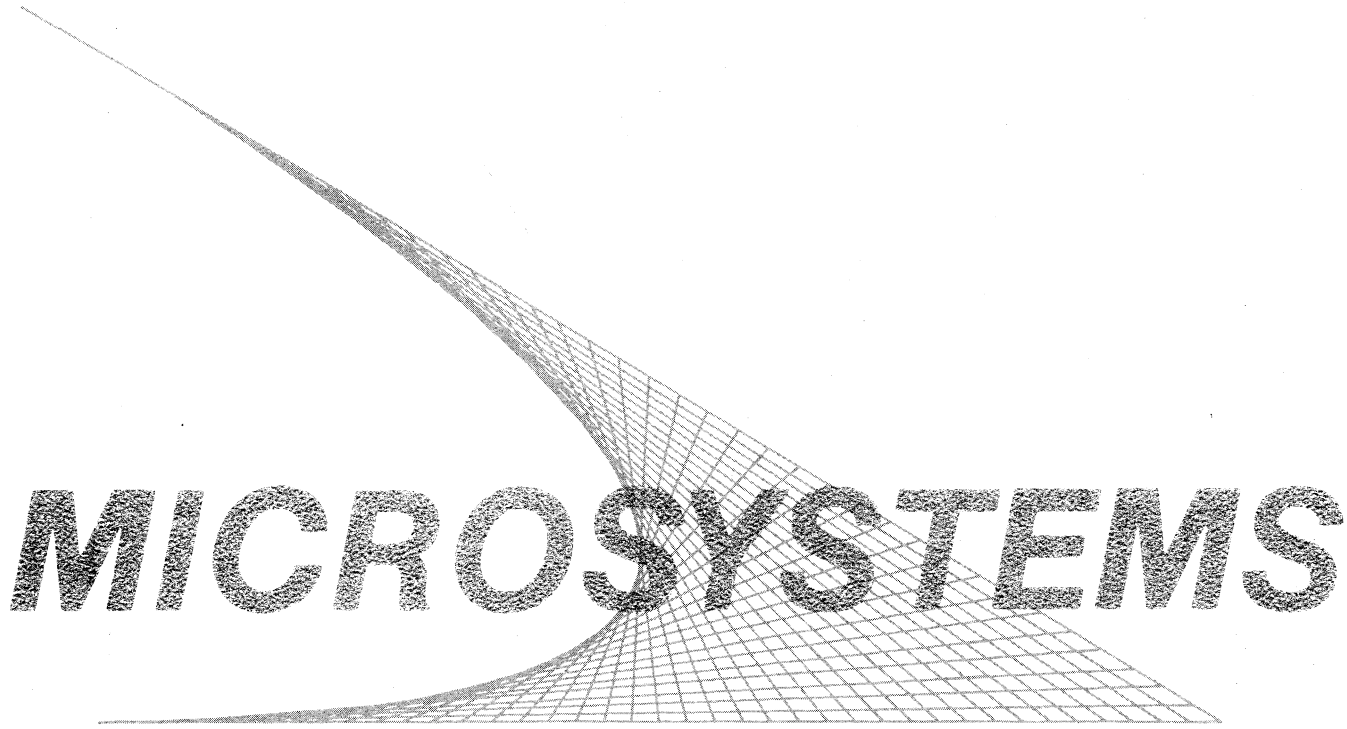
MOTOROLA

MVME121SYS/D1

MVME121-Core System and MVME319/320 Controllers for SYSTEM V/68



QUALITY • PEOPLE • PERFORMANCE

MVME121SYS/D1
July 1985

MVME121-CORE SYSTEM AND MVME319/320 CONTROLLERS FOR SYSTEM V/68

First Edition

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 General

Two document types provide information about SYSTEM V/68: manuals and guides. The manuals describe commands, facilities, features, and error messages of the system. The guides provide supplemental details and instructions for system implementation, administration, and use. The manuals are organized as alphabetized entries within tabbed sections. The *SYSTEM V/68 User's Manual* contains sections 1 through 6. The *SYSTEM V/68 Administrator's Manual* contains sections 1M, 7, and 8. Throughout the documentation, references to these manuals are given as *name(section)*. For example, *chroot*(1M) is a reference to the *chroot* entry in section 1M of the *SYSTEM V/68 Administrator's Manual.* The following conventions identify arguments, literals, commands, and program names:

Boldface strings are literals and are to be typed as they appear.

Italic strings represent program names or substitutable argument prototypes.

Items within **[ ]** s are optional.

<cr> represents the "carriage return" key, RETURN. All operator inputs are followed by a carriage return.

In addition, SYSTEM V/68, Release 2, Version 1.1 incorporates a new convention for naming disk and tape devices. In releases prior to Release 2, Version 1, the standard format for naming disk devices was

/dev/dk*xy*

where *x* referred to the disk device number and *y* referred to the disk section or partition. Raw access to a disk device was indicated with an **r**, e.g., **/dev/rdk01.**

Standard format for naming tape devices was

/dev/mt*x*

where *x* referred to the magnetic tape device number. Raw access to a tape device was indicated with an **r**, e.g., **/dev/rmt0.**

Beginning with Release 2, Version 1, the disk and tape device naming convention creates separate subdirectories under **/dev** for each type of disk or tape device. The new format for disk devices is:

/dev/ {**r**} **dsk/** [**r**] *cntrlr*_[*controller_number***d**] *drive_number***s***section_number*

Fields in square brackets are entirely optional. The fields do not affect the operation of any software or hardware; they are for informational purposes only, for the convenience of administrators, operators, and users. Fields in curly brackets represent options that affect software; they must be present if that option is being selected.

r	(Not Required) (The first r) indicates a raw interface to the disk. The default is normal system buffering.
dsk/	(Required) Indicates that the device is a disk.
r	(Not Required) (The second r) indicates that this disk is on a remote system.

*cntrlr_* (Not Required) Indicates the appropriate disk device in systems with multiple disk drivers. In Release 2, Version 1 of SYSTEM V/68, the controller names are present and must be used on command lines that specify a disk device. Once the system has been installed, the system administrator may elect to eliminate the disk specification on single-drive systems. The disk devices available are:

- **vm21_** Intelligent Universal Disk Controller, M68KVM21
- **vm22_** Intelligent SMD Disk Controller with Floppy Disk, M68KVM22
- **wd_** Winchester Disk Controller, M68RWIN1
- **ud_** General Universal Disk Controller
- **c_** Generic controller
- **id_** Intelligent Disk Controller, MVME319
- **m320_** Intelligent Disk Controller, MVME320

*controller_number***d** (Not Required) System administrators decide whether or not to specify the controller number in the disk device name. If the controller number is specified, the **d** introduces the drive number.

drive_number (Required) The drive number. The field is free format; there is no default drive number.

s*section_number* (Required) The section number. The field is free format; there is no default section number.

As an example, the name for disk drive 0, section 0 might be **/dev/dsk/m320_0s0**.

The new format for tape device names is:

/dev/ {**r**} **mt/** [**c***controller_number***d**] *drive_number* [*density*] { **n** }

where:

r (Not Required) Indicates a raw device. The default is a blocked device.

mt/ (Required) Indicates a magnetic tape device.

c*controller_number***d** (Not Required) The **c** introduces the controller number. System administrators decide whether or not to specify the controller number in the tape device name. If the controller number is specified, the **d** introduces the device number.

device_number (Required) The drive number. The drive number is followed immediately by the density value of the tape.

density (Required) Tape density must be specified for each drive. The density is indicated with an **h**, **m**, or **l**, where:

- **h** (high) is a tape density of 6250 bpi
- **m** (medium) is a tape density of 1600 bpi

	l (low) is a tape density of 800 bpi
n	(Not Required) Indicates no rewind on close. The default condition is to rewind.

As an example, the name for a 6250 bpi magnetic tape drive might be **/dev/mt/0h.**

To make the transition to the new names less painful, the old names can exist in SYSTEM V/68 Release 2, Version 1. However, all documentation and sample shell scripts distributed with this release use the new naming convention. System administrators are encouraged to rename existing devices and incorporate the new names into shell scripts as soon as possible.

The following table compares existing device filenames with the new filenames that will be found in the documentation.

Disk Devices		Tape Devices	
Old Disk Name	New Disk Name	Old Tape Name	New Tape Name
/dev/dk00	**/dev/dsk/***cntrlr*_**0s0**	**/dev/mt01**	**/dev/mt/0l**
/dev/dk10	**/dev/dsk/***cntrlr*_**1s0**	**/dev/mt5**	**/dev/mt/5mn**
/dev/rdk00	**/dev/rdsk/***cntrlr*_**0s0**		

1.2 Contents

This manual contains hardware and software installation instructions, and administrative procedures for an MVME121-based microcomputer system configured with either the MVME319 Intelligent Disk Controller (hereafter referred to as the MVME319) or the MVME320 VMEbus Disk Controller (hereafter referred to as the MVME320) and operating under SYSTEM V/68.

An MVME121-based system configured with the MVME319 controller can support 5¼-inch diskettes, 8-inch floppy disks, floppy tape and 40Mb fixed disk media. The MVME320 controller supports 5¼-inch diskettes and 40Mb fixed disk media.

Much of the hardware, software and administrative information in this manual is controller-specific, and this is reflected in the manual organization. Chapters 2, 3, and 4 contain information for the MVME121/MVME319 system: operator information, installation procedures and error messages. Chapters 5, 6, and 7 contain information for the MVME121/MVME320 system: operator information, installation procedures and error messages. Chapter 8 contains new and revised pages from the *SYSTEM V/68 Administrator's Manual* and the *SYSTEM V/68 User's Manual* that may be required for the MVME121 microprocessor, the MVME319 controller, or the MVME320 controller.

Administrators, operators and users need read only the chapters directed to their particular system configuration.

NOTE: The MVME319 is supported by the MVME120 debug monitor; the MVME320 is supported by both the MVME120 and MVME12x debuggers. The MVME121 microprocessor is shipped with the MVME12x debugging package. The MVME120 debug monitor may be obtained as a separate product. Part numbers for the MVME120bug are:

MVME120BUG	MVME120BUG Debugging Package for use with MVME120 Family Boards

M68VIFSBG120	MVME120 Board Family Debug Source and Object Modules on 8-inch diskettes
M68VIXSBG120	MVME120 Board Family Debug Source and Object Modules on 5¼-inch diskettes

Chapter 2 contains hardware information for an MVME121-based system configured with an MVME319 and operating under SYSTEM V/68.

Chapter 3 contains instructions for installing the SYSTEM V/68 Release 2, Version 1.1 object and/or source distributions on an MVME121/MVME319 system. The chapter also contains specific administrative procedures such as formatting media and setting system variables.

Chapter 4 describes error messages that originate in the MVME319 driver and suggested corrective actions.

Chapters 5, 6, and 7 contain hardware information, installation instructions and error messages, respectively, for the MVME121/MVME320 system.

Chapter 8 contains pages from the *SYSTEM V/68 Administrator's Manual* and *SYSTEM V/68 User's Manual* that may be required for the MVME121 microprocessor, the MVME319 controller, or the MVME320 controller.

This manual is meant as a complement to the information presented in the basic SYSTEM V/68 documentation set. Administrators, operators and users are advised to familiarize themselves with the basic documentation.

1.3 Getting Started

SYSTEM V/68 arrives with an abundance of reference documentation. To get started with an MVME121-based system, users are advised to install and configure the hardware following the directions provided in Chapter 2 or Chapter 5 of this manual, depending on the system controller.

Next, follow the step-by-step installation contained in Chapter 3 (MVME121/MVME319) or Chapter 6 (MVME121/MVME320). If questions arise, refer to the administrative procedures contained in that chapter, or consult the *SYSTEM V/68 Administrator's Guide.*

After the software is installed, novice users may wish to familiarize themselves with SYSTEM V/68, starting with the *SYSTEM V/68 User's Manual*, described in the following section. More experienced users may go directly to the manual or guide most important for their application. The *SYSTEM V/68 System Release Description* describes new facilities and transition information for those facilities. The *SYSTEM V/68 Customer Letter* contains information about media compatibility and the Release 2, Version 1 partitioning scheme.

The next section includes a description of each manual included in the basic SYSTEM V/68 documentation set.

1.4 Documentation

1.4.1 MVME121-based System Related Documentation. Manuals referenced in this document and not part of the basic SYSTEM V/68 set are listed with their part numbers in the following table. These publications may provide additional helpful information and may be obtained from Motorola's Literature Distribution Center, 616 West 24th Street, Tempe, AZ 85282; telephone (602) 994-6561.

Vendor manuals for the Computer Memories, Inc., TEAC, and Micropolis drives have been reprinted under Motorola cover with a Motorola document number and may be obtained from the Literature Distribution Center.

Table 1-1. Related Documentation for MVME121-based System

MANUAL	PART NUMBER
MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual	MVME050
MVME120, MVME121, MVME122, MVME123 VMEbus Microprocessor Module User's Manual	MVME120
MVME120 Debug Monitor User's Manual	MVME120BUG
MVME12x Debug Monitor User's Manual	MVME12XBUG
MVME200/201 64K/256K Byte Dynamic Memory Module User's Manual	MVME200
MVME202/222-1/222-2 512KB/1MB/2MB Dynamic Memory Module User's Manual	MVME202
MVME319 Intelligent Disk Controller User's Manual	MVME319
MVME320 VMEbus Disk Controller User's Manual	MVME320
MVME702 Disk Interface Module User's Manual	MVME702
Computer Memories Model CM5000 Series Winchester Disk Drive User's Manual	M68KVSWD1
Teac Model FD-55 Series Floppy Disk Drive User's Manual	M68KVSFD2
Micropolis Model 1300 Series Winchester Disk Drive User's Manual	M68KVSWD2
The following vendor manuals are available from the manufacturers:	
ACB4000 Winchester Disk Controller OEM Manual	
BASF6138 Mini-Disk Drive Specification	
Cipher 525CT Floppy Tape Product Description	
Coutant ML300 Power Supply Handbook	
Vertex V100 Series Maintenance Manual for Winchester Disks (40Mb)	

1.4.2 SYSTEM V/68 Documentation. SYSTEM V/68 Release 2, Version 1.1 documentation is divided into two classes:

1. Basic Documents. These documents are considered essential for normal operation of SYSTEM V/68. One copy of each document is included as part of the purchase price of the operating system.
2. Supplemental Documents. These documents provide information and learning aids for new users. These do not automatically come with the software.

The documents available in each of these classes are described in the following paragraphs. The document title is followed by its marketing number, by which it can be ordered. To order documents, call or write

Literature Distribution Center
616 West 24th Street
Tempe, AZ 85282
(602) 994-6561

1.4.2.1 Basic Documents. A copy of each of the following documents is included as part of the purchase price of SYSTEM V/68:

- **Administrator's Guide, M68KUNAG/D2.** This guide is a collection of information needed to install, administer and maintain the SYSTEM V/68 software. Major topics: Administrative Guidelines, Setting Up the System, Accounting, File System Checking (*fsck*), Line Printer Spooling, System Activity Package.
- **Administrator's Manual, M68KUNAM/D2.** This manual is a reference volume for those who administer SYSTEM V/68. Major topics: System Maintenance Commands and Application Programs, Special Files and Device Driver Information, Facility Descriptions and Software Maintenance Procedures.
- **Assembler User's Guide, M68KUNASG/D3.** This guide provides information about the assembler (*as*) for the MC68000, MC68010, MC68020-based processors. It includes instructions for the MC68881 floating point co-processor. Major topics: General Syntax Rules, Segments, Location Counters, Labels, Expressions, Pseudo-Operations, Span-Dependent Optimization, Address Mode Syntax, Machine Instructions.
- **Document Processing Guide, M68KUNDP/D2.** This guide provides information about text editing features of SYSTEM V/68 and programs that are used to format a document in a user-controlled system. Major topics: Advanced Editing and Stream Editor. Several sections provide information about text formatting capabilities that are part of the unbundled Documenter's Workbench software. These include: *nroff* and *troff* Formatters, Table Formatting (*tbl*), Mathematics Typesetting (*eqn/neqn*), Memorandum Macros (*mm*), and Viewgraphs and Slides Macros (*mvt*).
- **Error Message Manual, M68KUNMSG/D2.** This manual provides information about error messages printed on the console by SYSTEM V/68. For each possible error message, the manual provides: message type, message description, suggested corrective action, and references to message origin.
- **Graphics Guide, M68KUNGG/D1.** This guide describes the SYSTEM V/68 graphics facility for producing high resolution graphs, drawings, and pictures on Motorola VME/10, Tektronix 4010 series terminals, and Hewlett-Packard 7221A graphic plotters. Major topics: Overview of the Graphics Facility, Graphics Editor, Administrative Information on the Graphics Facility.
- **Common Link Editor Reference Manual, M68KUNLDM/D1.** This document describes link editor functions, including combining object files, performing relocation, resolving external references, and processing symbolic debugging information. Major topics: Using the Link Editor, Command Language, Error Messages, Special Procedures, Syntax for Input Directives.
- **Operator's Guide, M68KUNOG/D3.** This guide contains operator function information used during hardware installation and normal system operations. Major topics: Console Operations, Hardware System Specifics (configuration and requirements), Boot Procedures, and System Shutdown Procedures.

- **Programming Guide, M68KUNPG/D1.** This guide describes the programming language and language aids available on SYSTEM V/68. Major topics: C programming language, C program checker (*lint*), FORTRAN 77 and its variants, libraries, *curses/terminfo*.
- **System Release Description, M68KUNRD/D3.** This document describes new facilities of and transition information on SYSTEM V/68 Release 2, Version 1. Major topics: Release Overview, Detailed Feature Descriptions, Performance, Transition Information, Known Bugs, Fixed Bugs.
- **Support Tools Guide, M68KUNSTG/D1.** This guide describes the software development tools available with SYSTEM V/68 Release 2, Version 1. Major topics: Maintaining Computer Programs (*make*), Source Code Control System (*sccs*), Macro Processor (*m4*), Desk Calculator Languages (*bc* and *dc*), Lexical Analyzer Generator (*lex*), the *awk* Language, Yet Another Compiler Compiler (*yacc*), and Common Object File Format (COFF).
- **User's Guide, M68KUNUG/D1.** This guide contains information for beginners and tutorials that acquaint a user with SYSTEM V/68 and most of the tools available. Major topics: an Introduction to SYSTEM V/68, Basics for Beginners, Introduction to the *ed* and *vi* editors, Introduction to the Shell, UNIX to UNIX CoPy (*uucp*) Tutorial.
- **User's Manual, M68KUNUM/D1.** This manual provides information for the average user of SYSTEM V/68. Major topics: Commands and Applications Programs, System Calls, Library Functions and Subroutines, File Formats, Miscellaneous Facilities, Games.
- **Customer Letter, SYSTEM V/68 Release 2, Version 1, M68KV68LET/L4.** This letter provides information about the release, including a list of files contained on the release media.

1.4.2.2 Supplemental Documentation. The following documents are not included as part of the purchase price of SYSTEM V/68 Release 2, Version 1. They are available from the Literature Distribution Center listed above.

- **C Primer Plus, TB310.** This book teaches the basics of the C programming language. Major topics: Data Types, Character Strings, Operators, Expressions, Statements, Input/Output Functions and Redirection, Choosing Alternatives, Loops, Functions, Storage Classes and Program Development, C Preprocessor, Arrays and Pointers, Structures, C Library and File I/O. A reference card is also included. This book was written by Mitchell Waite, Stephen Prata, and Donald Martin, and was published by Howard Sams & Co., Inc. (Indianapolis, 1984).
- **UNIX SYSTEM V PRIMER, TB311.** This book presents information for beginners about many features of SYSTEM V/68. Major topics: Getting Started, Electronic Mail and On-Line Help, Files and Directories, Text Editors, Programming Languages, the Shell, Information Processing. The book also includes two reference cards, one on SYSTEM V/68 commands and one on text editing commands. This book was written by Mitchell Waite, Donald Martin, and Stephen Prata, and was published by Howard Sams & Co., (Indianapolis, 1984).

NOTES

2. OPERATOR INFORMATION FOR MVME121/MVME319

2.1 General

This chapter describes the hardware configuration for a MVME121-based system with an MVME319 running on SYSTEM V/68 software. The following information is provided:

- Minimum system configuration
- Default system configuration
- Hardware installation
- Boot procedures
- Board-level operations

2.2 Minimum System Configuration

A minimum configuration for a MVME121-based system with an MVME319 that supports SYSTEM V/68 software includes the following components:

a. MVME121 VMEbus Microprocessor Module with 512Kb of RAM, the MC68451 MMU, 4Kb logical instruction cache, and one serial communication port.

b. MVME050 System Controller Module with two serial communication ports and one Centronics printer interface

c. MVME701 I/O Transition Module for the controller module

d. MVME202 512Kb Dynamic Memory Module with byte parity

e. MVME319 Intelligent Disk Controller with diskette and FloppyTape interfaces, and an SASI/SCSI bus host adapter for hard disk controllers

f. ACB4000 Adaptec Winchester Disk Controller

g. 40Mb Micropolis or Vertex 5¼-inch Winchester drive

h. 655Kb BASF 5¼-inch diskette drive

i. 24Mb CIPHER streamer tape

j. VME card cage with 300W power supply and ventilation

k. MVME120bug firmware-resident monitor for loading and debugging programs and for booting SYSTEM V/68.

2.3 Default System Configuration

SYSTEM V/68 is configured to support the following system components by default:

- One MVME121 monoboard microcomputer (including one terminal interface)
- One MVME050 system controller and one MVME701 I/O transition module (including two serial and one Centronics interface)
- Two MVME319 intelligent disk controllers
- Two Adaptec Winchester disk controllers
- Eight Winchester disk drives (40Mb, 6 data surfaces, 830 cylinders)

2

- Four 5¼-inch diskette drives
- Two CIPHER DATA PRODUCTS CT525 FloppyTape
- Several additional MVME202 or MVME222 memory modules

2.4 Hardware Installation.

This section describes the jumper and switch settings required to run an MVME121-based system under the default configuration of SYSTEM V/68. A non-standard hardware configuration must be reflected in the SYSTEM V/68 *master*(4) file and *dfile*(4) file, and requires the generation of a new kernel. For further information, refer to *master*(4), *dfile*(4), and *config*(1M).

2.4.1 Installation Considerations. Provide adequate grounding. It is important that the chassis ground is interconnected (e.g., drives, enclosure), and wired to the signal ground in one place only, typically at the power supply.

CAUTION

AVOID TOUCHING AREAS OF INTEGRATED CIRCUITRY; STATIC DISCHARGE CAN DAMAGE THESE CIRCUITS.

CAUTION

INSERTING/REMOVING MODULES WHILE POWER IS APPLIED MAY RESULT IN DAMAGE TO MODULE COMPONENTS.

2.4.2 MVME121 Microprocessor. SYSTEM V/68 requires an MVME121 microprocessor with a Memory Management Unit (MMU) installed and 512Kb of on-board memory.

The terminal interface for the MVME121 may be any TTY compatible terminal with a 25-pin RS-232C standard interface. The switch settings should be set as follows:

9600 Baud
Full duplex
8 data bits
1 stop bit
No parity

On the MVME121, the S3 switch settings are dependent on the jumpering for J20 and J21. To support SYSTEM V/68, the S3 switch settings should be set as follows:

SWITCH NUMBER	DESCRIPTION	SETTING
S3-1	Autoboot Select	Open
S3-2	Baud Rate Select (12.5-10 MHz)	Closed
S3-3	Reset Vector Fetch Select	Closed
S3-4	Reset Vector Fetch Select	Open

The MVME121 SYSTEM V/68-supported jumper settings are listed in Table 2-1 and illustrated in Figure 2-1. For more detailed information about the MVME121 microprocessor, refer to the *MVME120, MVME121, MVME122, MVME123 VMEbus Microprocessor Module User's Manual.*

Table 2-1. MVME121 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
J2	ACFAIL/SYSFAIL Select	2-3
J3	VMEbus Request Level Select	1-3, 4-6, 5-7, 9-11
J4	VMEbus Request Level Select	3-5, 4-6
J5	Abort Switch Disable Select	1-2
J6	Reset Switch Disable Select	1-2
J7	VMEbus Interrupt Connection Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14
J8	ROM/EPROM Device Configuration Select	1-2, 5-7, 9-10 11-12, 13-14
J9	Cache Configuration Select	1-2
J17	Cache Configuration Select	No Jumpers
J10	MMU Strobe Select	2-3
J11	MMU Address Strobe Select	No Jumpers
J12	Map Decode Time Select	2-3
J13	RAM Cycle Timing Select	1-2
J14	RAM Cycle Timing Select	1-2
J16	RAM Cycle Timing Select	2-3
J18	RAM Cycle Timing Select	2-3
J19	RAM Cycle Timing Select	2-3
J15	Refresh Period Select	No Jumper
J20	MSR Bit 1 Select	No Jumper
J21	MSR Bit 2 Select	2-3
J22	ROM Access Time Select	7-8
J23	RAM Size Select	1-2
J24	Reset Vector Fetch Mode Select	1-2
J25	Local Time-Out Select	1-2
	MVME121 BACKPLANE JUMPER SETTINGS	
A21-A22	IACKIN* TO IACKOUT*	No Jumpers
B4-B5	BG0IN* TO BG0OUT*	No Jumpers
B6-B7	BG1IN* TO BG1OUT*	No Jumpers
B8-B9	BG2IN* TO BG2OUT*	No Jumpers
B10-B11	BG3IN* TO BG3OUT*	No Jumpers

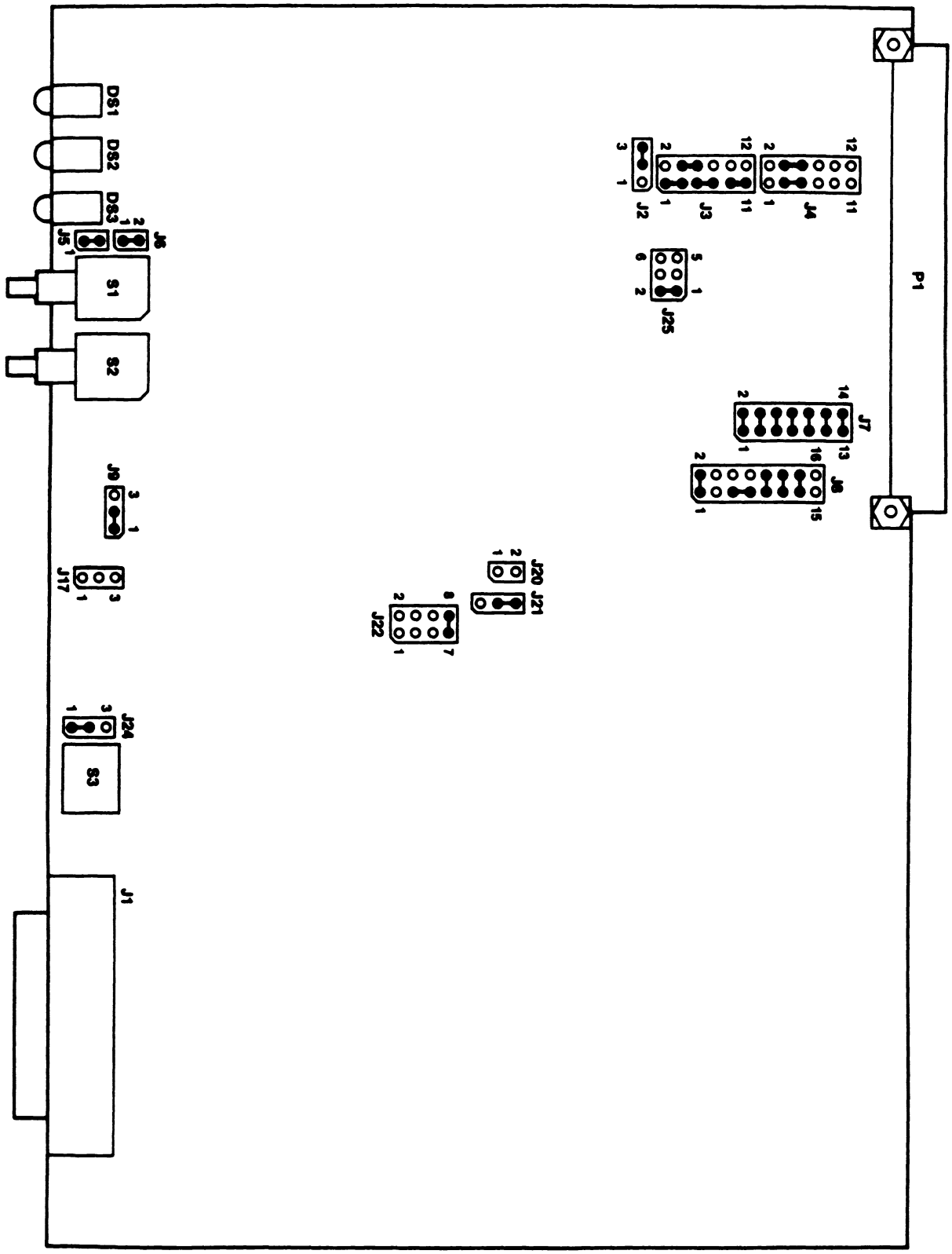


Figure 2-1. MVME121 Jumper Locations and Settings

2.4.3 MVME050 System Controller and MVME701 Transition Modules. SYSTEM V/68 supports the parallel port on the MVME050 controller module as a printer port. Header J29 selects the rising or falling edge of the printer acknowledge signal. The module is shipped with the jumper installed between pins 1 and 2 with the rising edge selected. The module is initialized to provide a Level 2 interrupt for the line printer. These are the configurations supported by the line printer driver for the MVME050 module. The line printer driver for the MVME050 module does not support unbuffered parallel to serial port conversion using the parallel port. The two serial ports on the MVME050 (or MVME701) interface with the terminals.

Tables 2-2 and 2-3 list the SYSTEM V/68-supported jumper settings for the two modules. Notice that Table 2-2 contains several jumpers that are considered "open." For these jumpers, SYSTEM V/68 supports any configuration selected by users. Figures 2-2 and 2-3 illustrate the jumper locations and settings for the MVME050 and MVME701 respectively. Notice that Figure 2-2 contains reference jumper settings for those jumpers that are considered "open" in Table 2-2. That is, Table 2-2 indicates SYSTEM V/68 supports all user-selected configurations for the "open" jumpers, and Figure 2-2 illustrates one standard configuration. Users may follow the jumpering shown in Figure 2-2 and eliminate the need to refer to the MVME050 manual.

The jumpering illustrated in Figure 2-2 differs from the MVME050 factory configuration at jumper locations J22, J23, and J24. The jumpering illustrated in Figure 2-3 differs from the MVME701 factory configuration at jumper locations J5, J6, J9, and J10.

The piano type 8-section switch on the front panel of the controller module, (S1), is a software readable switch. The user may include the functions of this switch in the software program. Section 8 of the switch may also function as blanking for the user status display.

For more detailed information about the MVME050 or MVME701, refer to the *MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual.*

2

Table 2-2. MVME050 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
J1	EPROM Base Address Select (A14-A23)	Open*
J2	EPROM Base Address Select (A24-A31)	Open*
J8	EPROM Size Select	Open*
J13	EPROM Quad Enable Select	Open*
J27	EPROM Access Time Select	Open*
J9	RAM Base Address Select (A14-A23)	Open*
J10	RAM Base Address Select (A24-A31)	Open*
J26	RAM Access Time Select	Open*
J14	RAM Quad Enable Select	Open*
J15	RAM Size Select	Open*
J12	AM1E Enable Select (EPROM Quad 1)	Open*
	AM16 Enable Select (EPROM Quad 2)	Open*
J3, J16, J17	EPROM/RAM Configuration Select (EPROM/RAM Quad 1)	Open*
J11, J18, J19	EPROM/RAM Configuration Select (EPROM/RAM Quad 2)	Open*
J4	Bus Time-Out Select	9-10
J5	Serial Clock Damping/Shorting Select	1-2
J6	System Clock Damping/Shorting Select	1-2
J7	System Controller Select	1-2, 3-4, 5-6, 7-8
J20	I/O Base Address Select	1-2, 3-4, 5-6, 9-10, 11-12, 13-14
J21	Time-of-day Clock Power and Battery Charge Select	1-3, 2-4, 7-8
J22	System Fail or Ground for Interrupt Source Select	1-2
J23	Internal/External Transmit Clock Serial Port 1 Select	1-2
J24	Internal/External Transmit Clock Serial Port 2 Select	1-2
J25	Display Blanking Enable Select	1-2
J28	Reset Switch Disable Select	1-2
J29	Printer Acknowledge Edge Select	1-2

Open*: Any configuration selected by users as appropriate for their applications is supported by SYSTEM V/68.

MVME050 BACKPLANE JUMPER SETTINGS		
A21-A22	IACKIN* TO IACKOUT*	No Jumpers
B4-B5	BG0IN* TO BG0OUT*	No Jumpers
B6-B7	BG1IN* TO BG1OUT*	No Jumpers
B8-B9	BG2IN* TO BG2OUT*	No Jumpers
B10-B11	BG3IN* TO BG3OUT*	No Jumpers

Table 2-3. MVME701 SYSTEM V/68-Supported Jumpers

JUMPER NUMBER	DESCRIPTION	SETTING
J5	Port 1 to Terminal Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14 15-16, 17-18, 19-20
J6	Port 1 to Modem Select	No Jumpers
J11	Port 1 CTS Select	2-3
J7	Port 1 DSR Select	2-3
J9	Port 2 to Terminal Select	No Jumpers
J10	Port 2 to Modem Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14 15-16, 17-18, 19-20
J8	Port 2 DSR Select	2-3
J12	Port 2 CTS Select	2-3

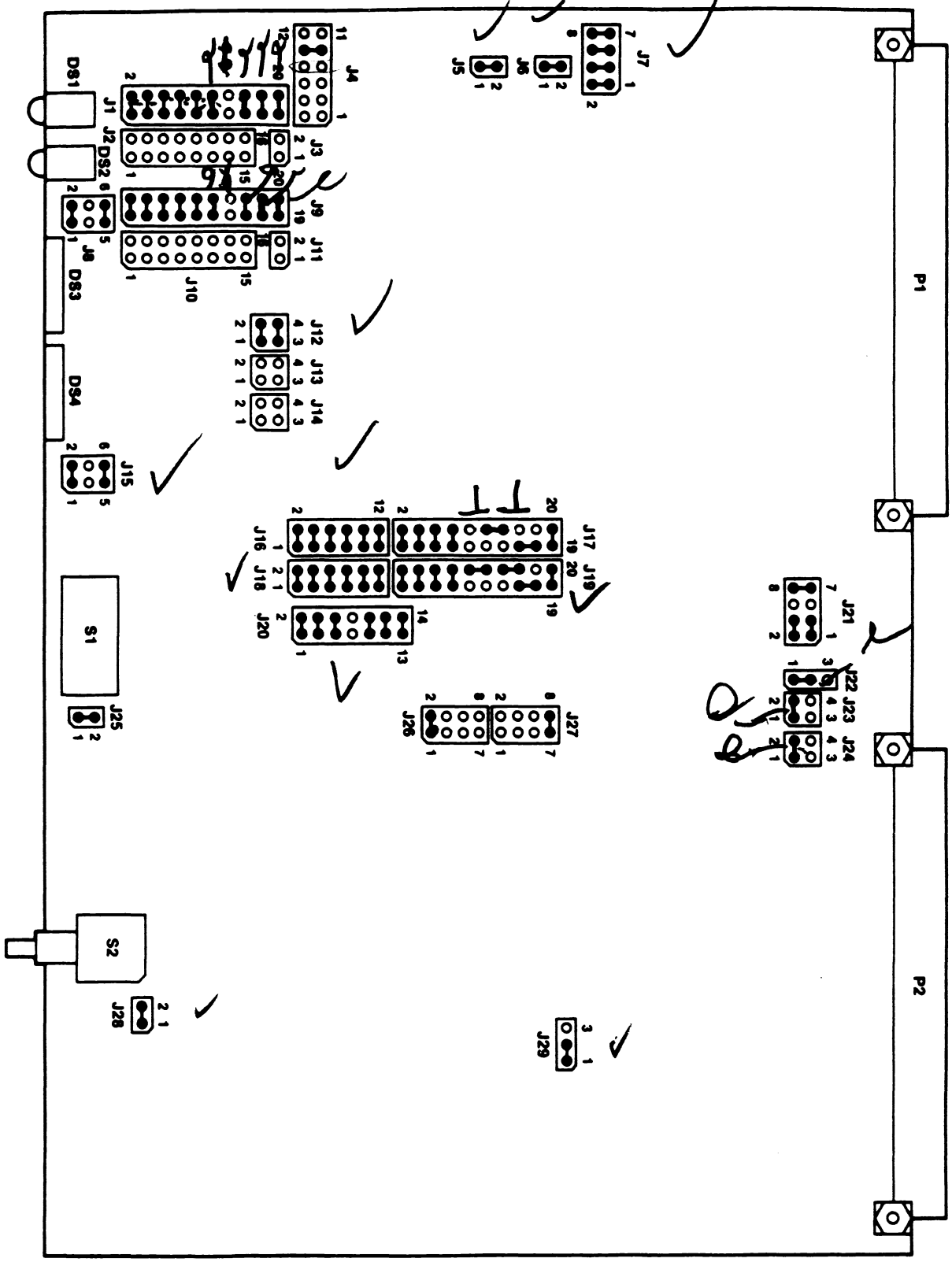


Figure 2-2. MVME050 Jumper Locations and Settings

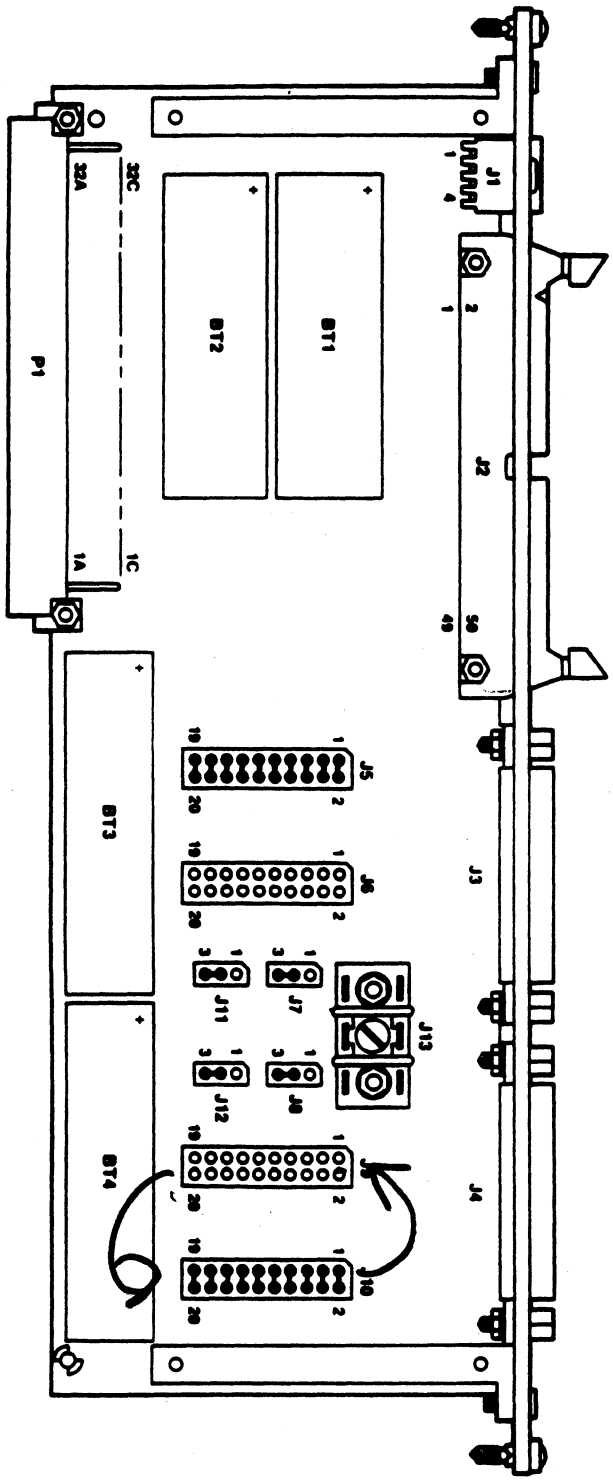


Figure 2-3. MVME701 Jumper Locations and Settings

2

2.4.4 MVME319 Intelligent Disk Controller. The MVME319 module is shipped with factory-installed jumpers ready for use as a single disk controller in a VME system. However, the jumper areas that determine the VMEbus requester priority level, the VMEbus interrupter priority level, the command channel base address, the VMEbus system fail output and the 5¼-inch diskette DRV3* line must be configured according to the actual system requirements.

Table 2-4 lists the function of each jumper and lists the factory configured setting, which also supports SYSTEM V/68. Figure 2-4 illustrates the jumper location and the setting.

The disk storage capacity can be expanded by adding one or more disk drives or disk controllers. An additional MVME319 disk controller improves disk data throughput significantly when the root filesystem and the user filesystems are served by separate MVME319 controllers. The VME short I/O address must be strapped to $FF0200 for the second MVME319 controller.

The disk and SCSI controller configuration can be changed by editing the file **/usr/include/sys/io/idio.h**. Parameter tables for different types of disk devices can be found in the file **/usr/include/sys/id.h**. Different Winchester devices can be supported by adding new parameter entries to this file, as long as the devices are compatible with the Adaptec and MVME319 controllers. SYSTEM V/68 must be relinked to support non-default configurations. Refer to the *MVME319 VMEbus Disk Controller User's Manual* and the *id*(7) entry in *SYSTEM V/68 Administrator's Manual* for further information.

Table 2-4. MVME319 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
K1, K2, K3	VMEbus Requester Priority Level (Level 3)	All 3 jumpers set at 1-2. (Factory Setting)
K6, K8	VMEbus Interrupter Priority Level (Level 3)	K-6 set 3-8 K-8 set 1-6 (Factory Setting)
K15	Command Channel Base Address	1-14, 2-13, 3-12, 4-11 5-10, 6-9, 7-8 (Factory Setting)
K7	System Fail Enabled	1-2 (Factory Setting)
K16	Drive Select DRV3* Enabled	1-2 (Factory Setting)
K10	Interrupt Controller Config.	No option
K9, K11	Local RAM Configuration	No option
K12, K13, K14	Local RAM Configuration	No option

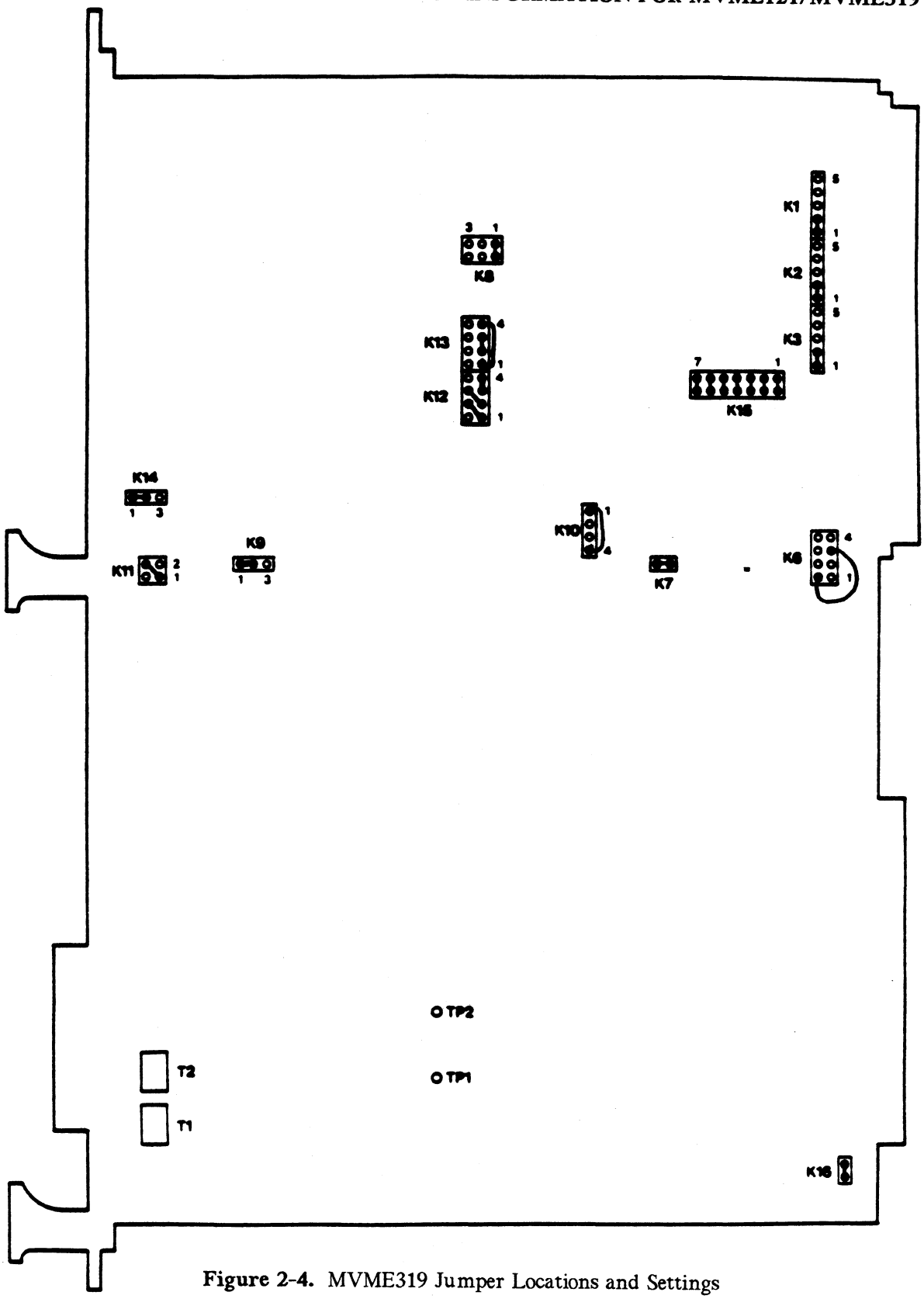


Figure 2-4. MVME319 Jumper Locations and Settings

2

2.4.5 Memory. An MVME121-based system, as configured, contains .5Mb of dynamic memory. If more memory is needed, additional MVME202 or MVME222 memory modules can be used. The MVME202 module must be strapped to provide contiguous addressing starting at hex address $80000. Refer to the *MVME202/MVME222-1/MVME222-2 512KB/1MB/2MB Dynamic Memory Module User's Manual* for further information.

2.4.6 Cache. An MVME121-based system has 4Kb of on-board cache. The cache can be jumpered to support four different cache configurations, as described in Table 2-5. A series of Benchmark tests were run to determine the optimal cache configuration for SYSTEM V/68. The test results indicate the operating system performs best when the cache is configured for mixed supervisor and user.

Table 2-5. Cache Configurations for MVME121-based System

CONFIGURATION	JUMPER J9	JUMPER J17
4Kb for user only	pins 2-3	pins 1-2
4Kb for supervisor only	pins 2-3	pins 2-3
4Kb mixed supervisor and user	pins 2-3	No Jumpers
2Kb supervisor, 2Kb user	pins 1-2	No Jumpers

On the MVME121 board, both Jumpers J9 and J17 have 3 pins associated with them. A rectangle with a notched corner has been drawn around the pins; Pin 1 is the pin closest to the notch. The recommended configuration for the MVME121 board (Table 2-1) includes the 2Kb supervisor, 2Kb user setting. The recommended configuration is also the default setting as shipped from the factory.

Cache on the MVME121 can be enabled/disabled in software. In some instances, it is desirable to disable cache regardless of the software. To disable the cache:

1. Connect J20 pins 1 and 2, using a shorting jumper.
2. Place S3 position 1 in the closed/on position.
3. Press RESET on the board.

Notice that when S3 is in the closed position, the MVME120bug performs auto-boot when power is first supplied to the system. If auto-boot is not desired, press the system RESET button about 5 seconds after power is supplied.

To remove the hardware disable of cache function:

1. Disconnect J20 pins 1 and 2. (Remove the shorting jumper.)
2. Press RESET on the board.

Notice that when J20 pins 1 and 2 are disconnected, S3 position does not affect the hardware disable of cache.

2.5 Boot Procedures

This section is provided as reference information. Do not boot the system without first reading Chapter 3 and installing the software according to the step-by-step installation plan.

2.5.1 Boot From the Winchester Drive. The following procedure boots the system after a routine shutdown, or after the system has crashed.

1. Power up the system. Press the RESET button on the MVME050 board to reset the system. The MVME120bug debug monitor displays the following question on the console:

   Size and initialize RAM (y/n) ?

   Answer with **y** .

2. The MVME120bug prompt appears. Wait a few seconds for the Winchester drive to spin up. The Winchester disk drive corresponds to Logical Unit 0. After the drive select light on the Winchester drive has turned off, it is necessary to wait at least 15 seconds before the boot command is entered. If the waiting period is neglected, SYSTEM V/68 may not load properly and errors may occur.

   Bootstrap the operating system from the Winchester disk with the MVME120bug boot command:

   bo 0,1

3. SYSTEM V/68 is loaded into memory and executed. The system reports the amount of available memory and comes up in "single user mode".

4. Enter multi-user mode immediately with the command:

   init 2

2.5.2 Boot From the Diskette Drive. This section is provided as reference information. Do not boot the system without first reading Chapter 3 and installing the software according to the step-by-step installation plan.

If you are unable to boot from the Winchester disk, the minimum system utility diskette supplied with this release can boot the system. Two "minload" diskettes labeled "1 of 4" and "2 of 4" are included in the Release 2, Version 1.1 distribution. Each minload diskette is designed to be used with a particular system: the diskette labeled "1 of 4" is the minload diskette for customers whose systems include an MVME319 disk controller.

DO NOT USE the diskette labeled "2 of 4"; it is the minload diskette for MVME121/MVME320 systems.

To boot your system from the minload diskette:

1. Remove the write protect label from the minload diskette labeled "1 of 4". Insert the diskette into the diskette drive. The diskette drive corresponds to the Logical Unit 6.

2. Bootstrap the operating system from the diskette with the MVME120bug boot command:

   bo 6,1

2.5.3 Boot Errors. If MVME120bug delivers an error message during the boot procedure, reset the system by pressing the MVME050 system controller RESET button. Enter the command

vi

Repeat the boot procedure. If the problem remains, power off the system and power it on again after a few seconds. Try the boot procedure again. If this does not help, try to boot from a different media. Do not try to boot from the "2 of 4" minload diskette.

2

2.6 Board-level Operations

This section provides information regarding special operation of MVME121-based system switches and LEDs when SYSTEM V/68 is installed. SYSTEM V/68-specific information with respect to the operation of the firmware-resident monitor/debugger MVME120bug is included in paragraph 2.7 and in the *MVME120bug Debug Monitor User's Manual.*

2.6.1 MVME121 Board. The MVME121 board contains the following pushbutton control switches:

RESET

When this momentary-action pushbutton switch is pressed, it resets the MVME121 logic circuits. If the MVME121 is in the SYSTEM V/68 operating system, MVME120bug is entered by pressing RESET.

ABORT

When this momentary-action pushbutton switch is pressed, the MVME121 enters MVME120bug, but the MVME121 logic circuits are not reset. After an abort, the user can enter **G** to continue execution of the current program prior to the abort.

NOTE: On the MVME120 family of processor boards, the level 7 interrupt that should result from pushing the ABORT button is not generated while a STOP instruction is being executed. That is, if the processor is executing a STOP instruction, the firmware debugger Software Abort function does not give control to the user until another interrupt occurs that ends the STOP instruction execution.

2.6.2 MVME050 Board. The MVME050 board contains a RESET pushbutton. When the RESET button is pressed, a system-level function is initiated that enables the RESET signal to generate the system reset, SYSRESET*. The SYSRESET* is sent to the other modules in the system through the VMEbus.

For further information about the operation of the MVME050 module, refer to the *MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual.*

2.7 MVME120bug Firmware Monitor Commands

MVME120bug is the firmware-resident monitor/debugger from which SYSTEM V/68 is booted. The MVME120bug prompt is

MVME120 1.*r* >

where *r* identifies the revision number of the debug monitor.

Twelve MVME120bug primitive commands should not be used with SYSTEM V/68. The commands can be used only in a stand-alone mode.

BI	Block initialize.
BT	Block test.
DU	Dump memory (S-records).
IOC	Issue disk I/O command.
IOP	Issue physical read/write.
IOT	Teach disk controller a configuration.
LO	Load (S-records).
PF	Port format.
ST	Self-test.
TM	Transparent mode.
VE	Verify (S-records).
VI	Vector initialize.

The following commands may be useful for debugging, but should be used only in single-user mode after *sync*(1) has executed. Use of these commands could result in the need for system reboot. More detailed information about the MVME120bug commands is provided in the *MVME120 Debug Monitor User's Manual.*

BD	Bootstrap dump.
BF	Block fill. (Note: MVME120bug distinguishes between uppercase and lowercase characters when filling a block.)
BH	Boot and halt.
BM	Block move.
BO	Boot operating system.
BR, NOBR	Set and remove breakpoints.
BS	Block search.
CACHE	Manipulate cache memory.
DC	Data conversion/evaluation.
DF	Display formatted registers.
GD	Go direct.
GO	Install breakpoints and go.
GT	Go until address.
HE	Display commands/registers.
MD	Memory display.
MM	Memory modify.
MP	Map memory for disk I/O.
MS	Memory set.
OF	Offset register display.
PA, NOPA	Printer attach/detach.
TA	Terminal attach.
TR	Trace.

2

TT	Trace until address.
.<register>	
	• .A0 - .A7
	• .D0 - .D7
	• .DFC
	• .PC
	• .R0 - .R6
	• .SFC
	• .SR
	• .SSP
	• .USP
	• .VBR
(BREAK)	Abort command or process.
(DEL)	Delete character.
(CTRL)D	Redisplay line.
(CTRL)H	Delete character.
(CTRL)S	Suspend output; any character continues.
(CTRL)X	Cancel command line.
(RETURN)	Process current/previous command line.

3. SOFTWARE INSTALLATION FOR MVME121/MVME319

This chapter contains step-by-step instructions for installing SYSTEM V/68 Release 2, Version 1.1 object and/or source code. Administrative procedures that are needed during the installation and are considered part of system administration (e.g., initializing media, changing system variables, generating a kernel) are also included.

3.1 Distribution Description

SYSTEM V/68 Release 2, Version 1.1 is available in both object and source distributions. Both object and source distributions use SYSTEM V/68 Release 2, Version 1 as their foundation. The Version 1.1 release is loaded in layers of code: the base is Release 2, Version 1 code; the Release 2, Version 1 Update is loaded next; and overlaying both is the Version 1.1 code. The Release 2, Version 1.1 distribution is composed of the following diskettes:

- Release 2, Version 1 object and/or source code
- Release 2, Version 1 Update object and/or source code
- Release 2, Version 1.1 object and/or source code

3.1.1 Release 2, Version 1.1 Object Distribution. The Release 2, Version 1 object distribution consists of 33 diskettes; the source distribution consists of 24 diskettes. The Release 2, Version 1 Update distribution varies in size with each update. The distribution is described in the customer letter that accompanies the diskette(s). The Release 2, Version 1.1 object distribution consists of 4 diskettes.

Not all diskettes from these three sets are loaded to form the complete Release 2, Version 1.1 object distribution. Several diskettes will damage the system if they are loaded. At this point, clearly mark the following diskettes, or set them aside, to ensure that they will not be loaded during the installation procedure.

CAUTION

DO NOT LOAD THE FOLLOWING DISKETTES AS PART OF THE RELEASE 2, VERSION 1.1 DISTRIBUTION. THESE DISKETTES CONTAIN KERNELS AND OTHER FILES THAT ARE NOT COMPATIBLE WITH THE MVME121/MVME320 SYSTEM. IF THE DISKETTES ARE LOADED, THE SYSTEM WILL NOT BOOT.

DO NOT LOAD THESE DISKETTES:		
Set	**Diskette Number**	**Description**
Release 2, Version 1	***82V1XBSVB1**	**VME/10 stand-alone boot**
	"0 of 31"	**VME/10 minload**
	"5 of 31"	**VME/10 files**
	"24 of 31"	**VM03 files**
	"25 of 31"	**VM03 files**
Release 2, Version 1.1	**"2 of 4"**	**MVME320 files**

The complete SYSTEM V/68 Release 2, Version 1.1 object distribution is listed in Table 3-1. The table lists each diskette, its *cpio* block count and a description of the files contained on the diskette. Notice that the Release 2, Version 1 Update diskette(s) block count is not included in the table. Because the update may change quarterly, the block counts for the Update are included in the *SYSTEM V/68 Update Customer Letter*.

Table 3-1. Release 2, Version 1.1 Object Diskettes: Description and Block Counts

Set	Diskette Number	Cpio Block Count	Description
RELEASE 2, VERSION 1	1 of 31	1159	/bin
	2 of 31	1092	/bin
	3 of 31	1160	/etc
	4 of 31	981	/etc, /usr subdirectories
	5 of 31		**DO NOT USE.**
	6 of 31	1139	/lib
	7 of 31	1152	/usr/bin
	8 of 31	1151	/usr/bin
	9 of 31	1121	/usr/bin
	10 of 31	1149	/usr/bin
	11 of 31	1141	/usr/bin, /usr/lib
	12 of 31	1109	/usr/lib
	13 of 31	1114	/usr/lib
	14 of 31	620	Includes
	15 of 31	685	Accounting
	16 of 31	1097	Text processing
	17 of 31	959	Text processing
	18 of 31	781	Games
	19 of 31	1082	Graphics
	20 of 31	1067	Graphics
	21 of 31	816	Graphics
	22 of 31	736	Uucp
	23 of 31	553	Kernel build area
	24 of 31		**DO NOT USE.**
	25 of 31		**DO NOT USE.**
	26 of 31	614	Terminfo
	27 of 31	1129	On-line documentation
	28 of 31	838	On-line documentation
	29 of 31	901	On-line documentation
	30 of 31	417	On-line documentation
	31 of 31	619	Miscellaneous utilities
RELEASE 2, VERSION 1 UPDATE			**Refer to Update Customer Letter.**
RELEASE 2, VERSION 1.1	1 of 4	1145	MVME121/319 Minload
	2 of 4		**DO NOT USE.**
	3 of 4	1008	Updated object
	4 of 4	258	Kernel build area On-line documentation

3.1.2 Release 2, Version 1.1 Source Distribution. Source distribution for Release 2, Version 1.1 includes the source code for Release 2, Version 1; the Release 2, Version 1 Update for source; and the Release 2, Version 1.1 source code. Like the object code, source code is installed in layers. Table 3-2 lists the source diskettes and their *cpio* block counts. To install the source code, users must first complete the object code installation. The source installation begins as Step 11 in the installation procedure.

Table 3-2. Release 2, Version 1.1 Source Diskettes and Block Counts

Set	Diskette Number	Cpio Block Count
RELEASE 2, Version 1	1 of 24	845
	2 of 24	990
	3 of 24	891
	4 of 24	547
	5 of 24	638
	6 of 24	1038
	7 of 24	939
	8 of 24	938
	9 of 24	1025
	10 of 24	728
	11 of 24	834
	12 of 24	985
	13 of 24	946
	14 of 24	1027
	15 of 24	1113
	16 of 24	1110
	17 of 24	1048
	18 of 24	1109
	19 of 24	981
	20 of 24	997
	21 of 24	1012
	22 of 24	1053
	23 of 24	1087
	24 of 24	805
RELEASE 2, VERSION 1 UPDATE		**Refer to Update Customer Letter**
RELEASE 2, VERSION 1.1	1 of 1	862

3.2 Object and Source Installation Instructions

The procedure to install the SYSTEM V/68 object distribution is described in this paragraph as a series of ten steps. Source distribution installation follows the object installation, and continues with 3 additional steps, 11 through 13.

During the installation, users may want to refer to additional information that is provided toward the end of this chapter. References to the information are located in the steps where the information may be needed.

STEP 1: DISK SPACE CONSIDERATIONS

You should have labeled six diskettes from the object distribution **DO NOT USE**, or have set them aside to ensure they will not be loaded. (Refer to paragraph 3.1.1.) By not loading these diskettes, administrators gain approximately 1500 blocks of extra file space in their systems.

After considering the number of users and the type of processing anticipated, administrators of 40Mb systems may wish to omit some of the SYSTEM V/68 software functions to allow more space for user files. The following table lists the block counts for some software functions that can be omitted easily.

Function	Diskette Number	Block Count
Accounting	Diskette 15	685
Games	Diskette 18	781
Graphics	Diskettes 19, 20, 21	2965
Text Processing	Diskettes 16, 17	2056
On-line Documentation	Diskettes 27-30	2280

If any diskettes are to be omitted from the loading procedure, label them now and set them aside.

STEP 2: POWER UP AND BOOT FROM DISKETTE

a. Power up the system.

b. Respond to the MVME120bug prompt "Size and Initialize RAM (Y/N)?" with **y** .

c. REMOVE THE WRITE PROTECT TAPE from the minload diskette labeled "1 of 4" and insert the diskette into the device drive #6. Boot from the diskette with the command:

bo 6,1

The following messages appear on the terminal screen:

M68010/sysV68: unixR2V1.1
real memory = *n*
available memory = *n*

INIT: SINGLE USER MODE

STEP 3: FORMAT DISK AND LOAD MINLOAD ONTO FIXED DISK

a. When the SYSTEM V/68 prompt # appears, enter the command:

/etc/loadtools/minload

The screen displays a script that formats the Winchester disk, and loads boot information and a minimal set of utilities. The script asks whether the user wants to reformat the Winchester disk, and prompts for information about the disk.

To answer the prompts from the system, you must know the manufacturer of your disk, either Micropolis or Vertex, and the location of any disk defects. The necessary information is provided in the Winchester verification report that is packed with the disk drive. Paragraph 3.3.1 contains instructions for entering bad block locations from the information included in your verification report. (Refer also to *badlist*(4) in Chapter 8 of this manual.)

(Instructions for formatting a Winchester disk without loading the minimum system utilities, i.e., when formatting a second disk, are included in paragraph 3.3.1.)

b. Format the disk, following the prompts and directions that appear on the screen as part of the **minload** routine.

c. After the formatting routine, the system begins to copy the minimum system utilities onto the disk. Approximately 50 filenames are listed on the screen as they are copied. The screen should indicate that 1145 blocks have been copied. If some other number of blocks appears, repeat this step.

STEP 4: BOOT FROM FIXED DISK

When the lights on both drives are no longer illuminated, press the RESET button on the MVME050 module. In response to the MVME120bug prompt, enter the boot command:

bo 0,1

The command boots the MVME121-based system from the fixed disk.

STEP 5: BACKUP OBJECT DISKETTES

To backup the object distribution diskettes, you need to format at least 32 diskettes with the *dinit*(1M) program:

28 diskettes for the SYSTEM V/68 Release 2, Version 1 object distribution,
(Diskettes 0, 5, 24 and 25 are not backed-up since they are not loaded.)
1 or more diskettes for the Release 2, Version 1 Update, and
3 diskettes for the Release 2, Version 1.1 distribution.
(Backup the minload diskette and the 2 object code diskettes.)

a. The procedure to format diskettes for Release 2, Version 1.1 is described in paragraph 3.3.2 of this manual. Format as many diskettes as you need to backup the distribution. (Users who will be installing source distribution may wish to backup the source distribution diskettes now.)

b. To backup the distribution, the master object distribution diskette is first copied onto the fixed disk. Then, a copy is made from the fixed disk onto a newly formatted, backup diskette. To do this, first insert an object distribution diskette into drive #6.

c. Enter the command:

dd if=/dev/rdsk/id_6s7 of=/flop bs=16k

to copy the master diskette to the Winchester.

d. Remove the master diskette and insert a newly formatted diskette into the drive.

e. Enter the command:

dd if=/flop of=/dev/rdsk/id_6s7 bs=16k

to copy from the Winchester to the new diskette.

f. Repeat these steps for each of the distribution diskettes.

STEP 6: CREATE AND MOUNT usr FILESYSTEM

The system administrator may want to modify the recommended allocation of space for object files. The recommended procedure, as outlined in this section, places the **usr** filesystem in partition 1. Partitions 5 and 6 are reserved for source distribution. ("Object only" customers may use these partitions however they deem most useful.)

To create and mount the **usr** filesystem in the proper partitions, enter the commands:

```
mkfs /dev/dsk/id_0s1   28032:3200
labelit /dev/rdsk/id_0s1    usr  R2V1.1
mount /dev/dsk/id_0s1   /usr
```

STEP 7: LOAD OBJECT DISKETTES

Before you insert a diskette into the drive or enter any of the commands in this step, read through the entire step to be sure you understand the complete procedure.

In addition, take the following precautions:

- MAKE SURE YOU LOAD ONLY THE CORRECT DISKETTES.

  Double-check that you have marked five diskettes from Release 2, Version 1 and one diskette from Release 2, Version 1.1 "DO NOT LOAD". Refer to paragraph 3.1.1.

- DO NOT REMOVE THE WRITE PROTECT TAPE FROM ANY DISKETTE.

Briefly, users must load the diskettes in the following sequence:

a. Insert the first Release 2, Version 1 object diskette.

b. Enter the appropriate load command. Load all Release 2, Version 1 diskettes as prompted by the system.

c. As each diskette is loaded, the number blocks transferred and any load diagnostics are saved in a message file. Read the message file to check that all diskettes loaded correctly. Remove the message file. If necessary, reload any miscopied Release 2, Version 1 diskettes. Check the message file to verify the second load. Remove the message file.

d. Insert the first Release 2, Version 1 Update diskette.

e. Follow the loading instructions contained in the Update Customer Letter.

f. Insert the first Release 2, Version 1.1 object diskette.

g. Enter the appropriate load command. Load the second object diskette as prompted by the system.

h. Read the message file to check that all diskettes loaded correctly. Remove the message file. If necessary, reload any miscopied diskettes. Check the message file to verify the second load. Remove the message file.

The initial loading of the complete object distribution requires approximately three hours. However, the process can be interrupted at any time. The procedure for interrupting and re-initiating the load follows the loading instructions, near the end of this step.

Notice there are two possible load commands:

/etc/loadtools/readflops.obj 2>> /message
or
/etc/loadtools/readflops 2>> /message

(There is no space in the expression **2>>**)

CAUTION
The readflops.obj COMMAND DESTROYS DISK PARTITION 5. IF ANY FILES HAVE BEEN INSTALLED ON PARTITION 5, USE THE readflops COMMAND INSTEAD.

The **readflops.obj** command uses the *dd*(1) utility to copy the contents of the diskette into partition 5. The *cpio*(1) utility copies the files from partition 5 into their proper place. **Readflops.obj** is recommended because it is significantly faster than **readflops**. However, any information contained in partition 5 is overwritten and lost when **readflops.obj** is executed.

The second script, **readflops**, also located in **/etc/loadtools**, loads diskettes without destroying partition 5.

LOADING INSTRUCTIONS

a. Re-read the precautions listed at the beginning of this step.

b. Insert the first Release 2, Version 1 object diskette, "1 of 31", into drive #6.

c. Enter the appropriate load command:

/etc/loadtools/readflops.obj 2>> /message
or
/etc/loadtools/readflops 2>> /message

(There is no space in the expression **2>>**)

d. For each diskette that is loaded (using either the **readflops.obj** or **readflops** script), the number of blocks transferred is contained in the file **/message**. If load diagnostics are encountered, they are listed above the block count for the miscopied diskette. The first block count given in **/message** refers to diskette "1 of 31".

e. Load the 28 Release 2, Version 1 diskettes as prompted by the system.

f. Read the **/message** file to check the diskettes loaded correctly. Use the *ed*(1) utility to read the **/message** file. Enter the commands:

ed /message
1,10p

to print the first 10 lines in the file. If load diagnostics such as **"Cannot read** *filename***"** exist in the **/message** file, or the block count does not match the count provided in Table 3.1, the diskette was miscopied and must be loaded again.

To read the next 10 lines in the file **/message**, enter the commands:

10,20p

Continue reading the file, checking the block counts and looking for load diagnostics, until all diskettes have been accounted for. Make a note of each diskette that must be reloaded. Typically, **/message** contains about 80 lines for the Release 2, Version 1 distribution.

g. After the entire **/message** file has been examined to determine which diskettes must be reloaded, exit the editor and remove the **/message** file with the commands:

q
rm -f /message

h. Reload any miscopied Release 2, Version 1 diskettes. Place the miscopied diskette in drive #6 and enter the same load command you entered earlier. The diskettes need not be loaded in sequential order.

i. Examine the new file **/message** to ensure the diskette(s) loaded successfully. If the second load is unsuccessful, assume that the media itself is bad. Call Motorola/Four Phase Customer Support Organization to obtain new media.

j. After you have verified that all Release 2, Version 1 diskettes have been loaded properly, insert the first Release 2, Version 1 Update diskette. Instructions for installing the Update are contained in the Update Customer Letter. Follow the procedure described in the letter to verify that the Update has been loaded correctly.

k. Insert the first Release 2, Version 1.1 object diskette, "3 of 4", in drive #6. Enter the appropriate load command. Load "4 of 4" when the system prompts.

l. Verify the load by reading the **/message** file. Reload any miscopied diskettes following the procedures that were described for Release 2, Version 1 object diskettes.

INTERRUPTING AND RE-INITIATING THE LOAD

The loading process can be interrupted at any time. Finish copying the diskette you are loading. When prompted by the system for the next diskette, quit the load. Then, flush unwritten buffers out to the disk with the *sync*(1) command. The quit and "sync" is executed by entering the commands:

q
sync
sync
sync

You can now leave the system on, and return to the loading sequence later. To re-enter the loading sequence, insert the next diskette into disk drive #6 and enter the appropriate load command.

If you plan to power down the system, recall that the **/usr** filesystem is still mounted. Before you shut down, you must unmount the filesystem and clear out the buffers again. Remember that after you execute the third **sync** command in a series, you must wait until the disk drive light is no longer illuminated before you enter the next command. The complete procedure to discontinue the load and power down the system is as follows:

```
q
sync
sync
sync          (Wait for the drive light to turn off.)
umount    /dev/dsk/id_0s1
sync
sync
sync          (Wait for the drive light to turn off.)
(Press the RESET button on the MVME050 module.)
(Power off.)
```

When you are ready to power up the system and begin the loading sequence again, remember to mount the **/usr** filesystem before you re-enter the **readflops.obj** (or **readflops**) command, if the filesystem was unmounted earlier. Enter the commands:

```
(Power up.)
bo
mount    /dev/dsk/id_0s1    /usr
```

then insert the next diskette and re-enter the appropriate load command.

STEP 8: EDIT THE /etc/checklist AND /etc/rc FILES

Both the **/etc/checklist** and **/etc/rc** files must be edited.

a. Edit the file **/etc/rc** with the *ed*(1) utility. Enter the *ed* editor and search for the correct line with the commands:

```
ed /etc/rc
/mount/
```

b. If the first character is a # , remove the character to "uncomment" the line with the command:

```
s/^. //p
```

c. Next, change the line with the command:

```
s/cntrl_XsX/id_0s1/p
```

d. Write the change and quit the editor with the commands:

```
w
q
```

e. Append the device names to the file **/etc/checklist** as follows:

```
echo /dev/dsk/id_0s0 >> /etc/checklist
echo /dev/rdsk/id_0s1 >> /etc/checklist
```

(**Note:** The root filesystem should not be checked using the raw device designation. For a more detailed explanation, refer to *fsck*(1M).)

f. Save these changes and update the fixed disk by executing the **sync** command three times. Enter:

sync
sync
sync

STEP 9: UPDATE KERNEL LIBRARIES

This step updates the kernel library files contained on the Release 2, Version 1 object diskettes. Kernels included on the Release 2, Version 1.1 diskettes are already upgraded. However, if a user intends to generate a custom kernel, the Release 2, Version 1 kernel libraries are needed, and therefore, must be updated.

The Release 2, Version 1 Update contains two scripts, **installio** and **installos**, which archive new object files into the kernel libraries. The Release 2, Version 1.1 kernel object files have been installed so that they will be included when the scripts are executed. For further information about the scripts, refer to the README file included on the Release 2, Version 1 Update. The scripts assume that the libraries, **lib1** and **lib2**, exist in the **/usr/src/uts/m68k/M68010** directory.

a. To update the kernel libraries, change directories to access the scripts:

cd /usr/src/uts/m68k/M68010

b. Execute the **installio** script with the command:

./installio

The script archives the new object modules into the kernel library, **lib2**, and places the old modules in the **/usr/src/uts/m68k/io** directory as *filename*.**o.old**. The script attempts to find two old object modules that correspond to two new modules, **lp0** and **lp050**. However, there are no such modules and two error messages will display:

ar: lp0.o not found
mv: cannot access lp0.o

ar: lp050.o not found
mv: cannot access lp050.o

Ignore these messages.

c. Execute the **installos** script with the command:

./installos

The script archives the new object modules into the kernel library, **lib1**, and places the old modules in the **/usr/src/uts/m68k/os** directory as *filename*.**o.old**.

The kernel libraries are now updated with all changes applicable to Release 2, Version 1.1.

STEP 10: PERFORM ADMINISTRATIVE TASKS

Users who are planning to add the source distribution to their system should skip this step and proceed to Step 11. Return to this step when Step 13 is completed.

The system administrator who is installing object distribution only is advised to enter multi-user mode now with the command:

init 2

When the system prompts for "Console login", enter **root**. The **root** login ensures that you have permission to perform the following administrative duties. Take time now to perform the following tasks:

- Assign a password for **root**. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Create user ids and home directories. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Check that the environmental variables for the file **/etc/profile** are correct. On target systems, link **/etc/prfile.VME120** to **/etc/profile.** (The new values will not take effect until the system is rebooted.) Refer to paragraph 3.3.9 of this manual.
- Create and expand the **lost+found** directories in each mountable filesystem. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Change the nodename variable if the system is to be connected to a *uucp*(1C) network. Refer to the UNIX-to-UNIX CoPy (uucp) Tutorial included in the *SYSTEM V/68 User's Guide.*
- Create the *sxt*(7) devices to implement user job control capability. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Edit the file **/etc/motd** to reflect that you are running Release 2, Version 1.1 (R2V1.1), not Release 2, Version 1 (R2V1). Refer to the *SYSTEM V/68 Administrator's Guide.*

STEP 11: SOURCE INSTALLATION

This installation procedure for source distribution assumes that the object distribution has been loaded according to Steps 1 through 9.

Partitions 5 and 6 have been reserved for the **/usr/src**. After the object files have been loaded, the first step in loading the source code is to create and mount the **usrsrc** filesystem on partitions 5 and 6. The commands that follow assume the **/usr** filesystem is mounted. If you have unmounted the filesystem, mount it with the command:

mount /dev/dsk/id_0s1 /usr

a. After you have verified that the **/usr** filesystem is mounted, enter the commands:

mkfs /dev/dsk/id_0s5 30696:5600
labelit /dev/rdsk/id_0s5 usrsrc R2V1.1
mount /dev/dsk/id_0s5 /usr/src

b. Backup all source diskettes onto newly formatted diskettes.

This part of the source installation is similar to Step 5 of the object installation. The source distribution contains 24 diskettes from Release 2, Version 1; at least one Release 2, Version 1 Update source diskette, and one Release 2, Version 1.1 source diskette. Format

one backup diskette for each source diskette. Use the *dinit* format utility described in paragraph 3.3.2. Follow the commands presented in Step 5 to backup the source diskettes onto the newly formatted diskettes.

c. Insert the first source diskette from the Release 2, Version 1 distribution, "1 of 24", into drive #6. Copy the SYSTEM V/68 Release 2, Version 1 source diskette to the fixed disk with the command:

/etc/loadtools/readflops /usr/src 2≫ /message

Notice that the command for copying the 24 Release 2, Version 1 source diskettes specifies **/usr/src** as the destination directory for the source code.

Continue loading the Release 2, Version 1 source diskettes as prompted by the system.

d. For each diskette that is loaded, the number of blocks transferred is contained in the file **/message**. If load diagnostics are encountered, they are listed above the block count for the miscopied diskette. The first block count given in **/message** refers to diskette "1 of 24". Read the **/message** file, following the procedure described in Step 7. Table 3-2 provides the block counts for the source diskettes.

Remove the **/message** file. If any diskette did not copy properly, reload it now.

e. Load the Release 2, Version 1 Update source diskette(s). The command that loads the Update diskette(s) and the Release 2, Version 1.1 source diskette is different from the command that copies the Version 1 source diskettes — it does not specify a destination directory:

/etc/loadtools/readflops 2≫ /message

Check the **/message** file against the block counts included in the Update Customer Letter to ensure the update diskette(s) loaded properly. The remaining diskettes must be loaded in order: Update source first, then Release 2, Version 1.1 source. Remove the **/message** file. If an update diskette did not copy properly, reload it now. The update source diskette(s) must be properly loaded before loading the Release 2, Version 1.1 source.

f. Load the Release 2, Version 1.1 source diskette; compare the block count in the **message** file against that in Table 3-2; remove the **/message** file.

If any diskette fails to load properly after two attempts, call Motorola/Four Phase Customer Support Organization to obtain new media.

STEP 12: UPDATE KERNEL LIBRARIES

Because the source installation mounted the **/usr/src** filesystem on top of the **/usr** filesystem, the library update performed in Step 9 is no longer accessible. Since custom kernels may be needed in the future, the kernel libraries should be updated at this point.

The kernel libraries can be updated in either one of two ways: either move the libraries generated in Step 9 to make them accessible; or generate new libraries from the source. Creating libraries from source will take several hours.

METHOD 1: MOVE ALREADY GENERATED LIBRARIES

a. Unmount the **/usr/src** filesystem to gain access to the libraries, which are located in the **/usr** filesystem.

```
cd /
umount /dev/dsk/id_0s5
```

b. Move the libraries to a temporary, and accessible, location: the root directory.

```
mv /usr/src/uts/m68k/M68010/lib*
```

(The **lib*** moves all files that begin with the letters "lib".)

c. Mount the **/usr/src** filesystem and install the libraries.

```
mount /dev/dsk/id_0s5  /usr/src
mv  /lib*  /usr/src/uts/m68k/M68010
```

METHOD 2: CREATE LIBRARIES FROM SOURCE This method requires several hours.

a. Change directories to access the makefile:

```
cd /usr/src/uts/m68k
```

b. Execute the makefile, specifying the target that will remake the libraries:

```
make lib010
```

When the compilations and archiving have completed, the kernel libraries will be upgraded and may be used to generate new kernels. (Refer to paragraph 3.3.10 for further information about generating a kernel.)

STEP 13: EDIT THE /etc/checklist AND /etc/rc FILES
(For Source Distribution)

a. Edit the file **/etc/rc** with the *ed*(1) utility:

```
ed /etc/rc
/mount/
```

b. Add a line of code to mount the **/usr/src** filesystem when the system is brought up in multi-user mode:

```
a (CR)
mount  /dev/dsk/id_0s5     /usr/src (CTRL)I #mount /usr/src filesystem
.
```

where (CTRL)I generates a tab character.

c. Write the change and quit editor with the commands:

```
w
q
```

d. Append the new device name to the file **/etc/checklist**:

```
echo /dev/rdsk/id_0s5 >> /etc/checklist
```

e. Save all changes and update the fixed disk by entering the **sync** command three times. Enter:

sync
sync
sync

Now return to Step 10 and perform the recommended administrative tasks to complete the installation.

3.3 Administrative Procedures

The following paragraphs describe some of the procedures administrators will perform during the course of their work.

3.3.1 Initialize Winchester Disk. This section describes the initialization procedure for 40Mb Winchester disks.

The system should be in single-user mode when initializing media. The MVME319 disk controller remains "busy" until the media has been formatted completely. For Winchester disks, formatting requires about 10 minutes. Only the superuser has permission to format Winchester media.

Before initializing the disk, you must know the manufacturer of the disk included in your system (either Micropolis or Vertex), as well as the location of any imperfections on the disk. The required information can be found on the Winchester verification report included with the disk drive. You cannot initialize the Micropolis Winchester media without this information. The Vertex disk cannot be formatted with the procedure that follows. There is no entry in the **/etc/diskdefs** file to support Vertex 40Mb Winchester disk drives.

When you are ready to format and initialize the Micropolis Winchester disk, follow these steps:

1. Power up the system.
2. The MVME120bug performs an automatic self-test. It tests the processor and on-board RAM and ROM memory. If any test fails, an error message is displayed. Next, the MVME120bug prompt displays the question: "Size and Initialize RAM (Y/N) ?" Enter **y** to initialize the system.
3. The system displays the message:

   COLD Start
   MVME120 1.*r*

   where *r* is the revision number of the debug monitor.

   REMOVE THE WRITE PROTECT TAPE from the minload diskette labeled "1 of 4" and place it in the device number 6 drive. Boot from the diskette with the command:

   bo 6,1

4. Wait for the SYSTEM V/68 prompt **#** to appear. Enter the command:

   dinit -f -o -b /etc/loadtools/vmeboot idwm40 /dev/rdsk/id_xs7

   The **idwm40** is specific to the Micropolis Winchester drive.

   The *dinit*(1M) utility invokes *idfmt*(1M), a disk formatter for MVME319 devices. *Dinit* enters an interactive mode and prompts the user for bad track entries. Check the

Winchester verification report supplied by the disk manufacturer for a list of bad blocks or imperfections on the disk. If no bad blocks are listed, type a period (.) to terminate the bad track handling phase of disk initialization. The device is assumed to be perfect.

Enter each imperfection listed on the verification report when *idfmt*(1M) prompts. The media defects are identified in the report by cylinder number, head number and byte offset.

Enter each defect by location, following the format:

cylinder head byteoffset

where the *byte* and *offset* numbers are entered as a single 8-character field. There is no space between *byte* and *offset*. (For further information, refer to *badlist*(4), in Chapter 8 of this manual.)

Follow each entry with a carriage return (CR). To end this phase of initializing the disk, type a period followed by a carriage return.

After *idfmt* has formatted the device, *dinit* sets up the volume-id and the configuration sectors, records the bad track information, and installs the boot loader on the drive.

(**NOTE:** If a Winchester disk is formatted without making the required bad track entries, proper operation cannot be guaranteed.)

3.3.2 Initialize Diskettes. Only two diskette types can be formatted: a 5¼-inch double-sided, double density diskette, or an 8-inch double-sided, single density diskette.

The system should be in single-user mode when initializing media. Diskettes are formatted track by track, and the controller is "busy" until one track has been formatted. Although diskettes can be formatted in multi-user mode, system performance decreases significantly. If the system administrator decides to grant users permission to format diskettes, the change may be made when the *idfmt*(1M) execute permissions are set.

Support for the 8-inch diskette is not provided with the default system configuration. To add support, edit the files **/usr/include/sys/io/idio.h** and **/usr/include/sys/id.h**, and relink an appropriate kernel. Refer to the *MVME319 Intelligent Disk Controller User's Manual* for the requirements of an 8-inch diskette drive.

To format a 5¼-inch diskette, remove the write-protect tab from the diskette and insert the diskette into the drive. Enter the command:

dinit —f —o iddsdd5 /dev/rdsk/id_*x*s7

where *x* specifies the correct drive.

To make a filesystem on the diskette, invoke *mkfs*(1M) on the diskette's block device entry, specifying either 1264 or 1276 as the blocks argument:

SLICE	BLOCKS ARGUMENT	DESCRIPTION
0	1264	Identical to slice 2 on VM22, slice 0 on VME/10, and slice 0 on an MVME320.
7	1276	Entire diskette contents

To format an 8-inch diskette, install a write-enable tab, if applicable, and insert the diskette into the drive. Enter the command:

dinit —f —o iddssd8 /dev/rdsk/id_*x*s7

where *x* specifies the correct drive.

To make a filesystem on the diskette, invoke *mkfs*(1M) on the diskette's block device entry, specifying either 988 or 1001 as the blocks argument:

SLICE	BLOCKS ARGUMENT	DESCRIPTION
0	988	Identical to slice 0 on a VM21 or VM22.
7	1001	Entire disk contents.

A bootable diskette may be created by adding the **—b** option and specifying the bootloader, **/stand/m68k/boots/vmeboot**. When the **—b** option is invoked, a filesystem must be created on slice 0.

3.3.3 Initialize Floppy Tape. The system should be in single-user mode when initializing media. The MVME319 disk controller remains "busy" until the media has been formatted completely. The formatting procedure for tape requires about 30 minutes. Only the superuser has permission to format tape media.

Only one type of tape cartridge can be formatted, SCOTCH DC600A. To do so, insert a writable tape cartridge into the tape drive.

Wait until the tape is completely rewound. Enter the command:

dinit —f —o idftape /dev/rdsk/id_4s7

3.3.4 Media Backup Considerations. The MVME319 disk controller can be used to interface with a CIPHER DATA PRODUCT CT525 FloppyTape. The tape drive uses standard 1/4-inch data cartridges (3M DC600A or equivalent) and has a usable capacity of 24Mb. The data organization on the tape is similar to that on a diskette: single or contiguous data blocks can be addressed in a quasi start/stop mode. The tape is served by the *id*(7) disk driver as if it was a disk drive. It is recommended that the tape be used in the raw I/O mode whenever possible, and that read or write requests only be for very large chunks of data, i.e., 16Kb or 32Kb blocks. Only requests for 16Kb or 32Kb blocks keep the tape streaming and avoid time-consuming repositioning cycles.

The tape is most efficient as a backup medium when it is used with the *dd*(1) utility to copy physical filesystems. The blocksize (**bs** option) should be set to 16Kb. The count option specifies the number of 16Kb blocks to be copied to the tape. Note that the count option must be used with the *dd* utility.

3.3.5 Copy a Filesystem to Floppy Tape. The following example copies filesystem **/usr** mounted on **/dev/dsk/id_0s1** to floppy tape.

1. Check the size of the filesystem by using *df*(1M):

   df -t /dev/dsk/id_0s1

   The *df* utility displays the size of the filesystem and the free space. Calculate the number of 16Kb blocks needed to backup the filesystem. (Round the number to the next greater whole number.) This number is entered in the **dd** command as the count.

If the filesystem is not mounted, **labelit** can provide information about the filesystem size. (Refer to *volcopy*(1M) for further information about **labelit**.)

2. If the filesystem is mounted, unmount with the command:

   umount /dev/dsk/id_0s1

3. Insert a formatted and initialized tape cartridge into the tape drive. The tape rewinds automatically.

4. When the tape stops, invoke the **rewind** utility (*fseek*(3S)) to indicate that the tape medium has been changed:

   rewind /dev/rdsk/id_4s7

   (**Note:** Invoke the rewind utility whenever when a tape cartridge is replaced. If this is neglected, proper operation can not be guaranteed.)

5. Start the backup:

   dd if=/dev/rdsk/id_0s1 of=/dev/rdsk/id_4s0 bs=16k count=*nnn*

6. Remove the tape cartridge and label it carefully.

3.3.6 Restore the Backup From Floppy Tape. To restore the backup:

1. Insert the backup tape cartridge into the tape drive and wait until it is completely rewound.

2. Invoke the **rewind** utility (*fseek*(3S)) to indicate that the tape medium has been changed:

   rewind /dev/rdsk/id_4s7

   (**Note:** Invoke the rewind utility whenever a tape cartridge is replaced. If this is neglected, proper operation can not be guaranteed.)

3. Check that the destination filesystem is not mounted.

4. Copy the tape back to disk with the command:

   dd if=/dev/rdsk/id_4s0 of=/dev/rdsk/id_0s1 bs=16k count=*nnn*

   When data is lost, it may not be necessary to copy the entire tape. If only a few files are affected, the tape can be mounted like a disk to recover the lost data. However, bear in mind that searching for files on tape takes much longer than searching on a disk device.

   The backup procedure should be executed only by an experienced system administrator. The system should be in single-user mode when the backup is performed.

   If the tape drive is operating with new tape cartridges, the read/write head should be cleaned after two hours use. If the tapes are used, the drive should be cleaned after eight hours. A lintless swab moistened with Freon Degreaser Type TF (Trichlorotrifluoroethane) should be used to clean the read/write head and capstan roller.

   Tape cartridges which are not in use should be removed from the tape drive. When cartridges are left on the drive, read/write errors can result. If errors develop on a tape, the cartridge must be reformatted to lock out the bad spots.

3.3.7 Physical Backup of 40Mb Winchester Disk. A complete 40Mb Winchester disk can be copied to tape using two cartridges.

To write to the first cartridge, use the following commands:

```
rewind /dev/rdsk/id_4s7
dd if=/dev/rdsk/id_0s7 of=/dev/rdsk/id_4s0 bs=16k count=1460
```

To write to the second cartridge, use the following commands:

```
rewind /dev/rdsk/id_4s7
dd if=/dev/rdsk/id_0s7 of=/dev/rdsk/id_4s0 bs=16k skip=1460 count=963
```

To restore the backup, insert the first cartridge and enter the commands:

```
rewind /dev/rdsk/id_4s7
dd if=/dev/rdsk/id_4s0 of=/dev/rdsk/id_0s7 bs=16k count=1460
```

Remove the first cartridge. Insert the second cartridge and enter the commands:

```
rewind /dev/rdsk/id_4s7
dd if=/dev/rdsk/id_4s0 of=/dev/rdsk/id_0s7 bs=16k seek=1460 count=963
```

3.3.8 Logical Backup. If more sophisticated backups are needed, a small partition on a Winchester device can be kept as intermediate storage for backups using the **find | cpio** sequence. The partition can be copied to tape as described in paragraph 3.3.5.

3.3.9 Set SYSTEM V/68 Variables. The system environment contains four exported variables:

- MMU
- PROCESSOR
- STACKCHECK
- SYSTEM

The object distribution file **/etc/profile** contains the appropriate values for each variable for each system under Release 2, Version 1 and Version 1.1. For information about **FP**, the floating point variable, refer to *cc*(1). The defined values for each variable are as follows:

VARIABLE	VALUE
MMU	M68451
PROCESSOR	M68010
STACKCHECK	OFF
SYSTEM	VME120

A version of **/etc/profile** that is appropriate for an MVME121-based system is contained in the file **/etc/prfle.VME120**. Link the file to **/etc/profile** to generate the correct SYSTEM V/68 variables. The new values do not take effect until the system has been shut down and re-booted.

3.3.10 Generate a Kernel. Information about the kernel and kernel values may be found under the following manual entries: *config*(1M), *dfile*(4), *make*(1), *master*(4), and *sysdef*(1M).

If it is necessary to generate a new kernel, enter the commands:

```
cd /usr/src/uts/m68k
make vme120_x
```

where *x* is replaced with:

3 to generate a 40Mb system, MVME319 kernel

The new kernel is the file **VME120_3unix**, which is located in the directory **/usr/src/uts/m68k/M68010**.

Move the newly created kernel to the root directory for testing with the commands:

```
cp M68010/VME120_3unix    /unix.test
chmod 0664    /unix.test
```

Change directories to root, and if the **usr** and **/usr/src** filesystems are still mounted, unmount them with the commands:

```
cd /
umount /dev/dsk/id_0s5
umount /dev/dsk/id_0s1
```

Use the *sync*(1) utility to flush unwritten system buffers out to disk and save all changes. Enter the command three times:

```
sync
sync
sync
```

Wait until the drive light is no longer illuminated, then press the RESET button on the MVME050 board.

Wait for the MVME120bug prompt and bootstrap the system with the command:

```
bo „unix.test
```

Test to be sure the kernel is working properly.

After the kernel has been tested and proven, rename it (e.g., **new319unix**). When you are ready to make the new kernel the default booting kernel, link it to **/unix** and to **/stand/unix**, and do a "sync". (As an extra precaution, some administrators may want to copy the old kernel, **/unix**, to **/unix.old** before linking. Once the new kernel is linked to **/unix** and **/stand/unix**, the old kernel will be lost.)

The command sequence is:

```
cd /
mv unix.test new_kernel
ln new_kernel unix
ln new_kernel stand/unix
sync
sync
sync
```

Wait until the drive light is no longer illuminated. Press the RESET button on the MVME050 board and boot from the new kernel with the command:

```
bo
```

The table that follows lists the names of the system kernels and the new kernels that result from the *make*(1) command.

KERNEL	KERNEL PRODUCED BY MAKE	DESCRIPTION
vme120_1	/usr/src/uts/m68k/M68010/VME120_1unix	15Mb system, MVME320
vme120_2	/usr/src/uts/m68k/M68010/VME120_2unix	40Mb system, MVME320
vme120_3	/usr/src/uts/m68k/M68010/VME120_3unix	40Mb system, MVME319

(**NOTE:** A 15Mb system is not supported by this release.)

3.3.11 Configuration Planning. The following table lists the values of the basic parameters for the MVME121-based system. For more details of syntax and structure, refer to *config.68*(1), *dfile*(4), and the associated *master*(4).

ITEM	VALUE
nswap	4752
buffers	50
hashbuf	64
physbuf	4
inodes	90
files	90
mounts	8
coremap	100
swapmap	75
calls	50
procs	50
texts	40
clists	150
maxproc	25

SYSTEM V/68 Release 2, Version 1.1 supports cache with the inclusion of a new cache variable in the **master** and **dfile** files (refer to *master*(4) and *dfile*(4)). The variable is a cache descriptor derived from "OR-ing" applicable flags that describe the cache. The following table describes the various cache attributes.

Attribute	Descriptor Value	Description
LOGICAL	01	Cache is logical addresses.
PHYSICAL	02	Cache is physical addresses.
SUPERVISOR	04	Cache is supervisor.
USER	010	Cache is user.
DATA	020	Cache contains data.
INSTRUCTION	040	Cache contains instruction.

The attributes of the system cache are logical and instruction for both supervisor and user. The value of the cache flag in **master** is set to 055.

NOTES

4. ERROR MESSAGES FOR THE MVME319 CONTROLLER

Error messages for the MVME319 disk controller and suggested actions are summarized in the following table. Refer to the *MVME319 Intelligent Disk Controller User's Manual* for additional information about these messages and the error codes and types. Refer also to the following entries in the SYSTEM V/68 documentation: *id*(7), *idfmt*(1M), *badlist*(4), *rewind* (volcopy(1M)), *ftape*(7); and the local include files **/sys/io/idio.h** and **/sys/id.h.**

All error messages are preceded by the notification line:

ID: Error on IDC #*n* Unit #*n* Type #*n* Code #*n*

data CRC error at sector # (hex)

SUGGESTED ACTION: Reformat device with the correct lockout table.

write protected

SUGGESTED ACTION: Remove write protection and try again.

not ready

SUGGESTED ACTION: Device is not online. Enable device and try again.

invalid drive number

SUGGESTED ACTION: Check **/dev** directory for the special file for the id driver. The system is attempting to talk to a device that does not exist.

invalid device address: sector # (hex)

SUGGESTED ACTION: The sector address exceeds the total number of sectors on the disk. Check *dfile*(4) and **idio.h** configuration for correct slicing. Relink a new kernel with correct configuration.

restore error

SUGGESTED ACTION: There was an unsuccessful attempt to detect track zero. Either the disk malfunctioned or there is an incorrect stepping rate code in **idio.h** configuration. Relink a new kernel with correct configuration.

record not found: sector # (hex)

SUGGESTED ACTION: Reformat device with the correct lockout table.

id CRC error at sector # (hex)

SUGGESTED ACTION: Reformat device with the correct lockout table.

VMEbus DMA error

DESCRIPTION: VMEbus bus request or data transfer time-out. Hardware malfunction.

controller error

SUGGESTED ACTION: Internal malfunction from MVME319 controller, SASI/SCSI controller error, or nonexistent SASI/SCSI controller. Reset system and boot again. If problem recurs, check **idio.h** configuration. Relink a new kernel with correct configuration.

drive error

SUGGESTED ACTION: Configuration problem or device malfunction. Check **idio.h** configuration. Relink a new kernel with correct configuration.

seek error

SUGGESTED ACTION: Reformat disk.

I/O DMA error

DESCRIPTION: Hardware malfunction. Local bus time-out on SASI/SCSI bus.

5. OPERATOR INFORMATION FOR MVME121/MVME320

5.1 General

This chapter describes the hardware configuration for an MVME121-based system with the MVME320 running on SYSTEM V/68 software. The following information is provided:

- Minimum system configuration
- Default system configuration
- Hardware Installation
- Boot procedures
- Board-level operations

5.2 Minimum System Configuration

A minimum configuration for an MVME121-based system including an MVME320 that supports SYSTEM V/68 software includes the following components:

a. MVME121 VMEbus Microprocessor Module with 512Kb of RAM, the MC68451 MMU, 4Kb logical instruction cache, and one serial communication port.

b. MVME050 System Controller Module with two serial communication ports and one Centronics printer interface

c. MVME701 I/O Transition Module for the controller module

d. MVME202 512Kb Dynamic Memory Module with byte parity

e. MVME320 VMEbus Disk Controller with two hard disk interfaces and two diskette interfaces

f. MVME702 Disk Interface Module, one per MVME320 controller

g. 40Mb Micropolis or Vertex 5¼-inch Winchester drive

h. VME card cage with 300W power supply and ventilation

i. MVME12xbug firmware-resident monitor for loading and debugging programs, and for booting SYSTEM V/68

5.3 Default System Configuration

SYSTEM V/68 is configured to support the following system components by default:

- One MVME121 monoboard microcomputer (including one terminal interface)
- One MVME050 system controller and one MVME701 I/O transition module (including two serial and one Centronics interface)
- Two MVME320 VMEbus disk controllers
- Two MVME702 Disk Interface Modules
- Four Winchester disk drives (40Mb)
- Four 5¼-inch diskette drives
- Several additional MVME201, MVME202 or MVME222 memory modules

5.4 Hardware Installation

This section describes the jumper and switch settings required to run an MVME121-based system under the default configuration of SYSTEM V/68. A non-standard hardware configuration must be reflected in the SYSTEM V/68 *master*(4) file and *dfile*(4), and requires the generation of a new kernel. For further information, refer to *master*(4), *dfile*(4), and *config*(1M).

5.4.1 Installation Considerations. Provide adequate grounding. It is important that the chassis ground is interconnected (e.g., drives, enclosure), and wired to the signal ground in one place only, typically at the power supply.

CAUTION

AVOID TOUCHING AREAS OF INTEGRATED CIRCUITRY; STATIC DISCHARGE CAN DAMAGE THESE CIRCUITS.

CAUTION

INSERTING/REMOVING MODULES WHILE POWER IS APPLIED MAY RESULT IN DAMAGE TO MODULE COMPONENTS.

5.4.2 MVME121 Microprocessor. SYSTEM V/68 requires an MVME121 microprocessor with a Memory Management Unit (MMU) installed and 512Kb of on-board memory.

The MVME121 SYSTEM V/68-supported jumper settings are listed in Table 5-1 and illustrated in Figure 5-1.

Table 5-1. MVME121 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
J2	ACFAIL/SYSFAIL Select	2-3
J3	VMEbus Request Level Select	1-3, 4-6, 5-7, 9-11
J4	VMEbus Request Level Select	3-5, 4-6
J5	Abort Switch Disable Select	1-2
J6	Reset Switch Disable Select	1-2
J7	VMEbus Interrupt Connection Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14
J8	ROM/EPROM Device Configuration Select	1-2, 5-7, 9-10 11-12, 13-14
J9	Cache Configuration Select	1-2
J17	Cache Configuration Select	No Jumpers
J10	MMU Strobe Select	2-3
J11	MMU Address Strobe Select	No Jumpers
J12	Map Decode Time Select	2-3
J13	RAM Cycle Timing Select	1-2
J14	RAM Cycle Timing Select	1-2
J16	RAM Cycle Timing Select	2-3
J18	RAM Cycle Timing Select	2-3
J19	RAM Cycle Timing Select	2-3
J15	Refresh Period Select	No Jumper
J20	MSR Bit 1 Select	No Jumper
J21	MSR Bit 2 Select	2-3
J22	ROM Access Time Select	7-8
J23	RAM Size Select	1-2
J24	Reset Vector Fetch Mode Select	1-2
J25	Local Time-Out Select	1-2
	MVME121 BACKPLANE JUMPER SETTINGS	
A21-A22	IACKIN* TO IACKOUT*	No Jumpers
B4-B5	BG0IN* TO BG0OUT*	No Jumpers
B6-B7	BG1IN* TO BG1OUT*	No Jumpers
B8-B9	BG2IN* TO BG2OUT*	No Jumpers
B10-B11	BG3IN* TO BG3OUT*	No Jumpers

On the MVME121, the S3 switch settings are dependent on the jumpering for J20 and J21. To support SYSTEM V/68, the S3 switch settings should be set as follows:

SWITCH NUMBER	DESCRIPTION	SETTING
S3-1	Autoboot Select	Open
S3-2	Baud Rate Select (12.5-10 MHz)	Closed
S3-3	Reset Vector Fetch Select	Closed
S3-4	Reset Vector Fetch Select	Open

The terminal interface for the MVME121 may be any TTY compatible terminal with a 25-pin RS-232C standard interface. The switch settings should be set as follows:

9600 Baud
Full duplex
8 data bits
1 stop bit
No parity

For more detailed information about the MVME121 microprocessor, refer to the *MVME120, MVME121, MVME122, MVME123 VMEbus Microprocessor Module User's Manual.*

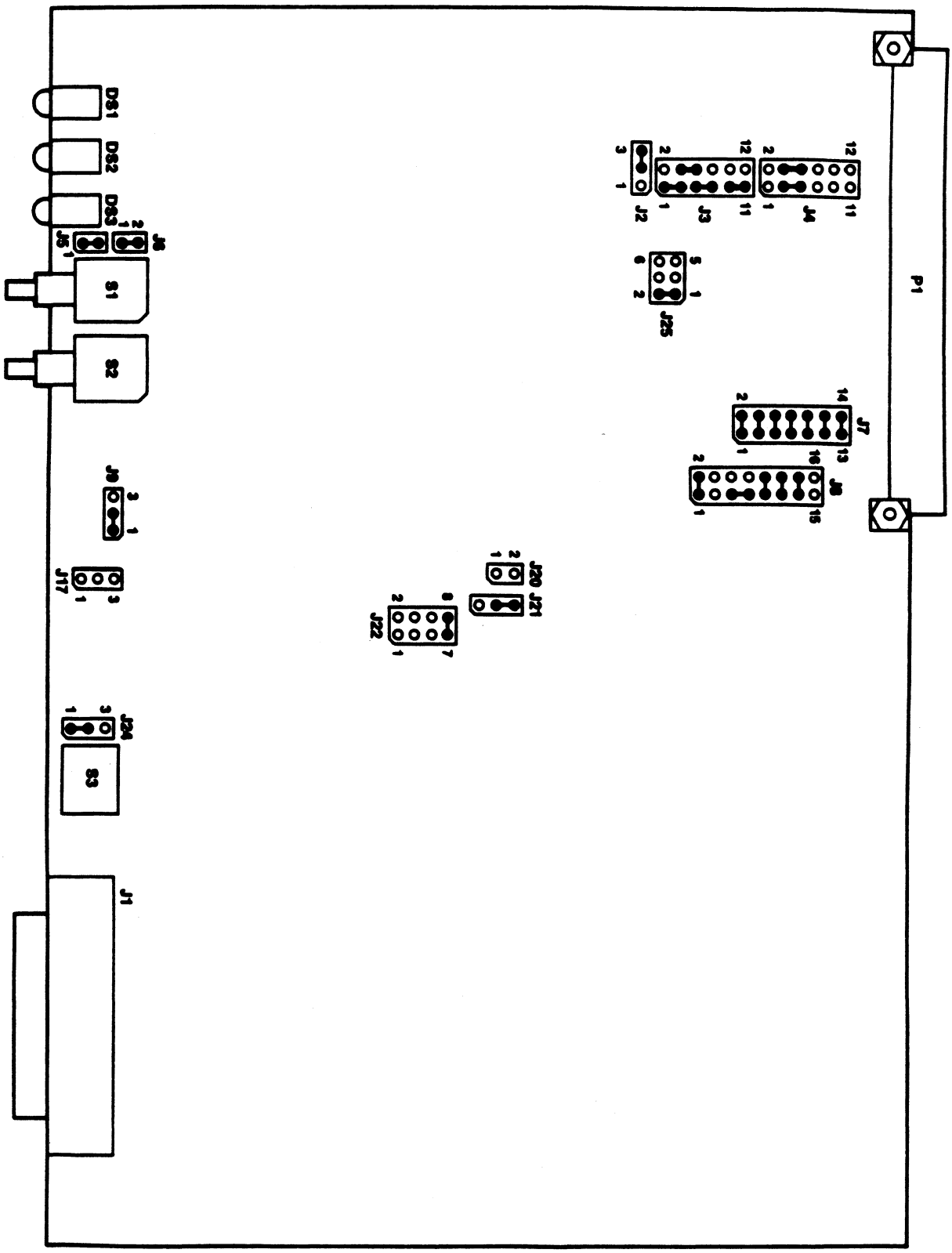


Figure 5-1. MVME121 Jumper Locations and Settings

5.4.3 MVME050 System Controller and MVME701 Transition Modules. SYSTEM V/68 supports the parallel port on the MVME050 controller module as a printer port. Header J29 selects the rising or falling edge of the printer acknowledge signal. The module is shipped with the jumper installed between pins 1 and 2 with the rising edge selected. The module is initialized to provide a Level 2 interrupt for the line printer. These are the configurations supported by the line printer driver for the MVME050 module. The line printer driver for the MVME050 module does not support unbuffered parallel to serial port conversion using the parallel port. The two serial ports on the MVME050 (or MVME701) interface with the terminals.

Tables 5-2 and 5-3 list the SYSTEM V/68-supported jumper settings for the two modules. Notice that Table 5-2 contains several jumpers that are considered "open." For these jumpers, SYSTEM V/68 supports any configuration selected by users. Figures 5-2 and 5-3 illustrate the jumper locations and settings for the MVME050 and MVME701 respectively. Notice that Figure 5-2 contains reference jumper settings for those jumpers that are considered "open" in Table 5-2. That is, Table 5-2 indicates SYSTEM V/68 supports all user-selected configurations for the "open" jumpers, and Figure 5-2 illustrates one standard configuration. Users may follow the jumpering shown in Figure 5-2 and eliminate the need to refer to the MVME050 manual.

The jumpering illustrated in Figure 5-2 differs from the MVME050 factory configuration at jumper locations J22, J23, and J24. The jumpering illustrated in Figure 5-3 differs from the MVME701 factory configuration at jumper locations J5, J6, J9, and J10.

The piano type 8-section switch on the front panel of the controller module, (S1), is a software readable switch. The user may include the functions of this switch in the software program. Section 8 of the switch may also function as blanking for the user status display.

For more detailed information about the MVME050 or MVME701, refer to the *MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual.*

Table 5-2. MVME050 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
J1	EPROM Base Address Select (A14-A23)	Open*
J2	EPROM Base Address Select (A24-A31)	Open*
J8	EPROM Size Select	Open*
J13	EPROM Quad Enable Select	Open*
J27	EPROM Access Time Select	Open*
J9	RAM Base Address Select (A14-A23)	Open*
J10	RAM Base Address Select (A24-A31)	Open*
J26	RAM Access Time Select	Open*
J14	RAM Quad Enable Select	Open*
J15	RAM Size Select	Open*
J12	AM1E Enable Select (EPROM Quad 1)	Open*
	AM16 Enable Select (EPROM Quad 2)	Open*
J3, J16, J17	EPROM/RAM Configuration Select (EPROM/RAM Quad 1)	Open*
J11, J18, J19	EPROM/RAM Configuration Select (EPROM/RAM Quad 2)	Open*
J4	Bus Time-Out Select	9-10
J5	Serial Clock Damping/Shorting Select	1-2
J6	System Clock Damping/Shorting Select	1-2
J7	System Controller Select	1-2, 3-4, 5-6, 7-8
J20	I/O Base Address Select	1-2, 3-4, 5-6, 9-10, 11-12, 13-14
J21	Time-of-day Clock Power and Battery Charge Select	1-3, 2-4, 7-8
J22	System Fail or Ground for Interrupt Source Select	1-2
J23	Internal/External Transmit Clock Serial Port 1 Select	1-2
J24	Internal/External Transmit Clock Serial Port 2 Select	1-2
J25	Display Blanking Enable Select	1-2
J28	Reset Switch Disable Select	1-2
J29	Printer Acknowledge Edge Select	1-2
	Open*: Any configuration selected by users as appropriate for their applications is supported by SYSTEM V/68.	
	MVME050 BACKPLANE JUMPER SETTINGS	
A21-A22	IACKIN* TO IACKOUT*	No Jumpers
B4-B5	BGOIN* TO BGOOUT*	No Jumpers
B6-B7	BG1IN* TO BG1OUT*	No Jumpers
B8-B9	BG2IN* TO BG2OUT*	No Jumpers
B10-B11	BG2IN* TO BG3OUT*	No Jumpers

Table 5-3. MVME701 SYSTEM V/68-Supported Jumpers

JUMPER NUMBER	DESCRIPTION	SETTING
J5	Port 1 to Terminal Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14 15-16, 17-18, 19-20
J6	Port 1 to Modem Select	No Jumpers
J11	Port 1 CTS Select	2-3
J7	Port 1 DSR Select	2-3
J9	Port 2 to Terminal Select	No Jumpers
J10	Port 2 to Modem Select	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14 15-16, 17-18, 19-20
J8	Port 2 DSR Select	2-3
J12	Port 2 CTS Select	2-3

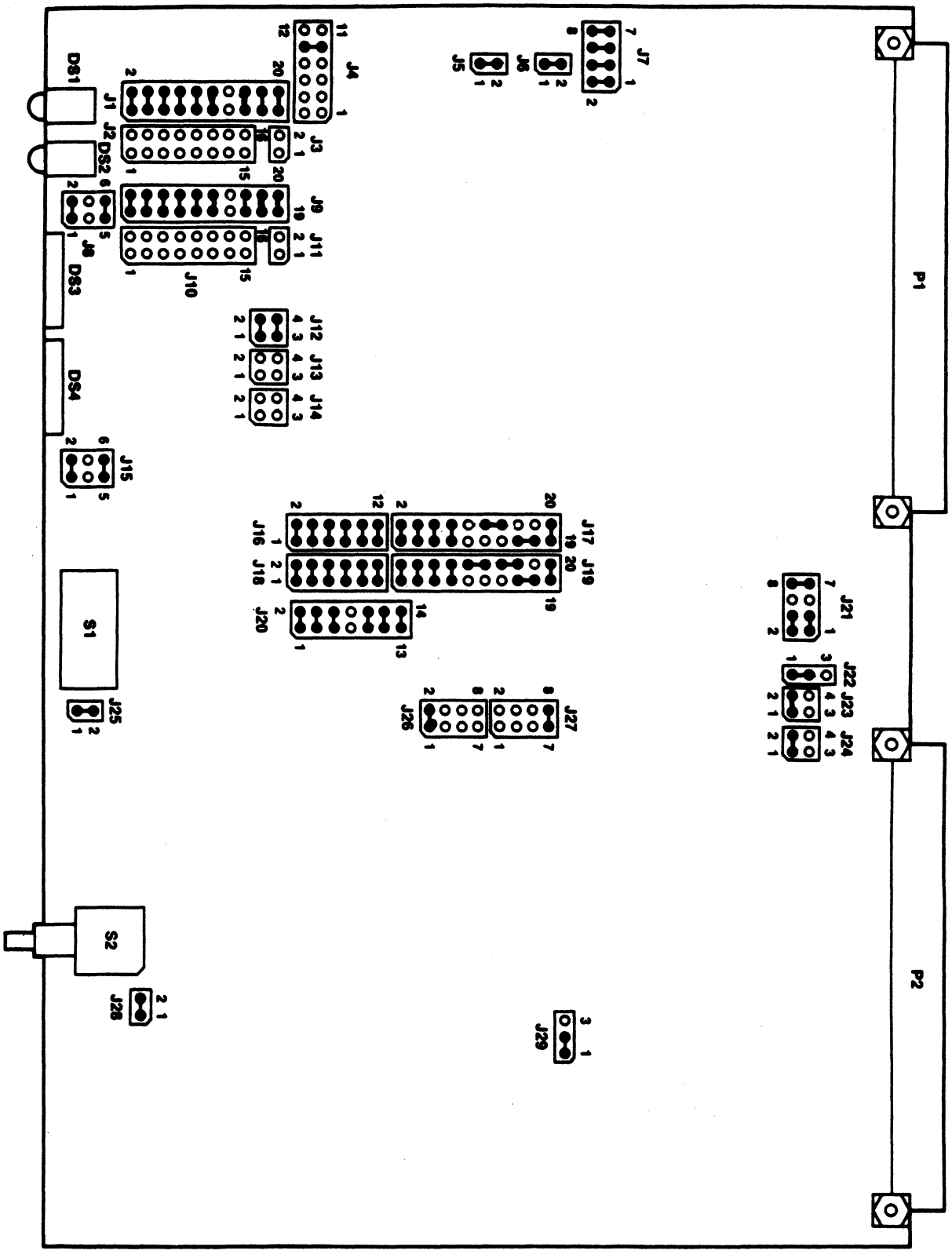


Figure 5-2. MVME050 Jumper Locations and Settings

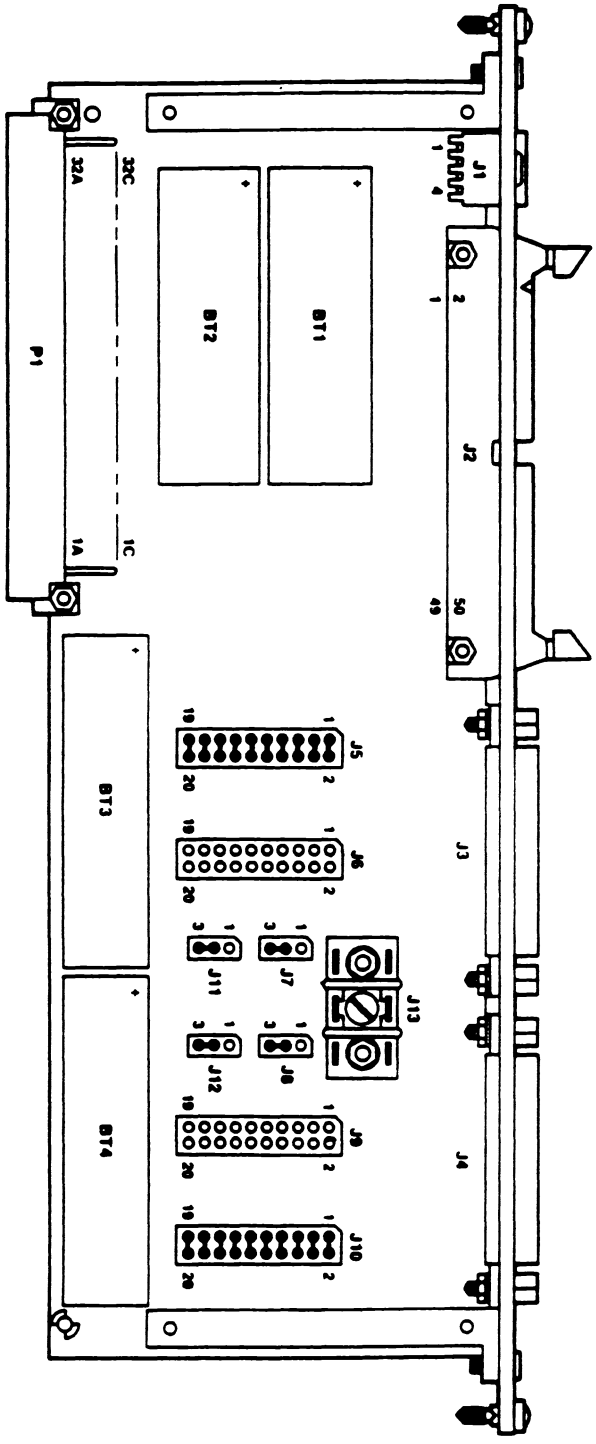


Figure 5-3. MVME701 Jumper Locations and Settings

5.4.4 MVME320 Disk Controller and MVME702 Interface.

5.4.4.1 MVME320. The MVME320 Disk Controller Module adds mass storage capability to a VMEbus-based system. Data is transferred between system memory and Winchester hard disk drives or floppy disk drives over high-performance Direct Memory Access (DMA) channels.

The MVME320 can control one or two Winchester hard disk and two 5¼-inch diskette drives. All drives connected to the MVME320 must be soft-sectored.

The jumper header configurations as shipped from the factory support SYSTEM V/68. No changes need be made. The default jumper header settings are listed in Table 5-4 and illustrated in Figure 5-4. For more detailed information about the configuration options, refer to the *MVME320 VMEbus Disk Controller Module User's Manual.*

The system's disk storage capacity can be expanded by adding one or more disk drives or disk controllers. An additional MVME320 disk controller improves disk data throughput significantly when the root filesystem and user filesystems are served by separate MVME320 controllers. The VME short I/O address must be strapped to $FFAC00 for the second MVME320 controller. Different Winchester devices can be supported by adding new parameter entries to the file **/etc/diskdefs**, as long as the devices are compatible with the MVME320 controllers. Refer to the *MVME320 VMEbus Disk Controller User's Manual* and the *m320*(7) entry in *SYSTEM V/68 Administrator's Manual* for further information.

Table 5-4. MVME320 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
JW0	PROM Size Select	No option.
JW1	Interrupt Request Level Select (IRQ Level 3)	3-8
JW2	Interrupt Acknowledge Level Select (IACK Level 3)	2-5, 3-6
JW3	Bus Request Level Select (BUS REQ Level 3)	1-2
JW4	Bus Grant Level Select (BUS GRANT Level 3)	1-9, 2-10, 3-4 5-6, 7-8
JW5	Address Modifier Select (User or Supervisor Access)	2-3
JW6	Memory Address Select (Default = 101100)	1-2, 3-4, 9-10
MVME320 BACKPLANE JUMPER SETTINGS		
A21-A22	IACKIN* TO IACKOUT*	No jumpers
B4-B5	BG0IN* TO BG0OUT*	Jumper
B6-B7	BG1IN* TO BG1OUT*	Jumper
B8-B9	BG2IN* TO BG2OUT*	Jumper
B10-B11	BG3IN* TO BG3OUT*	No jumpers

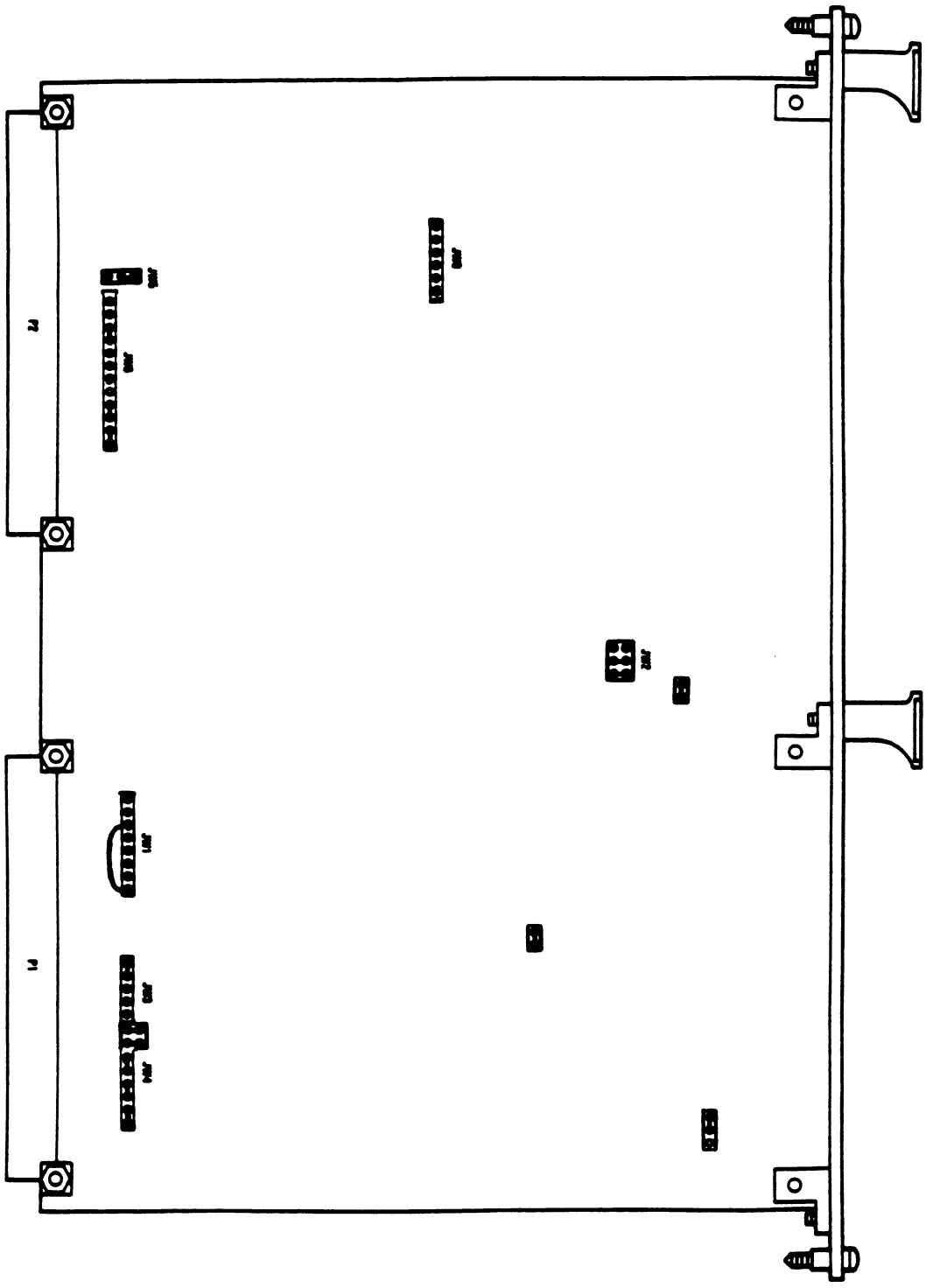


Figure 5-4. MVME320 Jumper Locations and Settings

5.4.4.2 MVME702. The MVME702 Disk Interface Module is a double-high VMEmodule, designed to perform signal switching and provide standard disk cable connections for different drives. The module as shipped from the factory supports SYSTEM V/68. The required jumper settings are listed in Table 5-5 and illustrated in Figure 5-5. For detailed information about the jumper settings, refer to the *MVME702 Disk Interface Module User's Manual.*

Table 5-5. MVME702 SYSTEM V/68-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
J4	Signal Termination Select	13-14
J5	Signal Termination Select	1-2, 3-4, 7-8, 9-10, 11-12, 13-14
J7	Signal Termination Select	3-4, 5-6, 7-8, 9-10, 11-12, 13-14

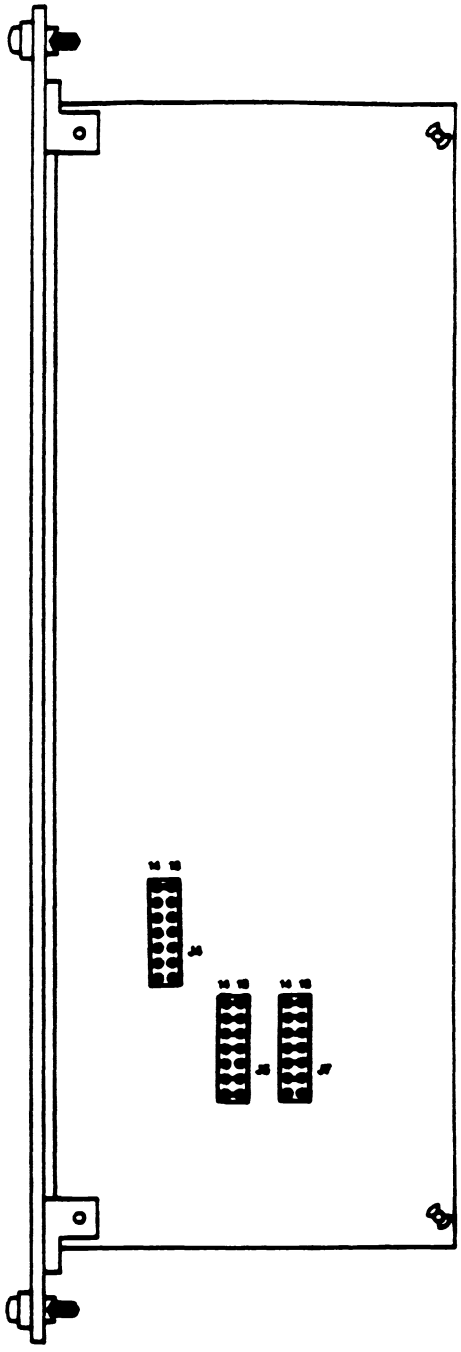


Figure 5-5. MVME702 Jumper Locations and Settings

5.4.5 Memory. An MVME121-based system, as configured, contains .5Mb of dynamic memory. If more memory is needed, additional MVME201, MVME202 or MVME222 memory modules can be used. The MVME201 and MVME202 modules must be strapped to provide contiguous addressing starting at hex address $80000. Refer to the *MVME202/MVME222-1/MVME222-2 512KB/1MB/2MB Dynamic Memory Module User's Manual* for further information.

5.4.6 Cache. An MVME121-based system has 4Kb of on-board cache. The cache can be jumpered to support four different cache configurations, as described in Table 5-6. A series of Benchmark tests were run to determine the optimal cache configuration for SYSTEM V/68. The test results indicate the operating system performs best when the cache is configured for mixed supervisor and user.

Table 5-6. Cache Configurations for MVME121-based System

CONFIGURATION	JUMPER J9	JUMPER J17
4Kb for user only	pins 2-3	pins 1-2
4Kb for supervisor only	pins 2-3	pins 2-3
4Kb mixed supervisor and user	pins 2-3	remove all jumpers
2Kb supervisor, 2Kb user	pins 1-2	remove all jumpers

On the MVME121 board, both Jumpers J9 and J17 have 3 pins associated with them. A rectangle with a notched corner has been drawn around the pins; Pin 1 is the pin closest to the notch. The recommended setting (Table 5-1) is also the default setting as shipped from the factory, and is configured for 2Kb supervisor, 2Kb user.

Cache on the MVME121 can be enabled/disabled in software. In some instances, it is desirable to disable cache regardless of the software. To disable the cache:

1. Connect J20 pins 1 and 2, using a shorting jumper.
2. Place S3 position 1 in the closed/on position.
3. Press RESET on the board.

Notice that when S3 is in the closed position, the MVME12xbug performs auto-boot when power is first supplied to the system. If auto-boot is not desired, press the system RESET button about 5 seconds after power is supplied.

To remove the hardware disable of cache function:

1. Disconnect J20 pins 1 and 2. (Remove the shorting jumper.)
2. Press RESET on the board.

Notice that when J20 pins 1 and 2 are disconnected, S3 position does not affect the hardware disable of cache.

5.5 Boot Procedures

This section is provided as reference information. DO NOT BOOT the system without first reading Chapter 6 and installing the software according to the step-by-step installation plan.

5.5.1 Boot From the Winchester Drive. The following procedure boots the system after a routine shutdown, or after the system has crashed.

1. Power up the system. Press the RESET button on the MVME050 board to reset the system. The MVME12xbug debug monitor displays the following question on the console:

   Size and initialize RAM (y/n) ?

   Answer with **y** . (Later versions of the debug monitor may not display this message.)

2. The MVME12xbug prompt appears. Wait a few seconds for the Winchester drive to spin up. The Winchester disk drive corresponds to Logical Unit 0. Bootstrap the operating system from the Winchester disk with the MVME12xbug boot command:

   bo

3. SYSTEM V/68 is loaded into memory and executed. The system reports the amount of available memory and comes up in "single user mode".

4. Enter multi-user mode immediately with the command:

   init 2

5.5.2 Boot From the Diskette Drive. This section is provided as reference information. Do not boot the system without first reading Chapter 6 and installing the software according to the step-by-step installation plan.

If you are unable to boot from the Winchester disk, the minimum system utility diskette supplied with this release can boot the system. Two "minload" diskettes labeled "1 of 4" and "2 of 4" are included in the Release 2, Version 1.1 distribution. Each minload diskette is designed to be used with a particular system: the diskette labeled "2 of 4" is the minload diskette for customers whose systems include an MVME320 disk controller.

DO NOT USE the diskette labeled "1 of 4"; it is the minload diskette for MVME121/MVME319 systems.

To boot your system from the minload diskette:

1. Remove the write protect label from the minload diskette labeled "2 of 4". Insert the diskette into the diskette drive. The diskette drive corresponds to the Logical Unit 2.

2. Bootstrap the operating system from the diskette with the MVME12xbug boot command:

   bo 2

5.5.3 Boot Errors. If MVME12xbug delivers an error message during the boot procedure, reset the system by pressing the MVME050 system controller RESET button. Enter the command: **vi** .

Repeat the boot procedure. If the problem remains, power off the system and power it on again after a few seconds. Try the boot procedure again. If this does not help, try to boot from a different media. Do not try to boot from the "1 of 4" minload diskette.

5.6 Board-level Operations

This section provides information regarding special operation of MVME121-based system switches and LEDs when SYSTEM V/68 is installed. SYSTEM V/68-specific information with respect to the operation of the firmware-resident monitor/debugger MVME12xbug is included in paragraph 5.7 and in the *MVME12x Debug Monitor User's Manual.*

5.6.1 MVME121 Board. The MVME121 board contains the following pushbutton control switches:

RESET

When this momentary-action pushbutton switch is pressed, it resets the MVME121 logic circuits. If the MVME121 is in the SYSTEM V/68 operating system, MVME12xbug is entered by pressing RESET.

ABORT

When this momentary-action pushbutton switch is pressed, the MVME121 enters MVME12xbug, but the MVME121 logic circuits are not reset. After an abort, the user can enter **G** to continue execution of the current program prior to the abort.

NOTE: On the MVME120 family of processor boards, the level 7 interrupt that should result from pushing the ABORT button is not generated while a STOP instruction is being executed. That is, if the processor is executing a STOP instruction, the firmware debugger Software Abort function does not give control to the user until another interrupt occurs that ends the STOP instruction execution.

5.6.2 MVME050 Board. The MVME050 board contains a RESET pushbutton. When the RESET button is pressed, a system-level function is initiated that enables the RESET signal to generate the system reset, SYSRESET*. The SYSRESET* is sent to the other modules in the system through the VMEbus.

For further information about the operation of the MVME050 module, refer to the *MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual.*

5.7 MVME12xbug Firmware Monitor Commands

MVME12xbug is the firmware-resident monitor/debugger from which SYSTEM V/68 is booted. The MVME12xbug prompt is

MVME12x 2.*r* >

where *r* identifies the revision number of the debug monitor.

Seven MVME12xbug primitive commands should **NOT** be used with SYSTEM V/68. The following commands can be used only in a stand-alone mode.

DU	Dump memory (S-records).
LO	Load (S-records).
PF	Port format.
ST	Self-test.
TM	Transparent mode.
VE	Verify (S-records).
VI	Vector initialize.

The following commands may be useful for debugging, but should be used only in single-user mode after *sync*(1) has executed. Use of these commands could result in the need for system reboot. More detailed information about the MVME12xbug commands is provided in the *MVME12xbug Debugging Package User's Manual.*

BD Bootstrap dump.

BF Block fill. (Note: MVME12xbug distinguishes between uppercase and lowercase characters when filling a block.)

BH	Boot and halt.
BI	Block initialize.
BM	Block move.
BO	Boot operating system.
BR, NOBR	Set and remove breakpoints.
BS	Block search.
BT	Block test.
CACHE	Manipulate cache memory.
CD	Cache display.
DC	Data conversion/evaluation.
DF	Display formatted registers.
GD	Go direct.
GO	Install breakpoints and go.
GT	Go until address.
HE	Display commands/registers.
IOC	Issue disk I/O command.
IOP	Issue physical read/write.
IOT	Teach disk controller a configuration.
MD	Memory display.
MM	Memory modify.
MP	Map memory for disk I/O.
MS	Memory set.
OF	Offset register display.
PA, NOPA	Printer attach/detach.
TA	Terminal attach.
TR	Trace.
TT	Trace until address.

.<register>

- .A0 - .A7
- .D0 - .D7
- .DFC
- .PC
- .R0 - .R6
- .SFC

- .SR
- .SSP
- .USP
- .VBR

(BREAK) Abort command or process.

(DEL) Delete character.

(CTRL)D Redisplay line.

(CTRL)H Delete character.

(CTRL)S Suspend output; any character continues.

(CTRL)X Cancel command line.

(RETURN) Process current/previous command line.

NOTES

6. SOFTWARE INSTALLATION FOR MVME121/MVME320

This chapter contains step-by-step instructions for installing SYSTEM V/68 Release 2, Version 1.1 object and/or source code. Administrative procedures that are needed during the installation and are considered part of system administration (e.g., initializing media, changing system variables, generating a kernel) are also included.

6.1 Distribution Description

SYSTEM V/68 Release 2, Version 1.1 is available in both object and source distributions. Both object and source distributions use SYSTEM V/68 Release 2, Version 1 as their foundation. The Version 1.1 release is loaded in layers of code: the base is Release 2, Version 1 code; the Release 2, Version 1 Update is loaded next; and overlaying both is the Version 1.1 code. The Release 2, Version 1.1 distribution is composed of the following diskettes:

- Release 2, Version 1 object and/or source code
- Release 2, Version 1 Update object and/or source code
- Release 2, Version 1.1 object and/or source code

6.1.1 Release 2, Version 1.1 Object Distribution. The Release 2, Version 1 object distribution consists of 33 diskettes; the source distribution consists of 24 diskettes. The Release 2, Version 1 Update distribution varies in size with each update. The distribution is described in the customer letter that accompanies the diskette(s). The Release 2, Version 1.1 object distribution consists of 4 diskettes.

Not all diskettes from these three sets are loaded to form the complete Release 2, Version 1.1 object distribution. Several diskettes will damage the system if they are loaded. At this point, clearly mark the following diskettes, or set them aside, to ensure that they will not be loaded during the installation procedure.

CAUTION

DO NOT LOAD THE FOLLOWING DISKETTES AS PART OF THE RELEASE 2, VERSION 1.1 DISTRIBUTION. THESE DISKETTES CONTAIN KERNELS AND OTHER FILES THAT ARE NOT COMPATIBLE WITH THE MVME121/MVME320 SYSTEM. IF THE DISKETTES ARE LOADED, THE SYSTEM WILL NOT BOOT.

DO NOT LOAD THESE DISKETTES:		
Set	**Diskette Number**	**Description**
Release 2, Version 1	***82V1XBSVB1**	**VME/10 stand-alone boot**
	"0 of 31"	**VME/10 minload**
	"5 of 31"	**VME/10 files**
	"24 of 31"	**VM03 files**
	"25 of 31"	**VM03 files**
Release 2, Version 1.1	**"1 of 4"**	**MVME319 files**

The complete SYSTEM V/68 Release 2, Version 1.1 object distribution is listed in Table 6-1. The table lists each diskette, its *cpio* block count and a description of the files contained on the diskette. Notice that the Release 2, Version 1 Update diskette(s) block count is not included in the table. Because the update may change quarterly, the block counts for the Update are included in the *SYSTEM V/68 Update Customer Letter*.

Table 6-1. Release 2, Version 1.1 Object Diskettes: Description and Block Counts

Set	Diskette Number	Cpio Block Count	Description
RELEASE 2, VERSION 1	1 of 31	1159	/bin
	2 of 31	1092	/bin
	3 of 31	1160	/etc
	4 of 31	981	/etc, /usr subdirectories
	5 of 31		**DO NOT USE.**
	6 of 31	1139	/lib
	7 of 31	1152	/usr/bin
	8 of 31	1151	/usr/bin
	9 of 31	1121	/usr/bin
	10 of 31	1149	/usr/bin
	11 of 31	1141	/usr/bin, /usr/lib
	12 of 31	1109	/usr/lib
	13 of 31	1114	/usr/lib
	14 of 31	620	Includes
	15 of 31	685	Accounting
	16 of 31	1097	Text processing
	17 of 31	959	Text processing
	18 of 31	781	Games
	19 of 31	1082	Graphics
	20 of 31	1067	Graphics
	21 of 31	816	Graphics
	22 of 31	736	Uucp
	23 of 31	553	Kernel build area
	24 of 31		**DO NOT USE.**
	25 of 31		**DO NOT USE.**
	26 of 31	614	Terminfo
	27 of 31	1129	On-line documentation
	28 of 31	838	On-line documentation
	29 of 31	901	On-line documentation
	30 of 31	417	On-line documentation
	31 of 31	619	Miscellaneous utilities
RELEASE 2, VERSION 1 UPDATE			**Refer to Update Customer Letter.**
RELEASE 2, VERSION 1.1	1 of 4		**DO NOT USE.**
	2 of 4	1113	MVME121/320 Minload
	3 of 4	1008	Updated object
	4 of 4	258	Kernel build area On-line documentation

6.1.2 Release 2, Version 1.1 Source Distribution. Source distribution for Release 2, Version 1.1 includes the source code for Release 2, Version 1; the Release 2, Version 1 Update for source; and the Release 2, Version 1.1 source code. Like the object code, source code is installed in layers. Table 6-2 lists the source diskettes and their *cpio* block counts. To install the source code, users must first complete the object code installation. The source installation begins as Step 11 in the installation procedure.

Table 6-2. Release 2, Version 1.1 Source Diskettes and Block Counts

Set	Diskette Number	Cpio Block Count
RELEASE 2, Version 1	1 of 24	845
	2 of 24	990
	3 of 24	891
	4 of 24	547
	5 of 24	638
	6 of 24	1038
	7 of 24	939
	8 of 24	938
	9 of 24	1025
	10 of 24	728
	11 of 24	834
	12 of 24	985
	13 of 24	946
	14 of 24	1027
	15 of 24	1113
	16 of 24	1110
	17 of 24	1048
	18 of 24	1109
	19 of 24	981
	20 of 24	997
	21 of 24	1012
	22 of 24	1053
	23 of 24	1087
	24 of 24	805
RELEASE 2, VERSION 1 UPDATE		**Refer to Update Customer Letter**
RELEASE 2, VERSION 1.1	1 of 1	862

6.2 Object and Source Installation Instructions

The procedure to install the SYSTEM V/68 object distribution is described in this paragraph as a series of ten steps. Source distribution installation follows the object installation, and continues with 3 additional steps, 11 through 13.

During the installation, users may want to refer to additional information that is provided toward the end of this chapter. References to the information are located in the steps where the information may be needed.

STEP 1: DISK SPACE CONSIDERATIONS

You should have labeled six diskettes from the object distribution **DO NOT USE**, or have set them aside to ensure they will not be loaded. (Refer to paragraph 6.1.1.) By not loading these diskettes, administrators gain approximately 1500 blocks of extra file space in their systems.

After considering the number of users and the type of processing anticipated, administrators of 40Mb systems may wish to omit some of the SYSTEM V/68 software functions to allow more space for user files. The following table lists the block counts for some software functions that can be omitted easily.

Function	Diskette Number	Block Count
Accounting	Diskette 15	685
Games	Diskette 18	781
Graphics	Diskettes 19, 20, 21	2965
Text Processing	Diskettes 16, 17	2056
On-line Documentation	Diskettes 27-30	2280

If any diskettes are to be omitted from the loading procedure, label them now and set them aside.

STEP 2: POWER UP AND BOOT FROM DISKETTE

a. Power up the system.

b. Respond to the MVME12xbug prompt "Size and Initialize RAM (Y/N)?" with **y** . (Later versions of the debug monitor may not prompt with this question.)

c. REMOVE THE WRITE PROTECT TAPE from the minload diskette labeled "2 of 4" and insert the diskette into the device drive #2. Boot from the diskette with the command:

bo 2

The following messages appear on the terminal screen:

M68010/sysV68: unixR2V1.1
real memory = *n*
available memory = *n*

INIT: SINGLE USER MODE

If the Winchester drive has not yet been formatted, or if the configuration information on the disk has been destroyed, the following two error messages appear on the screen:

M320: drive 0, ctl 0, cmd 5, main 1, ext 0x8, i/o error

M320: drive 0, ctl 0, open found no format/config

Both messages indicate the Winchester drive is not formatted properly. The messages may be ignored for now; the Winchester drive will be formatted in the next step.

STEP 3: FORMAT DISK AND LOAD MINLOAD ONTO FIXED DISK

a. When the SYSTEM V/68 prompt # appears, enter the command:

/etc/loadtools/minload

The screen displays a script that formats the Winchester disk, and loads boot information and a minimal set of utilities. The script asks whether the user wants to reformat the Winchester disk, and prompts for information about the disk.

To answer the prompts from the system, you must know the manufacturer of your disk, either Micropolis or Vertex, and the location of any disk defects. The necessary information is provided in the Winchester verification report that is packed with the disk drive. Paragraph 6.3.2 contains instructions for calculating bad block locations from the information included in your verification report. (Instructions for formatting a Winchester disk without loading the minimum system utilities, i.e., when formatting a second disk, are included in paragraph 6.3.3.)

b. Format the disk, following the prompts and directions that appear on the screen as part of the **minload** routine.

c. After the formatting routine, the system begins to copy the minimum system utilities onto the disk. Approximately 50 filenames are listed on the screen as they are copied. The screen should indicate that 1113 blocks have been copied. If some other number of blocks appears, repeat this step.

STEP 4: BOOT FROM FIXED DISK

When the lights on both drives are no longer illuminated, press the RESET button on the MVME050 module. In response to the MVME12xbug prompt, enter the boot command:

bo

The command boots the MVME121-based system from the fixed disk.

STEP 5: BACKUP OBJECT DISKETTES

To backup the object distribution diskettes, you need to format at least 32 diskettes with the *dinit*(1M) program:

28 diskettes for the SYSTEM V/68 Release 2, Version 1 object distribution,
(Diskettes 0, 5, 24 and 25 are not backed-up since they are not loaded.)
1 or more diskettes for the Release 2, Version 1 Update, and
3 diskettes for the Release 2, Version 1.1 distribution.
(Backup the minload diskette and the 2 object code diskettes.)

a. The procedure to format diskettes for Release 2, Version 1.1 is described in paragraph 6.3.4 of this manual. Format as many diskettes as you need to backup the distribution. (Users who will be installing source distribution may wish to backup the source distribution diskettes now.)

b. To backup the distribution, the master object distribution diskette is first copied onto the fixed disk. Then, a copy is made from the fixed disk onto a newly formatted, backup

diskette. To do this, first insert an object distribution diskette into the drive #2.

c. Enter the command:

dd if=/dev/rdsk/m320_2s7 of=/flop bs=16k

to copy the master diskette to the Winchester.

d. Remove the master diskette and insert a newly formatted diskette into the drive.

e. Enter the command:

dd if=/flop of=/dev/rdsk/m320_2s7 bs=16k

to copy from the Winchester to the new diskette.

f. Repeat these steps for each of the distribution diskettes.

STEP 6: CREATE AND MOUNT usr FILESYSTEM

The system administrator may want to modify the recommended allocation of space for object files. The recommended procedure, as outlined in this section, places the **usr** filesystem in partition 1. Partitions 5 and 6 are reserved for source distribution. ("Object only" customers may use these partitions however they deem most useful.)

To create and mount the **usr** filesystem in the proper partitions, enter the commands:

```
mkfs /dev/dsk/m320_0s1 28032:3200
labelit /dev/rdsk/m320_0s1 usr R2V1.1
mount /dev/dsk/m320_0s1 /usr
```

STEP 7: LOAD OBJECT DISKETTES

Before you insert a diskette into the drive or enter any of the commands in this step, read through the entire step to be sure you understand the complete procedure.

In addition, take the following precautions:

- MAKE SURE YOU LOAD ONLY THE CORRECT DISKETTES.

  Double-check that you have marked five diskettes from Release 2, Version 1 and one diskette from Release 2, Version 1.1 "DO NOT LOAD". Refer to paragraph 6.1.1.

- DO NOT REMOVE THE WRITE PROTECT TAPE FROM ANY DISKETTE.

Briefly, users must load the diskettes in the following sequence:

a. Insert the first Release 2, Version 1 object diskette.

b. Enter the appropriate load command. Load all Release 2, Version 1 diskettes as prompted by the system.

c. As each diskette is loaded, the number blocks transferred and any load diagnostics are saved in a message file. Read the message file to check that all diskettes loaded correctly. Remove the message file. If necessary, reload any miscopied Release 2, Version 1 diskettes. Check the message file to verify the second load. Remove the message file.

d. Insert the first Release 2, Version 1 Update diskette.

e. Follow the loading instructions contained in the Update Customer Letter.

f. Insert the first Release 2, Version 1.1 object diskette.

g. Enter the appropriate load command. Load the second object diskette as prompted by the system.

h. Read the message file to check that all diskettes loaded correctly. Remove the message file. If necessary, reload any miscopied diskettes. Check the message file to verify the second load. Remove the message file.

The initial loading of the complete object distribution requires approximately three hours. However, the process can be interrupted at any time. The procedure for interrupting and re-initiating the load follows the loading instructions, near the end of this step.

Notice there are two possible load commands:

/etc/loadtools/readflops.obj 2>> /message
or
/etc/loadtools/readflops 2>> /message

(There is no space in the expression **2>>**)

CAUTION
The readflops.obj COMMAND DESTROYS DISK PARTITION 5. IF ANY FILES HAVE BEEN INSTALLED ON PARTITION 5, USE THE readflops COMMAND INSTEAD.

The **readflops.obj** command uses the *dd*(1) utility to copy the contents of the diskette into partition 5. The *cpio*(1) utility copies the files from partition 5 into their proper place. **Readflops.obj** is recommended because it is significantly faster than **readflops**. However, any information contained in partition 5 is overwritten and lost when **readflops.obj** is executed.

The second script, **readflops**, also located in **/etc/loadtools**, loads diskettes without destroying partition 5.

LOADING INSTRUCTIONS

a. Re-read the precautions listed at the beginning of this step.

b. Insert the first Release 2, Version 1 object diskette, "1 of 31", into drive #2.

c. Enter the appropriate load command:

/etc/loadtools/readflops.obj 2>> /message
or
/etc/loadtools/readflops 2>> /message

(There is no space in the expression **2>>**)

d. For each diskette that is loaded (using either the **readflops.obj** or **readflops** script), the number of blocks transferred is contained in the file **/message**. If load diagnostics are

encountered, they are listed above the block count for the miscopied diskette. The first block count given in **/message** refers to diskette "1 of 31".

e. Load the 28 Release 2, Version 1 diskettes as prompted by the system.

f. Read the **/message** file to check the diskettes loaded correctly. Use the *ed*(1) utility to read the **/message** file. Enter the commands:

```
ed /message
1,10p
```

to print the first 10 lines in the file. If load diagnostics such as **"Cannot read** *filename***"** exist in the **/message** file, or the block count does not match the count provided in Table 6.1, the diskette was miscopied and must be loaded again.

To read the next 10 lines in the file **/message**, enter the commands:

```
10,20p
```

Continue reading the file, checking the block counts and looking for load diagnostics, until all diskettes have been accounted for. Make a note of each diskette that must be reloaded. Typically, **/message** contains about 80 lines for the Release 2, Version 1 distribution.

g. After the entire **/message** file has been examined to determine which diskettes must be reloaded, exit the editor and remove the **/message** file with the commands:

```
q
rm -f /message
```

h. Reload any miscopied Release 2, Version 1 diskettes. Place the miscopied diskette in drive #2 and enter the same load command you entered earlier. The diskettes need not be loaded in sequential order.

i. Examine the new file **/message** to ensure the diskette(s) loaded successfully. If the second load is unsuccessful, assume that the media itself is bad. Call Motorola/Four Phase Customer Support Organization to obtain new media.

j. After you have verified that all Release 2, Version 1 diskettes have been loaded properly, insert the first Release 2, Version 1 Update diskette. Instructions for installing the Update are contained in the Update Customer Letter. Follow the procedure described in the letter to verify that the Update has been loaded correctly.

k. Insert the first Release 2, Version 1.1 object diskette, "3 of 4", in drive #2. Enter the appropriate load command. Load "4 of 4" when the system prompts.

l. Verify the load by reading the **/message** file. Reload any miscopied diskettes following the procedures that were described for Release 2, Version 1 object diskettes.

INTERRUPTING AND RE-INITIATING THE LOAD

The loading process can be interrupted at any time. Finish copying the diskette you are loading. When prompted by the system for the next diskette, quit the load. Then, flush unwritten buffers out to the disk with the *sync*(1) command. The quit and "sync" is executed by entering the commands:

```
q
sync
sync
sync
```

You can now leave the system on, and return to the loading sequence later. To re-enter the loading sequence, insert the next diskette into disk drive #2 and enter the appropriate load command.

If you plan to power down the system, recall that the **/usr** filesystem is still mounted. Before you shut down, you must unmount the filesystem and clear out the buffers again. Remember that after you execute the third **sync** command in a series, you must wait until the disk drive light is no longer illuminated before you enter the next command. The complete procedure to discontinue the load and power down the system is as follows:

```
q
sync
sync
sync          (Wait for the drive light to turn off.)
umount    /dev/dsk/m320_0s1
sync
sync
sync          (Wait for the drive light to turn off.)
(Press the RESET button on the MVME050 module.)
(Power off.)
```

When you are ready to power up the system and begin the loading sequence again, remember to mount the **/usr** filesystem before you re-enter the **readflops.obj** (or **readflops**) command, if the filesystem was unmounted earlier. Enter the commands:

```
(Power up.)
bo
mount    /dev/dsk/m320_0s1    /usr
```

then insert the next diskette and re-enter the appropriate load command.

STEP 8: EDIT THE /etc/checklist AND /etc/rc FILES

Both the **/etc/checklist** and **/etc/rc** files must be edited.

a. Edit the file **/etc/rc** with the *ed*(1) utility. Enter the *ed* editor and search for the correct line with the commands:

```
ed /etc/rc
/mount/
```

b. If the first character is a # , remove the character to "uncomment" the line with the command:

```
s/^. //p
```

c. Next, change the line with the command:

```
s/cntrl_XsX/m320_0s1/p
```

d. Write the change and quit the editor with the commands:

w
q

e. Append the device names to the file **/etc/checklist** as follows:

echo /dev/dsk/m320_0s0 ≫ /etc/checklist
echo /dev/rdsk/m320_0s1 ≫ /etc/checklist

(**Note: The root filesystem should not be checked using the raw device designation. For a more detailed explanation, refer to** *fsck*(1M).)

f. Save these changes and update the fixed disk by executing the **sync** command three times. Enter:

sync
sync
sync

STEP 9: UPDATE KERNEL LIBRARIES

This step updates the kernel library files contained on the Release 2, Version 1 object diskettes. Kernels included on the Release 2, Version 1.1 diskettes are already upgraded. However, if a user intends to generate a custom kernel, the Release 2, Version 1 kernel libraries are needed, and therefore, must be updated.

The Release 2, Version 1 Update contains two scripts, **installio** and **installos**, which archive new object files into the kernel libraries. The Release 2, Version 1.1 kernel object files have been installed so that they will be included when the scripts are executed. For further information about the scripts, refer to the README file included on the Release 2, Version 1 Update. The scripts assume that the libraries, **lib1** and **lib2**, exist in the **/usr/src/uts/m68k/M68010** directory.

a. To update the kernel libraries, change directories to access the scripts:

cd /usr/src/uts/m68k/M68010

b. Execute the **installio** script with the command:

./installio

The script archives the new object modules into the kernel library, **lib2**, and places the old modules in the **/usr/src/uts/m68k/io** directory as *filename*.**o.old**. The script attempts to find two old object modules that correspond to two new modules, **lp0** and **lp050**. However, there are no such modules and two error messages will display:

ar: lp0.o not found
mv: cannot access lp0.o

ar: lp050.o not found
mv: cannot access lp050.o

Ignore these messages.

c. Execute the **installos** script with the command:

./installos

The script archives the new object modules into the kernel library, **lib1**, and places the

old modules in the **/usr/src/uts/m68k/os** directory as *filename*.**o.old**.

The kernel libraries are now updated with all changes applicable to Release 2, Version 1.1.

STEP 10: PERFORM ADMINISTRATIVE TASKS

Users who are planning to add the source distribution to their system should skip this step and proceed to Step 11. Return to this step when Step 13 is completed.

The system administrator who is installing object distribution only is advised to enter multi-user mode now with the command:

init 2

When the system prompts for "Console login", enter **root**. The **root** login ensures that you have permission to perform the following administrative duties. Take time now to perform the following tasks:

- Assign a password for **root**. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Create user ids and home directories. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Check that the environmental variables for the file **/etc/profile** are correct. On target systems, link **/etc/prfile.VME120** to **/etc/profile**. (The new values will not take effect until the system is rebooted.) Refer to paragraph 6.3.5 of this manual.
- Create and expand the **lost+found** directories in each mountable filesystem. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Change the nodename variable if the system is to be connected to a *uucp*(1C) network. Refer to the UNIX-to-UNIX CoPy (uucp) Tutorial included in the *SYSTEM V/68 User's Guide.*
- Create the *sxt*(7) devices to implement user job control capability. Refer to the *SYSTEM V/68 Administrator's Guide.*
- Edit the file **/etc/motd** to reflect that you are running Release 2, Version 1.1 (R2V1.1), not Release 2, Version 1 (R2V1). Refer to the *SYSTEM V/68 Administrator's Guide.*

STEP 11: SOURCE INSTALLATION

This installation procedure for source distribution assumes that the object distribution has been loaded according to Steps 1 through 9.

Partitions 5 and 6 have been reserved for the **/usr/src**. After the object files have been loaded, the first step in loading the source code is to create and mount the **usrsrc** filesystem on partitions 5 and 6. The commands that follow assume the **/usr** filesystem is mounted. If you have unmounted the filesystem, mount it with the command:

mount /dev/dsk/m320_0s1 /usr

a. After you have verified that the **/usr** filesystem is mounted, enter the commands:

```
mkfs /dev/dsk/m320_0s5   30696:5600
labelit /dev/rdsk/m320_0s5   usrsrc R2V1.1
mount /dev/dsk/m320_0s5   /usr/src
```

b. Backup all source diskettes onto newly formatted diskettes.

This part of the source installation is similar to Step 5 of the object installation. The source distribution contains 24 diskettes from Release 2, Version 1; at least one Release 2, Version 1 Update source diskette, and one Release 2, Version 1.1 source diskette. Format one backup diskette for each source diskette. Use the *dinit* format utility described in paragraph 6.3.4. Follow the commands presented in Step 5 to backup the source diskettes onto the newly formatted diskettes.

c. Insert the first source diskette from the Release 2, Version 1 distribution, "1 of 24", into drive #2. Copy the SYSTEM V/68 Release 2, Version 1 source diskette to the fixed disk with the command:

```
/etc/loadtools/readflops  /usr/src  2>>  /message
```

Notice that the command for copying the 24 Release 2, Version 1 source diskettes specifies **/usr/src** as the destination directory for the source code.

Continue loading the Release 2, Version 1 source diskettes as prompted by the system.

d. For each diskette that is loaded, the number of blocks transferred is contained in the file **/message**. If load diagnostics are encountered, they are listed above the block count for the miscopied diskette. The first block count given in **/message** refers to diskette "1 of 24". Read the **/message** file, following the procedure described in Step 7. Table 6-2 provides the block counts for the source diskettes.

Remove the **/message** file. If any diskette did not copy properly, reload it now.

e. Load the Release 2, Version 1 Update source diskette(s). The command that loads the Update diskette(s) and the Release 2, Version 1.1 source diskette is different from the command that copies the Version 1 source diskettes — it does not specify a destination directory:

```
/etc/loadtools/readflops  2>>  /message
```

Check the **/message** file against the block counts included in the Update Customer Letter to ensure the update diskette(s) loaded properly. The remaining diskettes must be loaded in order: Update source first, then Release 2, Version 1.1 source. Remove the **/message** file. If an update diskette did not copy properly, reload it now. The update source diskette(s) must be properly loaded before loading the Release 2, Version 1.1 source.

f. Load the Release 2, Version 1.1 source diskette; compare the block count in the **message** file against that in Table 6-2; remove the **/message** file.

If any diskette fails to load properly after two attempts, call Motorola/Four Phase Customer Support Organization to obtain new media.

STEP 12: UPDATE KERNEL LIBRARIES

Because the source installation mounted the **/usr/src** filesystem on top of the **/usr** filesystem, the library update performed in Step 9 is no longer accessible. Since custom kernels may be needed in the future, the kernel libraries should be updated at this point.

The kernel libraries can be updated in either one of two ways: either move the libraries generated in Step 9 to an accessible location; or generate new libraries from the source. Creating libraries from source will take several hours.

METHOD 1: MOVE ALREADY GENERATED LIBRARIES

a. Unmount the **/usr/src** filesystem to gain access to the libraries, which are located in the **/usr** filesystem.

```
cd /
umount /dev/dsk/m320_0s5
```

b. Move the libraries to a temporary, and accessible, location: the root directory.

```
mv /usr/src/uts/m68k/M68010/lib*
```

(The designation **lib*** moves all files that begin with the letters "lib".)

c. Mount the **/usr/src** filesystem and install the libraries.

```
mount /dev/dsk/m320_0s5  /usr/src
mv  /lib*  /usr/src/uts/m68k/M68010
```

METHOD 2: CREATE THE LIBRARIES FROM SOURCE

This method will require several hours.

a. To update the kernel libraries, change directories to access the makefile:

```
cd /usr/src/uts/m68k
```

b. Execute the makefile, specifying the target that will remake the libraries:

```
make lib010
```

When the compilations and archiving have completed, the kernel libraries will be upgraded and may be used to generate new kernels. (Refer to paragraph 6.3.6 for further information about generating a kernel.)

STEP 13: EDIT THE /etc/checklist AND /etc/rc FILES
(For Source Distribution)

a. Edit the file **/etc/rc** with the *ed*(1) utility:

```
ed /etc/rc
/mount/
```

b. Add a line of code to mount the **/usr/src** filesystem when the system is brought up in multi-user mode:

```
a (CR)
mount  /dev/dsk/m320_0s5     /usr/src (CTRL)I #mount /usr/src filesystem
```

where (CTRL)I generates a tab character.

c. Write the change and quit editor with the commands:

```
w
q
```

d. Append the new device name to the file **/etc/checklist**:

```
echo /dev/rdsk/m320_0s5 >> /etc/checklist
```

e. Save all changes and update the fixed disk by entering the **sync** command three times. Enter:

```
sync
sync
sync
```

Now return to Step 10 and perform the recommended administrative tasks to complete the installation.

6.3 Administrative Procedures

The following paragraphs describe some of the procedures administrators will perform during the course of their work.

6.3.1 Initialize Winchester Disk With Minload Script. This paragraph describes the initialization procedure for a 40Mb Winchester disk using the **/etc/loadtools/minload** script contained on the Release 2, Version 1.1 minimum system diskette, "2 of 4".

Users need to know the manufacturer of the disk supplied with their systems (either Micropolis or Vertex), as well as the location of any imperfections on their disks. The required information can be found on the Winchester verification report included with the disk drive. The Winchester media cannot be initialized without this information.

When you are ready to format and initialize the Winchester disk, follow these steps:

1. Power up the system.
2. The MVME12xbug performs an automatic self-test. It tests the processor and on-board RAM and ROM memory. If any test fails, an error message is displayed. Next, the MVME12xbug prompt displays the question: "Size and Initialize RAM (Y/N) ?" Enter **y** to initialize the system. (Later versions of MVME12xbug may not prompt with this question.)

   The system displays the message:

   ```
   COLD Start
   MVME12x 2.r
   ```

   where *r* is the revision number of the debug monitor.
3. REMOVE THE WRITE PROTECT TAPE from the minload diskette labeled "2 of 4" and place it in the device number 2 drive. Boot from the diskette with the command:

   ```
   bo 2
   ```

The following messages appear on the terminal screen:

```
M68010/sysV68:  unixR2V1.1
real memory = n
available memory = n

INIT:  SINGLE USER MODE
```

If the Winchester drive has not yet been formatted, or if the configuration information on the disk has been destroyed, the following two error messages appear on the screen:

```
M320: drive 0, ctl 0, cmd 5, main 1, ext 0x8, i/o error

M320: drive 0, ctl 0, open found no format/config
```

Both messages indicate the Winchester drive is not formatted properly. The messages may be ignored for now.

4. Enter the command:

   /etc/loadtools/minload

   The script prompts for information about the disk manufacturer and any imperfections listed on the Winchester disk verification report. If imperfections are listed on the verification report, you must perform some calculations to convert the information on the report into a form recognized by the utility.

   Refer to paragraph 6.3.2 for instructions on converting the information on the verification report into a list of bad tracks.

   Enter the bad tracks when prompted by the system. The interactive mode is terminated by typing a period (.). The period takes the system out of the bad track handling phase of disk initialization. If no defects are listed on the verification report, type a period when first prompted for a bad track. The system will assume the device is perfect.

   After the bad block information is accepted, approximately 50 filenames are listed on the screen as the minimum system utilities are copied from the minload diskette to the disk. When the routine finishes, the screen should indicate 1113 blocks have been copied. If some other number appears, repeat the procedure.

6.3.2 Calculate Bad Tracks from Winchester Verification Report. The Winchester disk can be initialized in two ways. The **minload** script contained on the Release 2, Version 1.1. minload diskette is the first, the *dinit*(1M) utility is the second. Both methods incorporate bad track handling on the MVME320 controller.

Whether the initialization is begun using the **minload** script or the *dinit* utility, the system prompts for a list of bad tracks. However, the information on the Winchester verification report does not identify the disk imperfections in terms of bad tracks. Instead, the report lists media imperfections in one of two ways: either the report gives the sector number of the first bad sector on a track, or the report identifies the problem area by head number, cylinder number, and byte offset.

To calculate the bad tracks from the information provided on the verification report, use one of the following methods, whichever is appropriate to your disk:

METHOD 1: Calculate Bad Tracks from Sector Numbers

To obtain the bad track numbers, divide each sector number listed in the Winchester verification report by the number of sectors per track. Since all supported drives (Computer Memories, Micropolis, and Vertex) contain 32 sectors per track, the conversion equation becomes:

track number = (sector number) / 32

METHOD 2: Calculate Bad Tracks from Head and Cylinder Numbers

[(cylinder number) x (total # of heads)] + (head number) = track number

The 40Mb Micropolis drive has six heads; the Vertex 40Mb drive has five.

6.3.3 Initializing the Winchester Disk with dinit. This section describes the initialization procedure for 40Mb Winchester disks using the *dinit*(1M) utility. This procedure does not copy the Release 2, Version 1.1 minimum system onto the disk.

Users will need to know the manufacturer of the disk supplied with their systems (either Micropolis or Vertex), as well as the location of any imperfections on their disks. The required information can be found on the Winchester verification report included with the disk drive. The Winchester media cannot be initialized without this information.

When you are ready to format and initialize the Winchester disk, follow these steps:

1. Power up the system.
2. The MVME12xbug performs an automatic self-test. It tests the processor and on-board RAM and ROM memory. If any test fails, an error message is displayed. Next, the MVME12xbug prompt displays the question: "Size and Initialize RAM (Y/N) ?" Enter **y** to initialize the system. (Later versions of MVME12xbug may not prompt with this question.)
3. The system displays the message:

   COLD Start
   MVME12x 2.*r*

   where *r* is the revision number of the debug monitor.

   REMOVE THE WRITE PROTECT TAPE from the minload diskette labeled "2 of 4" and place it in the device number 2 drive. Boot from the diskette with the command:

   bo 2

4. Wait for the SYSTEM V/68 prompt # to appear. Enter the command:

   dinit -f -o -b /etc/loadtools/vmeboot *drive* **/dev/rdsk/m320_*x*s7**

   where *drive* refers to:

m32040	for a 40Mb Micropolis Winchester drive
m32040v	for a 40Mb Vertex Winchester drive

   and *x* refers to the appropriate drive number.

   The *dinit*(1M) utility invokes *m320fmt*(1M), a disk formatter for MVME320 devices. *Dinit* enters an interactive mode and prompts the user for bad track entries. Check the Winchester verification report supplied by the disk manufacturer for a list of bad blocks or imperfections on the disk. If no bad blocks are listed, type a period (.) to terminate the bad track handling phase of disk initialization. The device is assumed to be perfect.

If imperfections are listed on the verification report, you must perform some calculations to convert the information on the report into a form recognized by the utility. Refer to the calculations described in paragraph 6.3.2 for instructions on converting the information on the verification report into a form acceptable to the *dinit* utility. The calculation information is also provided on the manual page for *m320fmt*(1M) included in Chapter 8 of this manual.

6.3.4 Format Diskettes. Only one type of 5¼-inch diskette can be formatted, a 5¼-inch double-sided, double density diskette.

To format a 5¼-inch diskette, remove the write-protect tab from the diskette and insert the diskette into the drive. Enter the command:

dinit —f —o m320dsdd5 /dev/rdsk/m320_*X*s7

where *X* specifies the correct drive.

A bootable diskette may be created by adding the **—b** option and specifying the bootloader, **/stand/m68k/boots/vmeboot**. When the **—b** option is invoked, a filesystem must be created on slice 0.

To make a filesystem on the diskette, invoke *mkfs*(1M) on the diskette's block device entry, specifying either 1264 or 1276 as the blocks argument:

6

Slice	Blocks Argument	Description
0	1264	Identical to slice 2 on VM22; slice 0 on VME/10; and slice 0 on a MVME319.
7	1276	Entire diskette contents

6.3.5 Set SYSTEM V/68 Variables. The system environment contains four exported variables:

- MMU
- PROCESSOR
- STACKCHECK
- SYSTEM

The object distribution file **/etc/profile** contains the appropriate values for each variable for each system under Release 2, Version 1 and Version 1.1. For information about **FP**, the floating point variable, refer to *cc*(1). The defined values for each variable are as follows:

VARIABLE	VALUE
MMU	M68451
PROCESSOR	M68010
STACKCHECK	OFF
SYSTEM	VME120

A version of **/etc/profile** that is appropriate for an MVME121-based system is contained in the file **/etc/prfile.VME120**. Link the file to **/etc/profile** to generate the correct SYSTEM V/68 variables. The new values do not take effect until the system has been shut down and re-booted.

6.3.6 Generate a Kernel. Information about the kernel and kernel values may be found under the following manual entries: *config*(1M), *dfile*(4), *make*(1), *master*(4), and *sysdef*(1M).

If it is necessary to generate a new kernel, enter the commands:

```
cd /usr/src/uts/m68k
make vme120_x
```

where *x* is replaced with:

2 to generate a 40Mb system, MVME320 kernel

The new kernel is the file **VME120_2unix**, which is located in the directory **/usr/src/uts/m68k/M68010**.

Move the newly created kernel to the root directory for testing with the commands:

```
cp M68010/VME120_2unix /unix.test
chmod 0664 /unix.test
```

Change directories to root, and if the **usr** and/or **/usr/src** filesystems are still mounted, unmount them with the commands:

```
cd /
umount /dev/dsk/m320_0s5
umount /dev/dsk/m320_0s1
```

Use the *sync*(1) utility to flush unwritten system buffers out to disk and save all changes. Enter the command three times:

```
sync
sync
sync
```

Wait until the drive light is no longer illuminated, then press the RESET button on the MVME050 board.

Wait for the MVME12xbug prompt and bootstrap the system with the command:

```
bo „unix.test
```

Test to be sure the kernel is working properly.

After the kernel has been tested and proven, rename it (e.g., **new320unix**). When you are ready to make the new kernel the default booting kernel, link it to **/unix** and to **/stand/unix**, and do a "sync". (As an extra precaution, some administrators may want to copy the old kernel, **/unix**, to **/unix.old** before linking. Once the new kernel is linked to **/unix** and **/stand/unixi**, the old kernel will be lost.)

The command sequence is:

```
cd /
mv unix.test new_kernel
ln new_kernel unix
ln new_kernel stand/unix
sync
sync
sync
```

Wait until the drive light is no longer illuminated. Press the RESET button on the MVME050 board and boot from the new kernel with the command:

```
bo
```

The following table lists the names of the system kernels and the new kernels that result from the *make*(1) command.

KERNEL	KERNEL PRODUCED BY MAKE	DESCRIPTION
vme120_1	/usr/src/uts/m68k/M68010/VME120_1unix	15Mb system, MVME320
vme120_2	/usr/src/uts/m68k/M68010/VME120_2unix	40Mb system, MVME320
vme120_3	/usr/src/uts/m68k/M68010/VME120_3unix	40Mb system, MVME319

(**NOTE:** This release does not support a 15Mb system.)

6.3.7 Configuration Planning. The following table lists the values of the basic parameters for the MVME121-based system. For more details of syntax and structure, refer to *config.68*(1), *dfile*(4), and the associated *master*(4).

ITEM	VALUE
nswap	4752
buffers	50
hashbuf	64
physbuf	4
inodes	90
files	90
mounts	8
coremap	100
swapmap	75
calls	50
procs	50
texts	40
clists	150
maxproc	25

SYSTEM V/68 Release 2, Version 1.1 supports cache with the inclusion of a new cache variable in the **master** and **dfile** files (refer to *master*(4) and *dfile*(4)). The variable is a cache descriptor derived from "OR-ing" applicable flags that describe the cache. The following table describes the various cache attributes.

Attribute	Descriptor Value	Description
LOGICAL	01	Cache is logical addresses.
PHYSICAL	02	Cache is physical addresses.
SUPERVISOR	04	Cache is supervisor.
USER	010	Cache is user.
DATA	020	Cache contains data.
INSTRUCTION	040	Cache contains instruction.

The attributes of the system cache are logical and instruction for both supervisor and user. The value of the cache flag in **master** is set to 055.

NOTES

7. ERROR MESSAGES FOR THE MVME320

Error messages for the MVME320 disk controller and suggested actions are summarized in the following listing. Refer to the *MVME320 VMEbus Disk Controller User's Manual* for additional information about these messages and the error codes and types. Refer also to the *m320fmt*(1M) and *mvme320*(7) entries in the SYSTEM V/68 documentation.

M320: drive #*n*, ctl #*n*, bad track data exceeds # buffer

DESCRIPTION: Open processing for the indicated drive and controller was unable to read the entire bad track table into a system buffer. This error also generates the "open found no format/config" message. (Refer to the message that follows.) The bad track subsystem generates a "Bad table overflow" message when there is insufficient space available in its storage pool.

M320: drive #*n*, ctl #*n*, open found no format/config

DESCRIPTION: Device open was unable to read the disk, or failed to find the required volume label information on the disk. This message appears only for hard disks.

SUGGESTED ACTION: Format the disk. All other operations return an I/O error. Refer to *dinit*(1M) for information describing the formatting operation.

M320: invalid special command (2) #

DESCRIPTION: An internal error exists in the MVME320 disk controller driver. Contact your support organization. Your driver may have been modified incorrectly.

M320: ctl #, spurious interrupt

DESCRIPTION: An unexpected interrupt has been received from the indicated MVME320 disk controller. If the message appears immediately after bootload, it may be ignored. If the message continues to appear, the controller may be malfunctioning, or the boards may be jumpered incorrectly. Contact the Motorola/Four Phase Customer Support Organization.

M320: ctl #*n*, interrupt when not (logically) busy

DESCRIPTION: An interrupt from the indicated controller has been received but the driver software has no record of an outstanding request. This error could result from either a malfunctioning controller, or an error in the driver.

M320: drive #*n*, ctl #*n*, mask 0x#, interrupt not on active unit

DESCRIPTION: An interrupt from the indicated controller shows request completion for some unit other than the expected drive. The mask is the hexadecimal value of controller register B. This error could result from either a malfunctioning controller, or an error in the driver. Refer to the *MVME320 VMEbus Disk Controller User's Manual.*

M320: drive #*n*, ctl #*n*, mask 0x#, extra interrupt

DESCRIPTION: An interrupt from the indicated controller shows request completion for some other unit, in addition to the expected drive. The mask is the hexadecimal value of controller register B. This error could result from either a malfunctioning controller or an error in the driver. Refer to the *MVME320 VMEbus Disk Controller User's Manual* for more information.

M320: drive #*n*, ctl #*n*, cmd #*n*, main #*n*, ext 0x#, i/o error

DESCRIPTION: This message describes an I/O error where drive #*n* and ctl #*n* refer to the MVME320 controller and drive where the error occurred, cmd #*n* defines the controller command code of the failed operation, main #*n* defines the the main status as a number and ext is the extended status expressed as a hex bit mask. Refer to *MVME320 VMEbus Disk Controller User's Manual* for list of definitions for the possible main status and extended status numbers.

NOTE: "Sector id not found" (mask bit 0x08) may be erroneously reported when the actual source of the problem cannot be determined.

M320: dump device 0x# not available

DESCRIPTION: The VME12xbug crash routine cannot execute because damage to the system is so severe that the driver software cannot function properly. Reboot the system, or contact your support organization.

7

M320: dump truncated at end of logical device
#*n* blocks written

DESCRIPTION: Space on the dump device is exhausted. Refer the problem to your system administrator.

M320: dump I/O error, terminating

DESCRIPTION: I/O on the dump device has failed after five retries. Reboot the system.

M320: controller # timed-out, disabled

DESCRIPTION: The indicated controller started a request more than five seconds ago and has not yet generated a completion interrupt. As a result, the MVME320 driver has set internal flags that make it impossible for new requests to reach the controller. The error message reflects a serious controller malfunction. Reboot the system.

M320: controllers #*n* - #*n*, csr address conflict

DESCRIPTION: Device driver initialization has detected a fatal conflict in the assignment of control and status register (csr) addresses for the two controllers indicated. Both are unusable. The SYSTEM V/68 configuration, the MVME320 board strappings, or both are incorrect. The csr address is set in Jumper JW6 on the board. Refer to the *SYSTEM V/68 Administrator's Manual* and the *MVME320 VMEbus Disk Controller User's Manual.*

8. MANUAL PAGES

This section contains new and revised pages for the *SYSTEM V/68 Administrator's Manual* and *SYSTEM V/68 User's Manual.* The following pages are provided:

a. *dinit*(1M) — disk initializer

b. *idfmt*(1M) — disk formatter for the MVME319 disk controller

c. *m320fmt*(1M) — format disks on the MVME320 disk controller

d. *rewind*(1M) — program for rewinding and reading bad segment table on floppy tape

e. *badlist*(4) — media defect tables for MVME319 Winchester devices

f. *ftape*(7) — CIPHER DATA PRODUCTS CT525 FloppyTape

g. *id*(7) — general disk driver for the MVME319 disk controller

h. *lp050*(7) — MVME050 line printer interface

i. *m050*(7) — MVME050 serial port

j. *m320*(7) — general disk driver for the MVME320 disk controller

k. *mfp*(7) — serial port on the MVME121-core system

l. *mvme050*(7) — driver for the MVME050 controller

m. *sa400flwd*(7) — 5¼-inch floppy disk drive for the Winchester Disk Driver and the general disk driver for the MVME319 and MVME320 Disk Controllers

n. *sa800flid*(7) — 8-inch floppy disk drive for MVME319 disk controller

o. *wd15*(7) — 15Mb Winchester Disk Drive

p. *wd40*(7) — 40Mb Winchester Disk Drive

DINIT (1M)

NAME

dinit — disk initializer

SYNOPSIS

/etc/dinit [**—foce**] [**—d** desc] [**—b** file] [**—t** file] type rdev

DESCRIPTION

Dinit can be used to initialize specified disk *types*. The *type* must be one from the file **/etc/diskdefs**. Current values are shown in the following tables:

VM21 CONTROLLER

Drive Name	*type* Value
50Mb Lark Module Drive	vm21lark25
16Mb Lark Module Drive	vm21lark8
80Mb Cartridge Module Drive	vm21cmd80
16Mb Cartridge Module Drive	vm21cmd16
Double-sided 8" Floppy Diskette	vm21dssd8
Single-sided 8" Floppy Diskette	vm21sssd8

VM22 CONTROLLER

Drive Name	*type* Value
Removable 25Mb Lark	vm22R25L
Fixed 25Mb Lark	vm22F25L
Removable 8Mb Lark	vm22R8L
Fixed 8Mb Lark	vm22F8L
Removable 16Mb Cmd	vm22R16C
Fixed 16Mb CMD	vm22F16C
Fixed 80Mb CMD	vm22F80C
*Double-sided Double Density 8" Floppy	vm22dsdd8
*Single-sided Double Density 8" Floppy	vm22ssdd8
*Double-sided Single Density 8" Floppy	vm22dssd8
*Single-sided Single Density 8" Floppy	vm22sssd8
**Double-sided Double Density 5¼" Floppy	vm22dsdd5
**Single-sided Double Density 5¼" Floppy	vm22ssdd5
**Double-sided Single Density 5¼" Floppy	vm22dssd5
**Single-sided Single Density 5¼" Floppy	vm22sssd5

* Motorola 8" Format

** IBM 5¼" Format

MVME319 CONTROLLER (ID CONTROLLER)

Drive Name	*type* Value
40Mb Micropolis Winchester	idwm40
**Double-sided Double Density 5¼" Floppy	iddsdd5
*Double-sided Single Density 8" Floppy	iddssd8
*Cipher Data Products CT525 FloppyTape	idftape

* Motorola 8" Format

** IBM 5¼" Format

MVME320 CONTROLLER

Drive Name	*type* Value
40Mb Vertex Winchester	m32040v
15Mb Computer Memories Winchester	m32015
40Mb Micropolis Winchester	m32040
**Double-sided Double Density 5¼" Floppy	m320dsdd5

** IBM 5¼" Format

DINIT (1M)

For disk types with software bad track handling, the alternate track numbers will be taken from the file **/etc/diskalts/***type*, where *type* is the type name given in **/etc/diskdefs.** If no file **/etc/diskalts/***type* exists, the user will be prompted to enter the alternate track numbers interactively. There is no software bad track support for floppy diskettes.

The *rdev* argument specifies a raw device, which must be of the form **/dev/r***string*. There must be a corresponding block device **/dev/***string* with the same minor device number as the character device. *Dinit* must be executed over slice 7 of the raw device.

The following options are provided for *dinit*:

—f Format disk. When formatting an unformatted disk, two read errors appear on the screen. These errors occur because the controller is trying to read configuration information from the disk. The messages can be ignored; the disk will be formatted as requested.

—o Override disk contents, including type and bad tracks.

—c Check for bad tracks.

—e Use **EXORMACS** instead of **MOTOROLA** in sector 0, for compatibility with VM03 and EXORmacs bootloaders.

—d *desc* Use *desc* as description string in sector 0.

—b *file* Use *file* (a.out format) as the bootloader program.

—t *file* Take bad track numbers from *file*, instead of interactively.

Unless the **—f** or **—o** options are given, *dinit* will examine the disk to ensure the disk type is not being changed. Also, it will read the previous bad track list. Therefore, it is not necessary to re-enter bad track numbers on subsequent use of *dinit* on a disk. This is useful for changing the bootloader, description string, etc. (For calculations of bad track numbers, refer to the specific format utility, e.g., *m320fmt*(1M).)

Whenever new bad tracks are given to *dinit*, the layout of the areas of the disk provided for file systems may be arbitrarily remapped. Therefore, all useful information from the disk should be copied to backup media before adding bad tracks and then copied back when *dinit* has finished.

FILES

/etc/diskdefs	disk definition file
/etc/diskalts/*	alternate track numbers

SEE ALSO

m320fmt(1M)

"Setting Up SYSTEM V/68" in the *SYSTEM V/68 Administrator's Guide.*

IDFMT (1M)

NAME

idfmt — format disks on the MVME319 disk controller

SYNOPSIS

idfmt —**w** [—**p** —**v**] special
idfmt —**t** special
idfmt —**5** [—**v**] special
idfmt —**8** [—**v**] special

DESCRIPTION

The *idfmt* utility is used to format disks or floppy tapes on the MVME319 disk controller. Floppy disks are formatted track by track, while Winchester and tape devices are formatted always as a whole. This causes the MVME319 controller to be busy until a "format Winchester" or a "format tape" request has been terminated. Therefore, it is recommended that *idfmt* be invoked only on a quiet system (i.e., single-user mode) when formatting those devices. An additional reason for this is that the effective user-id must be 0 (superuser) to format Winchester and tape devices.

The following options are available for *idfmt*:

—**w** Format Winchester. All Winchester drives for which the system is configured can be formatted. Bad block handling is done by hardware. A media defect list for each Winchester device must be transmitted to the MVME319 during a format session. As long as the option —**p** is not specified, *idfmt* tries to read the file **/etc/badlist** and looks for bad block lockout entries. If this file cannot be opened or there is no media defect list for the device in **/etc/badlist**, *idfmt* switches to an interactive mode and asks for lockout entries. If no input is made, the device is assumed to be perfect.

—**t** Format Floppy Tape. Only 3M DC600A tape cartridges or equivalent can be formatted. Bad segment handling is done by the MVME319. A bad segment table is created by verifying the tape after it has been formatted.

—**5** Format 5¼-inch Floppy. Only 5¼-inch double-sided, double density, IBM format floppies can be produced. The media must be perfect.

—**8** Format 8-inch Floppy. Only 8-inch double-sided, single density, Motorola Inc. format floppies can be produced. The media must be perfect.

—**p** Perfect Winchester. The Winchester device is assumed to be perfect. *Idfmt* does not look for **/etc/badlist** and no bad block handling is done after the disk has been formatted.

—**v** Verify device. Only Winchester or floppy devices can be verified. When formatting tape, a verify is done by the MVME319 itself. It is recommended that this option be used when formatting Winchester or floppy devices.

FILES

/etc/badlist

SEE ALSO

dinit(1M), id(7), badlist(4).

NAME

m320fmt — format disks on the MVME320 disk controller

SYNOPSIS

m320fmt [**hard_disk_enable** —**h** heads —**c** cylinders] rawdev

DESCRIPTION

The *m320fmt* utility is used to format disks on the MVME320 controller. Winchester ("hard") disks are formatted in a continuous operation which will keep the controller busy until it completes. Floppy disks are formatted track-by-track, permitting other I/O operations to intervene. Support for bad track handling on the MVME320 controller is done in software only. Therefore the *m320fmt* utility should be used only for diskettes or for media that has no defects listed on the Winchester verification report. To format any media that contains imperfections or that is to be booted, use the *dinit*(1M) utility. Refer to *dinit*(1M). To perform the calculations needed to enter the media imperfections with the *dinit* utility, use the conversion procedures described below.

The following options are available:

hard_disk_enable The string **hard_disk_enable** must appear as shown to enable formatting of a hard disk. It may not appear if the target disk is a floppy.

—**h** The number of heads (surfaces per cylinder) on the target hard disk.

—**c** The number of cylinders on the target hard disk.

rawdev must be a raw device defined on the target unit (/dev/rdsk/m320_...) . The slice number is irrelevant for hard disks, but must be one which spans the entire volume for floppy disks. By convention, slice 7 is thus defined. *M320fmt* generates a warning message if it finds a floppy slice other than 7.

The *dinit*(1M) utility invokes *m320fmt*(1M), a disk formatter for MVME320 devices. *Dinit* enters an interactive mode and prompts the user for bad track entries. Check the Winchester verification report supplied by the disk manufacturer for a list of bad blocks or imperfections on the disk. If no bad blocks are listed, type a period (.) to terminate the bad track handling phase of disk initialization. The device is assumed to be perfect.

If imperfections are listed on the verification report, you must perform some calculations to convert the information on the report into a form recognized by the utility. The *dinit* utility expects a list of bad tracks. The Winchester verification report lists media imperfections in one of two ways: either the report gives the sector number of the first bad sector on a track, or the report identifies the problem area by head number, cylinder number, and byte offset. To calculate the bad tracks from the information provided on the verification report, use one of the following methods, whichever is appropriate to your disk:

METHOD 1: Calculate Bad Tracks from Sector Numbers

To obtain the bad track numbers, divide each sector number listed in the Winchester verification report by the number of sectors per track. Since all supported drives (Computer Memories, Micropolis, and Vertex) contain 32 sectors per track, the conversion equation becomes:

track number = (sector number) / 32

METHOD 2: Calculate Bad Tracks from Head and Cylinder Numbers

(cylinder number) x (total # of heads) + (head number) = track number

The 15Mb Computer Memories drive and the 40Mb Micropolis drive have 6 heads. The Vertex 40Mb drive has 5 heads.

M320FMT (1M)

After *m320fmt* has formatted the device, *dinit* sets up the volume-id and the configuration sectors, records the bad track information, and installs the boot loader on the drive.

(**NOTE:** If a Winchester disk is formatted without making the required bad track entries, proper operation cannot be guaranteed.)

BUGS

An error in specifying heads or cylinders for a hard disk may result in a disk which appears to be correctly formatted but generates physical I/O errors in high cylinders (bad precomp values) or seems to have "lost" some of its space (surface mapped out). *Dinit*(1M) references a Motorola-prepared file which contains accurate values for these parameters.

SEE ALSO

dinit(1M), m320(7).

NAME

rewind - rewinds and reads bad segment table on the CIPHER DATA PRODUCTS CT525 FloppyTape

SYNOPSIS

rewind special

DESCRIPTION

The *rewind* command is used to rewind the CIPHER DATA PRODUCTS CT525 FloppyTape on the MVME319 and to read the bad segment list at block 3 of the tape.

Whenever already formatted tape cartridges are inserted into the floppy tape, *rewind* has to be applied to the device (special). If this is neglected, proper read/write operations cannot be guaranteed.

The argument *special* must specify partition 7 of a raw device.

SEE ALSO

id(7), idfmt(1m), ftape(7).

NAME

badlist — contains media defect tables for MVME319 Winchester devices

DESCRIPTION

The file **badlist** resides in the directory **/etc** and contains media defect tables for MVME319 Winchester devices. The media defect tables are passed to the disk controller during an *idfmt*(1m) format session. Once this has been done, the controller performs bad sector replacement. It is recommended that an appropriate table be created within this file for each Winchester Drive connected to a system.

Each media defect table must be headed by the '#' character (must be at column 0) and the unit number of the device, both separated by a space character. The table entries must be ordered by cylinder, head, and byte offset. (Note: The byte offset must be entered as a single number.) These data can be taken from the Winchester Drive vendor's drive verification list. An asterisk '*' marks a comment line. Comment lines are not allowed within a media defect list for a device.

Examples:

```
* order like this:          * cylinder head [byte][offset]          *
* the next four lines are for unit 0
# 0
376 6 13245
111 2 103
829 3 5643
* the next two lines are for unit 2
# 2
12 5 12312
* the next three lines are for unit 8 (which is unit 0 on controller 1)
# 8
765 3 1234
565 1 234
```

SEE ALSO

id(7), idfmt(1m).

NAME

ftape — CIPHER DATA PRODUCTS CT525 FloppyTape

DESCRIPTION

The MVME319 disk controller provides the ability to interface to a CIPHER CT525 FloppyTape. Since the data organization on the tape is quite similar to that on a disk device, it is served by the *id*(7) disk driver. Data blocks are organized within "segments". A segment consists of 32 physical blocks (16Kb) and might be compared to a disk track.

Although it is possible to have 8 partitions for each floppy tape, only two are defined. Partition 7 represents the whole tape, partition 0 all the usable space. The segments that are not included in partition 0 are used to keep a bad segment table and replacement segments for defective ones.

The size and origin of the partitions are as follows:

PARTITION	SIZE	START
0	46848	32
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	47040	0

For most efficient operation, it is recommended that the floppy tape be used in the raw I/O mode whenever possible and that read/write requests be aligned to segment boundaries (i.e., 16Kb blocks).

FILES

/dev/dsk/*, /dev/rdsk/*

SEE ALSO

id(7), idfmt(7), rewind(7).

NAME

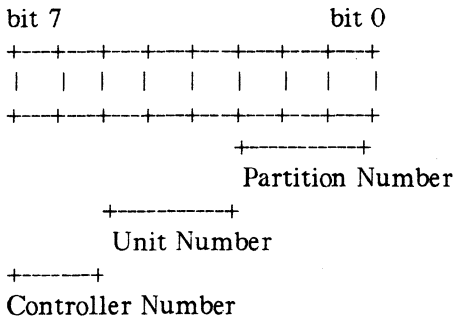
id — general disk driver for the MVME319

DESCRIPTION

Id provides a general interface to the MVME319 intelligent disk controller. The driver provides support for Winchester disks through the MVME319's SCSI interface and the ADAPTEC ACB4000 SCSI controller. In addition, *id* supports 8-inch and 5¼-inch floppy disks and the CIPHER DATA PRODUCTS CT525 FloppyTape.

Drive-dependent parameter initialization tables and drive partitioning tables must be selected when the system is configured. This should be done by modifying the two structure arrays *id_sizes* and *id_phys*, which can be found in the file **include/sys/io/idio.h**.

Id interprets the Minor Device Number as follows:



- Partition or Slice Number defines one of 8 disk partitions (see id_sizes)
- Unit Number defines one of 8 logical units. Units 0 to 3 specify only Winchester devices. Units 0 and 1 are connected to SCSI controller #0 and Units 2 and 3 are connected to SCSI controller #1. Units 4 and 5 specify either floppy tape or 5¼-inch floppy diskette devices. When a floppy tape is in a system configuration, Unit 5 is not usable for any other devices. Units 6 and 7 specify 8-inch or 5¼-inch floppy diskette devices.
- Controller Number defines one of two MVME319 controllers.

The *ioctl*(2) system calls use the following form:

#include <sys/id.h>
ioctl (*fildes, command, arg*)

Two commands are available:

FMT_TRK Format one disk track. *Arg* specifies the physical block number of the first block on the track that is to be formatted.

FMT_DSK Format a whole device. *Arg* is a pointer to a media defect list (declared as **struct badtable**). The MVME319 fetches this list by direct memory access from a user program. The user process is locked in core until the format request has been terminated. For perfect devices or floppy tapes, *arg* must be 0.

```
struct badtable {
        unsigned short      fill;
        unsigned short      length;
        struct {
                unsigned int  cyl_head;
                /* head    is in byte 0    */
                /* cylinder is in bytes 1 to 3 */
                unsigned int  byte;
                /* byteoffset from index */
        } defect[MAXDEFECTS];
};
```

NOTE:

- **FMT_TRK** can be only applied to floppy devices.
- **FMT_DSK** can be applied to either Winchesters or floppy tape devices. When formatting floppy tape devices, *arg* must be 0.

Since up to 127 sectors of a Winchester disk might be used by bad block handling, they should be not included when constructing partition tables for special disk devices.

FILES

/dev/dsk/*, /dev/rdsk/*
<sys/id.h>, <sys/io/idio.h>, <sys/io/win.h> <sys/io/sa400.h>, <sys/io/sa800.h>, <sys/space/idspace.h>

SEE ALSO

config(1M), dinit(1M), idfmt(1M), rewind(1M), ioctl(2), master(4), badlist(4), ftape(7), sa400f1wd(7), sa800flid(7).

NAME

lp050 — MVME050 line printer interface

DESCRIPTION

SYSTEM V/68 supports the parallel port on the MVME050 System Controller Module as the printer port. *Lp050* provides the interface to any of the standard Printronix-type parallel line printers or the standard Centronics-type parallel line printers. When the device is opened or closed, a suitable number of page ejects are generated. Bytes written are printed using a buffered interface.

The driver supports the printable ASCII character set (96 characters) and correctly interprets carriage returns, backspaces, tabs, and form-feeds. The defaults for line length, indent, and lines per page are 132, 4, and 66, respectively. Lines longer than the line length minus the indent (i.e., 128 characters, using the above defaults) are truncated. These defaults can be accessed and modified with an external program using the following calls.

The two *ioctl*(2) system calls available are of the following form:

#include <sys/lprio.h>
ioctl(*fildes, command, arg***)**
struct lprio ****arg*;**

The commands are:

LPRGET Get the current indent, columns per line, and lines per page and store in the **lprio** structure referenced by *arg*.

LPRSET Set the current indent, columns per line, and lines per page from the structure referenced by *arg*.

This driver does not support unbuffered parallel-to-serial-port conversion utilizing the parallel port on the MVME050 module.

FILES

/dev/lp*

SEE ALSO

lp(1), ioctl(2), mvme050(7), m050(7).
MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual.

NAME

m050 — MVME050 Serial Port

DESCRIPTION

Each MVME050 board supports two devices. Each line attached to an *m050* behaves as described in *termio*(7). The line speed of each device can be changed under software control (output speed = input speed). The number of data bits (5, 6, 7, or 8), parity (even, odd, or no), and the number of stop bits (1 or 2) are also software selectable (refer to *stty*(1)).

FILES

/dev/mpcc0, /dev/mpcc1

SEE ALSO

stty(1), termio(7).

NAME

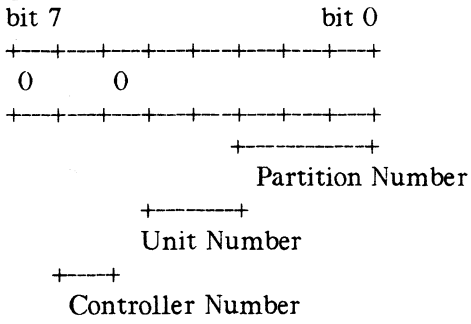
m320 — general disk driver for the MVME320

DESCRIPTION

The *m320* driver provides support for Winchester Disks and 5¼" IBM format floppy disks.

Winchester support includes a provision for mapping bad tracks to alternates on the disk. The *dinit*(1M) program must be used to record bad track information for the use of the driver.

The driver interprets the minor device number as follows:



The controller number defines one of two MVME320 controllers, 0 or 1.

The unit number defines one of four units on the controller. These assignments are fixed as follows:

unit 0 Winchester Disk

unit 1 Winchester Disk

unit 2 5¼" IBM format floppy disk

unit 3 5¼" IBM format floppy disk

The partition number (also referred to as a "slice") defines one of eight partitions (0 to 7). The *m320* driver uses the same slicing defined for the *wd*(7) driver.

Two *ioctl*(2) calls are available.

```
int fildes;
int command;
union
{
   int blkno;
   struct
   {
      int heads;
      int cylinders;
   }
      volfmt;
}
   arg;
.
.
.
ioctl (fildes, command, &arg);
```

Fildes must be an opened file descriptor (as from *open*(2)) for a raw device.

Command may be **1** or **2**. When *command* is **1**, a single track is formatted. *Arg.blkno* specifies a block in the track to be formatted relative to the origin of the slice opened as *fildes*.

When *command* is **2**, the entire volume is formatted. If the target disk is a Winchester, *arg.volfmt.heads* and *arg.volfmt.cylinders* must be the number of heads and cylinders, respectively, on the target disk. This parameter is not required for a floppy disk, but *&arg* must be a valid user memory address.

It is strongly recommended that *dinit*(1M) or *m320fmt*(1M) be used instead of direct calls to *ioctl*(2).

NOTES

A volume format request keeps the controller busy until the format completes.

On IBM format 5¼" floppy disks, track 0 contains 4 UNIX (512-byte) blocks (FM recording). Tracks 1 through 159 contain 8 blocks (MFM recording).

FILES

/dev/dsk/m320_*
/dev/rdsk/m320_*
<sys/mvme320.h>
<sys/io/m320io.h>
<sys/io/win.h>
<sys/io/sa400.h>
<sys/space/m320space.h>

SEE ALSO

config(1M), master(4), dinit(1M), m320fmt(1M).

NAME

mfp — serial port on the MVMESYS121

DESCRIPTION

Each MVMESYS121 module supports one serial port. The line behaves as described in *termio*(7). The line speed of each device can be changed under software control (output speed = input speed). The number of data bits (5, 6, 7, or 8), parity (even, odd, or no), and the number of stop bits (1 or 2) are also software selectable (refer to *stty*(1)).

FILES

/dev/mfp

SEE ALSO

stty(1), termio(7).

NAME

mvme050 — driver for the MVME050 controller

DESCRIPTION

Mvme050 provides initialization of and an interface to the MVME050 System Controller Module. Separate drivers and appropriate special devices exist for the system clock, two serial ports and parallel port which also exist on the MVME050 board.

Reading one byte from this device returns the current state of the switch bank accessible from the front of the controller. Each bit represents the state of one of the eight switches (ON=1, OFF=0).

Writing one byte to this device will cause the hex value of the byte to be displayed on the front panel display. Refer to the chapter "Operating Instructions" in the MVME050 System Controller Module and MVME701 I/O Transition Module User's Manual for additional information.

Two *ioctl*(2) system calls are available, which are of the form:

```
#define     DSPLY_ON 0
#define     DSPLY_OFF 1
ioctl( fildes, command, arg)
```

The argument *arg* is ignored.

The commands are:

DSPLY_ON Turns on the display.

DSPLY_OFF Turns off the hex display. The default condition of the display is ON.

FILES

/dev/mvme050

SEE ALSO

m050(7), lp050(7), ioctl(2).

NAME

sa400flwd — 5¼-inch Floppy Disk Drive for the Winchester Disk Driver and the general disk driver for the MVME319 and MVME320 Disk Controllers

DESCRIPTION

Although it is possible to have eight devices for each floppy disk drive, only two devices per floppy disk drive are currently defined: **dsk/**[*cntrlr*_]*x***s0** and **dsk/**[*cntrlr*_]*x***s7**.

Dsk/[*cntrlr*_]*x***s0** and **dsk/**[*cntrlr*_]*x***s7** are defined for double-sided disks in double-sided drives. These slices are created to be compatible with a double-sided, double density floppy created on a VME/10. A floppy created in **wd_0s0** on a RWIN1 (VME/10) can be read directly in [*cntrlr*_]*x***s0** on a M320 or ID.

The origin and size of the sections on each drive are as follows:

section	*start*	*length*	*description*
0	12	1264	double-sided double density
1	-	-	
2	-	-	
3	-	-	
4	-	-	
5	-	-	
6	-	-	
7	0	1276	Largest floppy size

The start address is a block address. The length is expressed in blocks, with each block containing 512 bytes. Information about recommended file system partitioning is provided in the "Administrative Guidelines" section of the *SYSTEM V/68 Administrator's Guide.*

The **dsk/*** files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a "raw" interface which provides for direct transmission between the disk and the user's read and write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files begin with **rdsk** and end with a number which selects the same disk section as the corresponding **dsk** file.

In raw I/O, the buffer must begin on a word boundary and counts must be a multiple of 512 bytes (a disk block). *Lseek*(2) calls should specify a multiple of 512 bytes.

FILES

/dev/dsk/*, /dev/rdsk/*

SEE ALSO

id(7), m320(7), wd15(7), wd40(7).

NAME

sa800flid — 8-inch Floppy Disk Drive for MVME319 Disk Controller

DESCRIPTION

Although it is possible to have eight devices for each floppy disk drive, only two devices per floppy disk drive are currently defined: **dsk/id_*x*s0** and **dsk/id_*x*s7**.

Dsk/id_*x*s1 and **dsk/id_*x*s7** are defined for double-sided disks in double-sided drives. These slices are created to be compatible with the VM21 and VM22 floppy diskettes.

The origin and size of the sections on each drive are as follows:

section	*start*	*length*	*description*
0	13	988	double-sided, single density
1	-	-	
2	-	-	
3	-	-	
4	-	-	
5	-	-	
6	-	-	
7	0	1001	

The start address is a block address. The length is expressed in blocks, with each block containing 512 bytes. Information about recommended file system partitioning is provided in the "Administrative Guidelines" section of the *SYSTEM V/68 Administrator's Guide.*

The **dsk/*** files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a "raw" interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files begin with **rdsk** and end with a number which selects the same disk section as the corresponding **dsk** file.

In raw I/O, the buffer must begin on a word boundary, and counts must be a multiple of 512 bytes (a disk block). *Lseek*(2) calls should specify a multiple of 512 bytes.

FILES

/dev/dsk/*, /dev/rdsk/*

SEE ALSO

id(7), wd40(7), sa400FLWD(7).

NAME

wd15 — 15Mb Winchester Disk Drive

DESCRIPTION

The files **dsk/***cntrlr***_0s0 ... dsk/***cntrlr***_0s7** refer to sections of the **wd15** disk drive 0. The files **dsk/***cntrlr***_1s0 ... dsk/***cntrlr***_1s7** refer to sections of drive 1. This slicing allows the device to be broken up into more manageable pieces.

The origin and size of the sections on each drive are as follows:

section	*start*	*length*
0	192	24504
1	7056	21168
2	10584	17640
3	14112	14112
4	17640	10584
5	21168	7056
6	24696	3528
7	0	28224

Slice 0 follows a 192-block reserved area used for tables and tracks for disk diagnostics.

The start address is a block address, with each block containing 512 bytes. It is extremely unwise for all of these files to be present in one installation because there is overlap in addresses, and protection becomes a problem.

The **dsk/*** files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a "raw" interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files begin with **rdsk** and end with a number which selects the same disk section as the corresponding **dsk** file.

In raw I/O the buffer must begin on a word boundary, and counts must be a multiple of 512 bytes (a disk block). Likewise, *lseek*(2) calls should specify a multiple of 512 bytes.

FILES

/dev/dsk/*, /dev/rdsk/*

SEE ALSO

cm16(7), cm80(7), fl8(7), id(7), lrk25(7), m320(7), ud(7), wd40(7).

NAME

wd40 — 40Mb Winchester Disk Drive

DESCRIPTION

The files **dsk/***cntrlr*_**0s0** ... **dsk/***cntrlr*_**0s7** refer to sections of the **wd40** disk drive 0. The files **dsk/***cntrlr*_**1s0** ... **dsk/***cntrlr*_**1s7** refer to sections of drive 1. This slicing allows the device to be broken up into more manageable pieces.

The origin and size of the sections on each drive are as follows:

section	*start*	*length*
0	192	77328
1	18792	58728
2	24192	53328
3	28224	49296
4	38640	38880
5	46824	30696
6	63072	14448
7	0	77520

Slice 0 follows a 192-block reserved area used for tables and tracks for disk diagnostics.

The start address is a block address, with each block containing 512 bytes. It is extremely unwise for all of these files to be present in one installation because there is overlap in addresses and protection becomes a problem.

The **dsk/*** files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a "raw" interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files begin with **rdsk** and end with a number that selects the same disk section as the corresponding **dsk** file.

In raw I/O the buffer must begin on a word boundary, and counts must be a multiple of 512 bytes (a disk block). Likewise, *lseek*(2) calls should specify a multiple of 512 bytes.

FILES

/dev/dsk/*, /dev/rdsk/*

SEE ALSO

id(7), wd15(7), m320(7).

SUGGESTION/PROBLEM REPORT

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

Motorola welcomes your comments on its products and publications. Please use this form.

To: Motorola Inc.
Microsystems
2900 S. Diablo Way
Tempe, Arizona 85282
Attention: Publications Manager
Maildrop DW164

Product: _____ Manual: _____

COMMENTS: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please Print

Name _____ Title _____

Company _____ Division _____

Street _____ Mail Drop _____ Phone _____

City _____ State _____ Zip _____

For Additional Motorola Publications
Literature Distribution Center
616 West 24th Street
Tempe, AZ 85282
(602) 994-6561

Four Phase/Motorola Customer Support, Tempe Operations
(800) 528-1908
(602) 438-3100

MOTOROLA

MOTOROLA ***Semiconductor Products Inc.***

P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

19109 PRINTED IN USA (8/85) MPS 1200