manual terminology 1-12
map decoder, GCSR 1-38
MC2 chip 1-19
  LCSR 2-12
MC68060 TRAP instructions 4-11
MC68xx060 Cache 1-16
memory base addresses 1-30
memory maps 1-33
  local bus 1-33
  local I/O devices 1-35
  VMEbus 1-38
  VMEbus short I/O 1-38
memory options 1-17
memory requirements, 172Bug 3-15
memory size A-17
metasymbols 4-3
modem 1-24
MPAR (Multiprocessor Address Register) 3-24

MPCR (Multiprocessor Control Register) 3-23
MPU clock speed calculation 3-14
multi-MPU programming considerations 1-39
multiple MVME172s, installation 2-20
Multiprocessor Address Register (MPAR) 3-24
Multiprocessor Control Register (MPCR) 3-23
multiprocessor support 3-23
MVME172
  as Ethernet controller E-1
  board-level hardware features 1-1
MVME172 module installation 2-17
MVME172 specifications 1-9
MVME172 switches, headers, connectors, fuses, and LEDs 2-4
MVME172BUG 3-1
MVME172Bug 1-17, 1-21
MVME172BUG debugging package (see 172Bug and debug monitor) 2-2
MVME320 disk/tape controllers D-2
MVME323 disk/tape controller D-3
MVME327A D-3
MVME328 disk/tape controller D-4
MVME350 controller D-4
MVME374 Ethernet controller E-1
MVME376 Ethernet controller E-1
MVME712-12 1-1
MVME712-13 1-1
MVME712A 1-1
MVME712AM 1-1
MVME712B 1-1
MVME712M 1-1
  installation 2-17
MVME712x 1-1
MVME712x serial ports 1-25

## N

negation 1-13
network
  boot control module 3-22
  controller data E-1
  controller modules E-1
  I/O error codes 3-23
  I/O support 3-21
Network Auto Boot 3-10
no display F-1
Non-Volatile RAM (NVRAM) A-3
normal address range 1-34
no-VMEbus interface option 1-17, 1-21, 2-5, 2-13, 3-8
numeric value 4-4
NVRAM (Non-Volatile RAM) A-3

## O

object code 4-10
offset registers 4-7
operating environment 4-12
operational parameters A-3
option field 4-3
overview 1-1

## P

P1 connector 1-32, 2-19
P2 adapter board (see adapter board) 1-1
P2 connector 1-1, 1-32, 2-19, 4-9
panel, front 1-15
parameters (see default 172Bug controller and device parameters) 3-20
part numbers, SIMM 2-6
parts location diagram 2-4
port 1 or 01 4-9
port number(s) 4-3, 4-9
port numbers 4-9
ports for debugging 4-9
power-up 3-24
preserving the debugger operating environment 4-12
program execution 3-23, 3-25
programmable tick timers 1-28
programming considerations, multi-MPU 1-39

INDEX

PROM (see 27C040 PROM) 3-3
prompt, debugger 3-8
proper grounding B-11
pseudo-registers 4-7
publications
  Non-Motorola 1-4

## Q

QIC-02 streaming tape drive D-4

## R

range 4-3
RARP/ARP protocol 3-22
readable jumpers 2-12
reading a program from disk 4-11
receivers
  EIA-232-D B-4
  EIA-485 B-11
  EIA-530 B-8
related documentation 1-3
relative address+offset 4-7
remote status and control connector (J6) 1-33
RESET switch 1-33, 1-39, 3-12
resetting the system 1-39, 3-11, A-3
resources, local 1-28
restarting the system 3-11
Reverse Address Resolution Protocol (RARP) 3-22
RF emissions 1-12
RFI (radio frequency interference) 2-17
ROMboot 3-10
RUN LED 1-15

## S

SCC (Serial Communications Controller) (see Z85230) 1-23, 3-7
scientific notation 4-21
SCON LED 1-15
SCSI
  Common Command Set (CCS) D-2
  controller (53C710) 1-27
  FIFO buffer 1-31
  interface 1-28
  LED 1-15
  termination 1-28
  terminator power 2-21
  transfers 1-31
SCSI Common Command Set (CCS) D-4
SD (switch directories) command 3-26
sector size 3-16
self-test routines 3-24
sequential access devices D-2, D-4
Serial Communications Controller (SCC) (see Z85230) 1-23, 3-7
serial communications interface 1-23
serial interconnections B-1
Serial Interface Module (SIMM)
  installation 2-8
  part numbers 2-6
  removal 2-6
  selection 2-5
serial interface module (SIMM)
  model numbers 1-24
serial interface parameters B-4
serial interface signals B-1
serial interfaces and transition boards 1-26
serial port 2 4-9
Serial Port 2, MVME712x 1-24
Serial Port 4, MVME712x 1-25
serial port B EIA-485/EIA-422 interconnect signals B-9
serial port B EIA-530 interconnect signals B-5
serial port interface 1-23
serial ports B-1
Set Environment to Bug/Operating System (ENV) command A-3
short I/O space 1-38, 3-26
sign field 4-19
signals 1-26
SIMMs 1-26, 2-5
  installation 2-8
single precision real 4-19
slave address decoders A-10

snooping 1-16
software initialization 1-39
software-programmable hardware interrupts 1-29
source line 4-10
special considerations for elevated temperature operation 1-10
specifications, board 1-9
SRAM (static RAM) 1-18
  battery backup source select header (J22) 2-10
  options 1-18
S-record format 4-10
stack 3-15
startup, 172Bug 3-4
STAT (status) LED 1-15
static RAM (SRAM) 1-18
static variable space 3-15
status bit 1-13
streaming tape drive (see QIC-2 streaming tape drive) D-4
string literal 4-4
strobe signal C-3
structure, data bus 1-16
switches and indicators 1-15
switching 3-26
switching directories (SD command) 3-26
synchronous/asynchronous protocols 1-23
syntactic variables 4-3
SYSFAIL* signal 3-14
system
  calls, TRAP #15 3-19
  considerations 2-19
  console 3-7
  controller function 3-5
  reset (SRST) 1-39
system console F-1
System Fail (SYSFAIL*) signal 3-10
System Mode, 172Bug 3-5

## T

temperature, high 1-10
terminal input/output control 3-16, 4-1
termination, SCSI 1-28
TFTP protocol 3-22
tick timers 1-28
time-of-day clock 1-22
timeout
  function 1-29
  global bus timeout 2-20
  local bus 1-29
timing performance 1-30
transfer type (TT) signals 1-34
transition boards and serial interfaces 1-26
transition module serial ports 1-24
transmitters
  EIA-232-D B-4
  EIA-485 B-10
  EIA-530 B-7
TRAP #15 4-11
TRAP #15 system calls 3-18
Trivial File Transfer Protocol (TFTP) 3-22
troubleshooting steps F-1
true 1-13
TT (see transfer type) 1-34
TX and RX clocks 1-25

## U

UDP/IP protocols 3-21
unpacking instructions 2-1
user-customized environments F-1
using the 172Bug debugger 4-1

## V

vector table 4-13
VME LED 1-16
VMEbus
  accesses to the local bus 1-38
  interface and VMEchip2 1-22
  interface, configuring A-10
  memory map 1-38
  short I/O memory map 1-38
  specification 1-4

INDEX

VMEbus, "no" option 1-17, 1-21, 2-5, 2-13, 3-8
VMEbus/local bus interface 1-22
VMEchip2 1-22
GCSR (Global Control and Status Registers) 2-20, 3-25

## W

watchdog timer 1-28
Winchester hard drive D-2, D-3
word 1-13

## X

XON/XOFF 3-7

## Z

Z85230 Serial Communications Controller (SCC) 1-23, 3-7

INDEX

**Artisan Technology Group** is an independent supplier of quality pre-owned equipment

## Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

## We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

## Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP