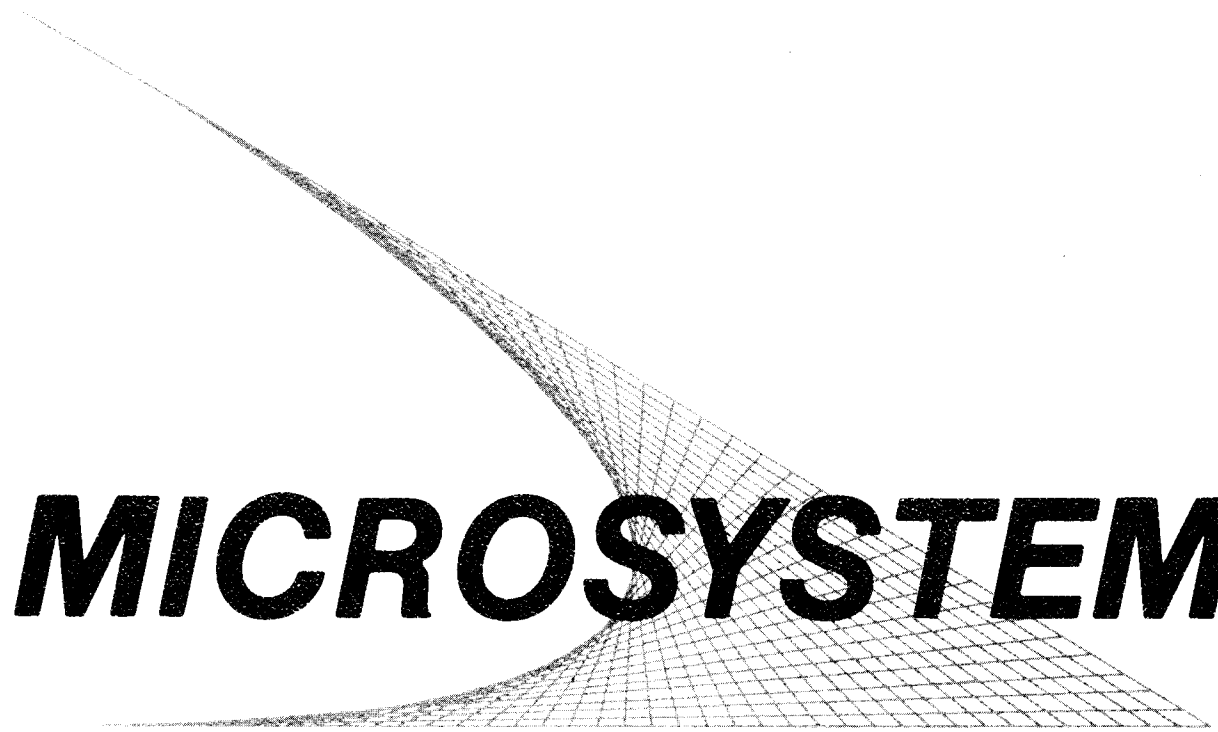
MOTOROLA

MVMEVDOS/D2

VERSAdos to VME Hardware and Software Configuration User's Manual



MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

MOTOROLA

MVMEVDOS/D2

MAY 1985

VERSAdos TO VME HARDWARE

AND

SOFTWARE CONFIGURATION

USER'S MANUAL

Second Edition

MICROSYSTEMS

SAFETY SUMMARY
SAFETY DEPENDS ON YOU

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment. Motorola Inc. assumes no liability for the customer's failure to comply with these requirements. The safety precautions listed below represent warnings of certain dangers of which we are aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

GROUND THE INSTRUMENT.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. The equipment is supplied with a three-conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter, with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE.

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

KEEP AWAY FROM LIVE CIRCUITS.

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

DO NOT SERVICE OR ADJUST ALONE.

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

USE CAUTION WHEN EXPOSING OR HANDLING THE CRT.

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

DO NOT SUBSTITUTE PARTS OR MODIFY EQUIPMENT.

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

DANGEROUS PROCEDURE WARNINGS.

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

WARNING

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

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

VMEmodules and the VME Family Support Products are modular system building blocks that reflect the latest in microcomputer architecture. VMEmodules make use of the performance and flexibility of the M68000 Family of microprocessors packaged on the internationally accepted Eurocard format. The VMEbus Specification assures mechanical integrity and compatibility for all VMEsystem products. This specification dictates the use of industrial quality 96-pin DIN connectors and sockets. These sealed, virtually gas-tight connectors give VMEsystem products exceptionally high immunity to vibration, shock, and corrosion.

Motorola's VERSAdos 4.4 Real-Time Operating System complements the advantages offered by the VMEsystem. VERSAdos provides software development capability on target VMEsystems and high-level language support. Other features include system file handling, I/O services and program loading capability, device drivers for most Motorola board-level products, and most importantly, the RMS68K real-time multitasking kernel. The powerful and universal characteristics of RMS68K can support a wide variety of application systems without large expenditures for design and programming efforts on complex real-time and multitasking functions.

The user's task is to configure a VME-based system for a defined environment using VERSAdos. In order to create a system that meets the needs of the application, the systems designer must be educated as to the alternatives available to him. The goal of this manual is to guide the systems designer through the following procedures:

- Define the VME hardware supported by VERSAdos
- Properly configure the hardware with respect to other elements in the system as well as VERSAdos
- Provide quick reference information for the user who must adjust jumper settings and utilize Motorola's debug monitors
- Install and run VERSAdos
- Provide support information on SYSGEN requirements, bus addresses, memory requirements, etc.

The user should view this document as a reference tool where pertinent component information can be found along with step-by-step procedures for configuring a system. The manual is organized in a modular format that enables the user to easily locate information pertaining to specific topics. This format enables a user to design a variety of systems including the standard VERSAdos-supported systems employing Motorola microprocessor modules.

1

This manual is organized by first presenting hardware-oriented information (through Chapter 7) followed by software-oriented material. For ease of reference, Chapters 2 through 7 describe the standard VERSAdos-supported configurations of various VMEmodules. Chapter 8 describes the standard systems supported by VERSAdos. Chapters 9 and 10 detail the procedures needed to configure various components to meet specific system requirements using VERSAdos. A set of appendices provide handy reference material for the user.

1.2 NON-VME COMPONENTS

While this manual is specifically geared to the VME/VERSAdos user, non-VME components are occasionally referred to in various tables and examples. This information is included due to the fact that while these modules are VERSAdos-supported, the information is not readily available in another document. It is hoped that the system designer in possession of one of the non-VMEmodules mentioned in this manual will be aided by the inclusion of this information.

1.3 RELATED DOCUMENTATION

It is assumed that the user designing a system with the help of this manual has already read the user's manual for each of the components in the target system. While some information (such as debug monitor commands) is provided as a reference, this manual does not attempt to duplicate the intended efforts of the user manuals. In addition to the component user manuals, the following documents are helpful references when configuring a VMEsystem using VERSAdos. They may be obtained from Motorola's Literature Distribution Center, 616 West 24th Street, Tempe, AZ 85282; telephone (602) 994-6561.

DOCUMENT TITLE	MOTOROLA PUBLICATION NUMBER
VMEsystem Architecture Manual	MVMESYSAM
VMEbus Specification Manual	MVMEBS
Guide to Writing Device Drivers for VERSAdos	M68KDRVGD
M68000 Real-Time Multitasking Software User's Manual	M68KRMS68K
System Generation Facility User's Manual	M68KSYSGEN
VERSAdos Messages Reference Manual	M68KVMSG

CHAPTER 2

MONOBOARD MICROCOMPUTERS

2.1 MVME101 MONOBOARD MICROCOMPUTER

The MVME101 Monoboard Microcomputer is a double Eurocard VMEmodule designed to operate as a monoboard system, as a single CPU controller in a VMEbus system, or as a CPU controller in a multiprocessor system.

The MVME101 Monoboard Microcomputer utilizes an 8 MHz MC68000 microprocessor incorporating a 16-bit data bus and a 16Mb addressing range. The module is equipped with eight 28-pin industry-standard JEDEC sockets capable of handling RAM or ROM devices ranging from 2Kb to 32Kb each. The MVME101 features 20 programmable I/O lines, two RS-232C serial I/O ports, seven jumper-selectable interrupt priority levels, and a triple-programmable 16-bit MC6840 Programmable Timer Module.

2.1.1 Standard VERSAdos Configuration

Figure 2-1 illustrates the MVME101 jumper configuration necessary to boot VERSAdos. Table 2-1 lists the jumpers and their locations. A detailed description of this information is provided in the VME101 System VERSAdos Hardware and Software Configuration Manual (MVMECNFG1). Refer to the MVME101 MC68000 Monoboard Microcomputer User's Manual (MVME101) to compare these settings with the factory-set jumper configuration.

2.1.2 MVME101bug Debug Monitor

The MVME101 module is configured for use with the MVME101bug Debug Monitor. The MVME101bug Debug Monitor commands are listed in Table 2-2.

2

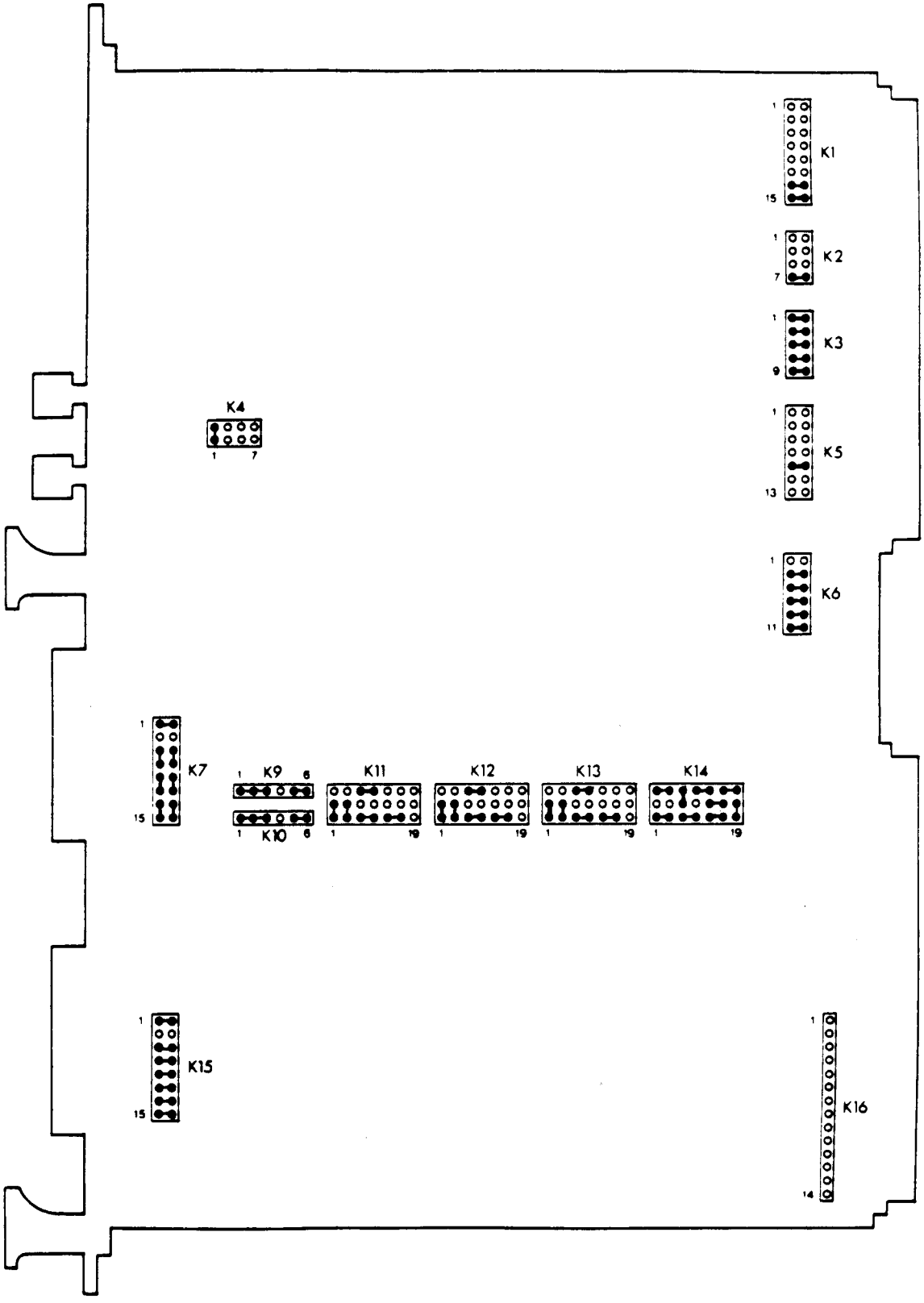


FIGURE 2-1. MVME101 Jumper Header Locations

TABLE 2-1. MVME101 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME101 VERSAdos JUMPER SETTINGS	
K1	VMEBUS REQUESTER PRIORITY LEVEL	13-14, 15-16
K2	SELECT	7-8
K3	VMEBUS SYSTEM CONTROLLER FUNCTION SELECT	1-2, 3-4, 5-6, 7-8 9-10
K4	LOCAL ROM ACCESS TIME SELECT	1-2
K5	VMEBUS INTERRUPT REQUEST SELECT	9-10
K6	LOCAL INTERRUPT REQUEST SELECT	3-4, 5-6, 7-8, 9-10, 11-12
K7	PORT 1 CONFIGURATION SELECT	1-2, 5-7, 6-8, 9-11, 10-12, 13-15, 14-16
K9	PORT 1 INTERFACE CONTROL SELECT	1-2, 2-3, 5-6
K15	PORT 2 CONFIGURATION SELECT	1-2, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16
K10	PORT 2 INTERFACE CONTROL SELECT	1-2, 2-3, 5-6
K16	PTM CONFIGURATION SELECT	NO JUMPERS
K11	MEM1 MEMORY CONFIGURATION SELECT	1-2, 4-5, 7-10, 9-12, 13-16
K12	MEM2 MEMORY CONFIGURATION SELECT	1-2, 4-5, 7-10, 9-12, 13-16
K13	MEM3 MEMORY CONFIGURATION SELECT	1-2, 4-5, 7-10, 9-12, 13-16
K14	MEM4 MEMORY CONFIGURATION SELECT	1-4, 3-6, 7-10, 8-9, 12-15, 13-16, 14-17, 18-21, 19-20
	MVME101 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 2-2. MVME101bug 3.n Commands

COMMAND	DESCRIPTION
BD	Bootstrap dump
BF <addr1> <addr2> <data>	Block of memory fill
BH	Bootstrap halt
BI <addr1> <addr2>	Block of memory initialize
BM <addr1> <addr2> <addr3>	Block of memory move
BO	Bootstrap program
[NO]BR [<addr1>[<;count1>]] ...	Set and remove breakpoints
BS	Block of memory search
BT <addr1> <addr2>	Block of memory test
DC <expr>	Data conversion
DF	Display formatted MPU registers
DU[<port>] <addr1> <addr2> [<text>]	Dump memory (S-records)
GD [<addr>]	Go direct execute program
GO [<addr>]	Go execute program
GT <addr>	Go execute program to breakpoint
HE	Help
IOP	Disk I/O physical
IOT	Disk I/O teach
LO[;<opts>][=<text>]	Load memory (S-records)
MD[<port>] <addr> [<count>][;<opt>]	Memory display/disassembly
MM <addr> [opts>]	Memory modify/disassembly/assembly
MS <addr> <data> [<data> ...]	Memory set
OF	Display all relative offsets
[NO]PA	Printer attach/detach
PF[<port>]	Port format
TM [<char>]	Transparent mode
TR [<count>]	Trace one instruction
TT <addr>	Trace to temporary breakpoint
VE[;=<text>]	Verify memory (S-records)
.<register> commands	
.A	Display all address registers
.A0-.A7 [<expr>]	Display/set address register
.D	Display all data registers
.D0-.D7 [<expr>]	Display/set data register
.PC [<expr>]	Display/set program counter
.R0-R6 [<expr>]	Display/set relative offset
.SR [<expr>]	Display/set status register
.SS [<expr>]	Display/set superv. stack pointer
.US [<expr>]	Display/set user stack pointer

2.2 MVME110-1 MONOBOARD MICROCOMPUTER

The MVME110-1 Monoboard Microcomputer is a double Eurocard VMEmodule designed to function as a stand-alone microcomputer, a single CPU/controller in a VMEbus system, or as a single CPU element in a multiprocessor VMEbus configuration. The MVME110-1 is equipped with an MC68000 16-bit microprocessor running at 8 MHz. Eight 28-pin sockets enable the module to possess a maximum of 128Kb of RAM, or 64Kb of ROM, or a mixture of these memories. Full support is provided for the Motorola I/O Channel which provides access to a large variety of peripheral and industrial I/O functions. Three independent 16-bit counter/timers are available to implement various forms of interrupts and watchdog timers. An RS-232C serial port is provided to allow use of a terminal or a down-loading function to the MVME110-1.

2.2.1 Standard VERSAdos Configuration

Figure 2-2 illustrates the VERSAdos-supported jumper configuration of the MVME110-1. Table 2-3 lists the jumper locations and settings. Refer to the MVME110-1 VMEmodule Monoboard Microcomputer User's Manual (MVME110) to compare these settings with the factory-set jumper configuration.

2.2.2 MVME110-1 Debug Monitor

The MVME110-1 Debug Monitor, MVME110bug, is a powerful debugging and evaluation tool for use in an MVME110-1 Monoboard Microcomputer environment. Table 2-4 describes the MVME110bug command set.

2

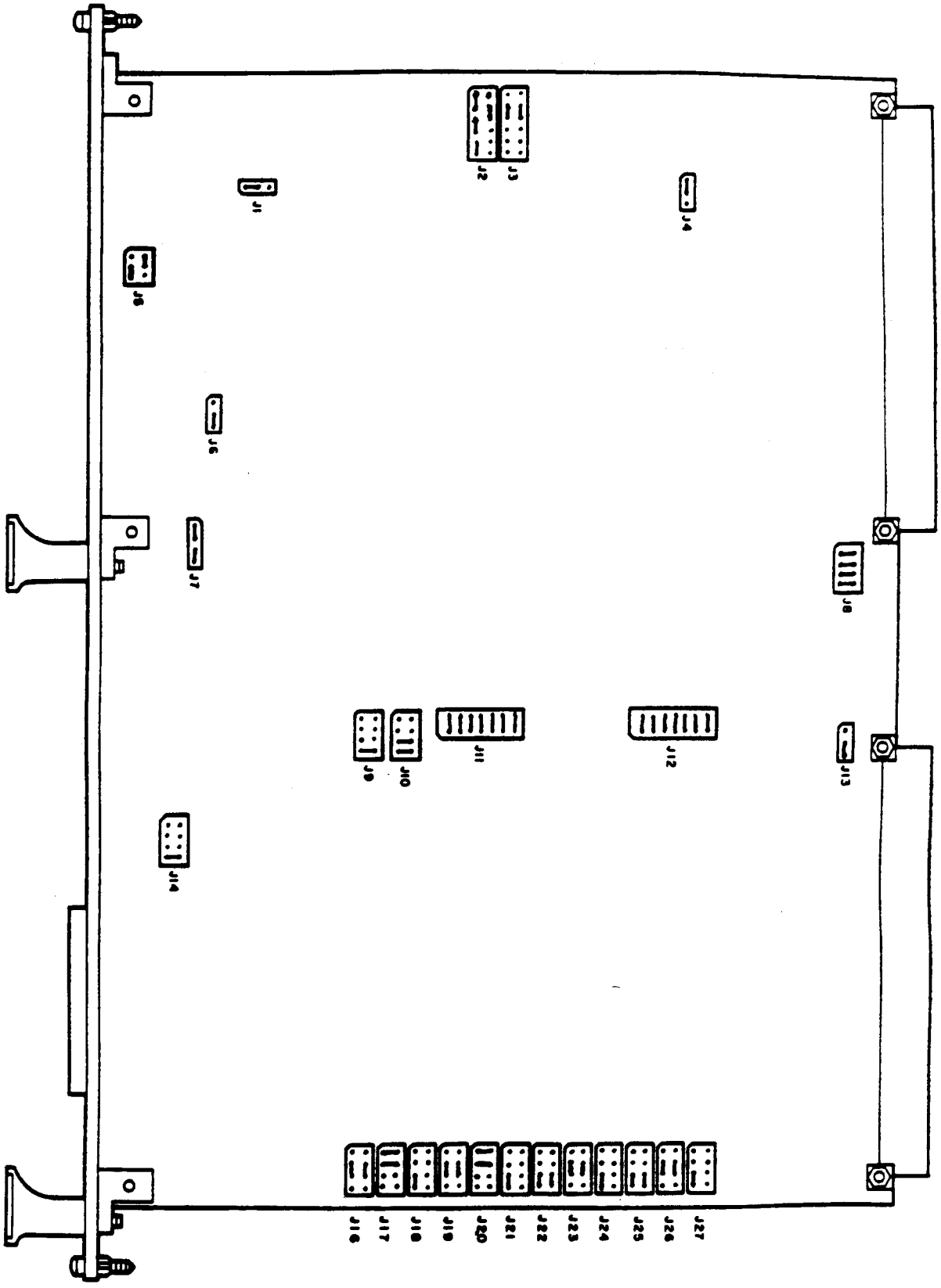


FIGURE 2-2. MVME110-1 Jumper Header Locations

2

TABLE 2-3. MVME110-1 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME110-1 VERSAdos JUMPER SETTINGS	
J1	MPU CLOCK SPEED SELECT	1-2
J2	BUS ARBITRATION REQUEST LEVEL SELECT	1-3, 4-6, 5-7, 9-11
J3		3-5,4-6
J4	ADDRESS MODIFIER 3 SELECT	1-2
J5	SOFTWARE STATUS BITS SELECT	2-4, 3-5
J6	LOCAL BUS TIME-OUT SELECT	2-3
J7	RESET & ABORT SWITCH ENABLE SELECT	1-2, 3-4
J8	SYSTEM CONTROLLER FUNCTIONS SELECT	1-2, 3-4, 5-6, 7-8
J9	ROM ACCESS TIME SELECT	7-8
J10	RAM ACCESS TIME SELECT	5-6, 7-8
J11	LOCAL INTERRUPT ENABLE SELECT	1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14
J12	VMEBUS INTERRUPT ENABLE SELECT	1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14
J13	I/O CHANNEL BUFFER OPTION	2-3
J14	BAUD RATE SELECT	7-8
J16	SOCKET PAIR 1 MEMORY CONFIGURATION	3-5, 4-6
J17	SELECT	1-2, 3-4
J18		5-7
J19	SOCKET PAIR 2 MEMORY CONFIGURATION	3-5, 4-6
J20	SELECT	1-2, 3-4
J21		5-7
J22	SOCKET PAIR 3 MEMORY CONFIGURATION	5-7, 6-8
J23	SELECT	3-5, 4-6
J24		5-7
J25	SOCKET PAIR 4 MEMORY CONFIGURATION	5-7, 6-8
J26	SELECT	3-5, 4-6
J27		5-7

2

TABLE 2-3. MVME110-1 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME110-1 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 2-4. MVME110bug Command and Option Summary

COMMAND	DESCRIPTION
BD [<device>][,<controller>]	Boot dump
BF <address1> <address2> <pattern>	Block of memory fill
BH [<device>][,<controller>]	Bootstrap halt
BI <address1> <address2>	Block of memory initialize
BM <address1> <address2> <address3>	Block of memory move
BO [<device>][,<controller>][,<string>]	Bootstrap operating system
[NO]BR [<address>[;<count>]] [<address>[;<count>]] . . .	Breakpoint set (and remove)
BS <address1> <address2> <data> [<mask>][;<option>]	Block of memory search; options B, W, L
BT <address1> <address2>	Block of memory test
DC <expression>	Data conversion
DF	Display formatted registers
DU[<port number>] <address1> <address2> [<text>]	Dump memory (S-records)
GD [<address>]	Go direct execute program
G[O] [<address>]	Go execute program
GT <temporary breakpoint address>	Go until breakpoint
HE	Help
IOC	I/O direct command
IOP	I/O physical to disk
IOT [<device>][,<controller>]	I/O "teach" to disk
LO[<port number>] [;[<options>]=<text>]	Load (S-records); options X, -C
MD[<port number>] <address> [<count>][;DI]	Memory display/disassembly
M[M] <address>[;<options>]	Memory modify; options W, L, O, V, N, DI
MS <address> <data>	Memory set
OF	Display offsets
[NO]PA[<port number>]	Printer attach (and detach)
PF[<port number>]	Port format
TM[<port number>][<exit character>[<trailing character>]]	Transparent mode
T[R] [<count>]	Trace
TT <breakpoint address>	Temporary breakpoint trace
VE[<port number>][;[<options>]=<text>]	Verify (S-records); options X, -C
VI	Vector initialize
.A0 - .A7 [<expression>]	Display/set address register
.D0 - .D7 [<expression>]	Display/set data register
.R0 - .R6 [<expression>]	Display/set relative offset register
.PC [<expression>]	Display/set program counter
.SR [<expression>]	Display/set status register
.SSP [<expression>]	Display/set supervisor stack pointer
.USP [<expression>]	Display/set user stack pointer

2

2.3 MVME115M MONOBOARD MICROCOMPUTER

The MVME115M Monoboard Microcomputer is designed to function as a single CPU in a VMEbus system, or as an element in a multiprocessor VMEbus configuration.The module is shipped with an 8 MHz MC68010 16-bit microprocessor and an MC68451 Memory Management Unit (MMU). The MVME115M accepts up to 64Kb of onboard ROM/PROM and up to 48Kb of RAM devices with a combined maximum of 96Kb of onboard memory. It is recommended that the MVME025 or MVME050 System Controller modules be added to systems employing the MVME115M.

The MVME115M can be configured to use Large-Scale Integrated (LSI) parts of a different maximum clock frequency than those supplied. Any frequency between 2.0 MHz and 10 MHz is supported. The actual frequency used must not exceed the rated maximum clock frequency of the MC68010 microprocessor, the MC68451 MMU, or the MC68320 Parallel Interface/Timer.

The MVME115M can be used with or without an MC68451 MMU. To run software which does not use the MMU, it is not required to remove the MMU. On system reset, the MMU automatically configures itself to a transparent state, in which all logical addresses generated by the MC68010 microprocessor are mapped to identical physical addresses. Until the MMU registers are configured by software to implement a different mapping, no segment fault can be generated.

The MVME115M contains two asynchronous serial ports, implemented primarily by a single SCN2681 Dual Asynchronous Receiver/Transmitter (DUART). This device provides two independent, full duplex, asynchronous transmitter/receiver channels which are independently programmable for operating mode, data format, and a baud rate ranging from 50.0 baud to 38.4 kilobaud.

2.3.1 Standard VERSAdos Configuration

Figure 2-3 illustrates the standard jumper locations and settings of the MVME115M necessary to boot VERSAdos. Table 2-5 lists the jumpers and the required settings. Refer to the MVME115M User's Manual (MVME115M) to compare these settings with the factory-set configuration.

2.3.2 MVME115 Debug Monitor

The MVME115M module utilizes the MVME115M Debug Monitor for debugging and diagnostic purposes. As a reference during debugging operations, the MVME115M Debug Monitor commands are listed in Table 2-6.

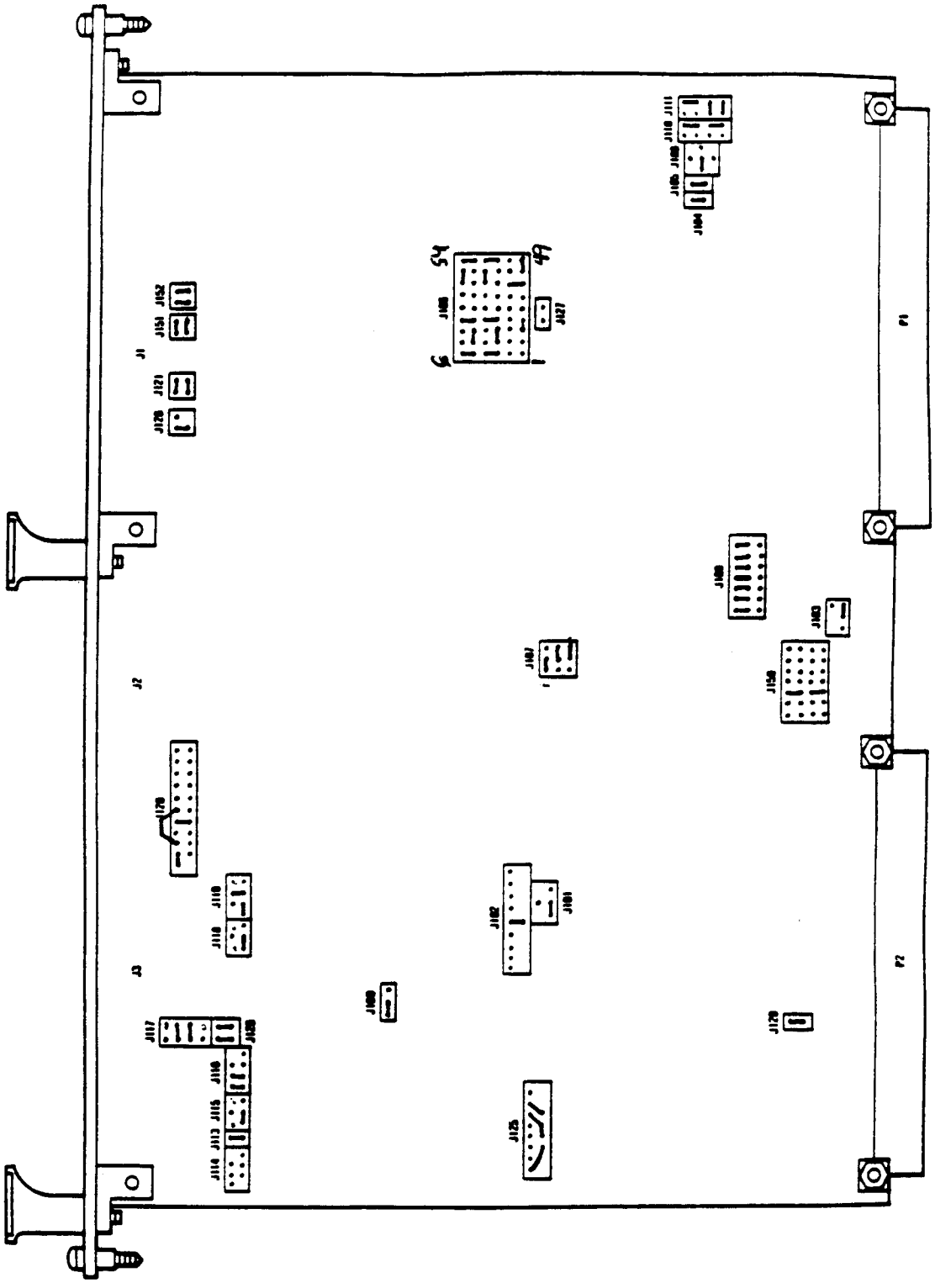


FIGURE 2-3. MVME115M Jumper Header Locations

2

TABLE 2-5. MVME115M VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME115M VERSAdos JUMPER SETTINGS	
J107	ONBOARD MEMORY ROM/RAM SELECT	1-2, 5-6, 8-9
J106	ONBOARD MEMORY CHIP TYPE SELECT	3-4, 5-6, 9-15, 11-17, 13-19, 21-22, 23-24, 25-26, 40-46, 42-48, 43-49, 51-52, 53-54
J125	ONBOARD MEMORY ACCESS TIME SELECT	1-9, 6-11-10, 7-12
J127	BATTERY BACKUP SELECT	NO JUMPER
J151	PORT A DIRECTION SELECT	1-2, 3-4
J152	PORT B DIRECTION SELECT	1-3, 2-4
J121	SERIAL PORT STATUS LINE CONFIGURATION SELECT	1-2, 3-4
J128		1-3
J113	PARALLEL PORT CONFIGURATION SELECT	1-2
J114		NO JUMPERS
J115		1-2
J116		1-2, 3-4
J117		1-4, 2-5
J118		1-2
J119		2-3, 5-6
J120		2-20, 4-9, 7-8
J126		1-2, 3-4
J150	ONBOARD INTERRUPT LEVEL SELECT	5-12, 19-26
J109	VMEBUS REQUEST LEVEL SELECT	2-3
J110		2-4, 6-8
J111		2-4, 5-6, 7-8
J129	VMEBUS ARBITRATION OPTION SELECT	1-2
J103	RESET SELECT	2-3
J104	BUS ERROR HANDLING SELECT	1-2
J105		1-2
J108	VMEBUS INTERRUPT LEVEL SELECT	1-8, 2-9, 3-10, 4-11, 5-12, 6-13, 7-14

TABLE 2-5. MVME115M VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
J101	STROBE TIMING SELECT	2-3
J102		5-10
J100	CLOCK SPEED SELECT	1-2
	MVME115M 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 2-6. MVME115 Debug Monitor Command and Option Summary

COMMAND	DESCRIPTION
BD [<device>][,<controller>]	Bootstrap dump
BF <address1> <address2> <pattern>	Block fill
BH [<device>][,<controller>]	Boot and halt
BI <address1> <address2>	Block initialize
BM <address1> <address2> <address3>	Block move
BO [<device>][,<controller>][,<string>]	Boot operating system
[NO]BR [<address>[;<count>]]...	Set and remove breakpoints
BS <address1> <address2> '<literal string>'	Block search; options ;B ;W ;L
<address1> <address2> <data>[<mask>][;<option>]	
BT <address1> <address2>	Block test
DC <expression>	Data conversion/evaluation
DF	Display formatted registers
DU[<port number>] <address1> <address2> [<text>]	Dump memory (S-records)
GD [<address>]	Go direct
G[O] [<address>]	Install breakpoints and go
GT <temporary breakpoint address>	Go until address
HE	Display commands/registers
IOC	Issue disk command
IOP	Issue physical read/write
IOT [<device>][,<controller>]	Teach disk controller a configuration
LO<port number> [;<options>] =<text>	Load (S-records)
MD[<port number>] <address> [<count>][;<options>]	Memory display; options ;DI ;S
M[M] <address>[;<options>]	Memory modify; options ;W ;L ;O ;V ;N ;DI
MS <address> <data>	Memory set (also ASCII)
OF	Offset register display
[NO]PA<port number>	Printer attach/detach
PF<port number>	Port format
TA[<port number>]	Terminal attach
TM <port number> [<exit character> [<trailing character>]]	Transparent mode
T[R] [<count>]	Trace
TT <breakpoint address>	Trace until address
VE<port number> [;<options>] =<text>	Verify (S-records)
VI	Vector initialize

TABLE 2-6. MVME115 Debug Monitor Command and Option Summary (cont'd)

COMMAND	DESCRIPTION
(See description of .<register> commands)	
.A0 - .A7 [<expression>]	Display/set address register
.D0 - .D7 [<expression>]	Display/set data register
.DFC [<expression>]	Display/set destination function code
.PC [<expression>]	Display/set program counter
.R0 - .R7 [<expression>]	Display/set relative offset register
.SFC [<expression>]	Display/set source function code
.SR [<expression>]	Display/set status register
.SS[P] [<expression>]	Display/set supervisor stack pointer
.US[P] [<expression>]	Display/set user stack pointer
.VBR [<expression>]	Display/set vector base register
Key Functions:	
(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

2.4 MVME120 SERIES MICROPROCESSOR MODULES

The MVME120 series of microprocessor modules consists of the MVME120, MVME121, MVME122, and MVME123 modules. These modules are based on the MVME120 and are offered with several combinations of important features.

The MVME120 VMEmodule is a high-performance microprocessor module that offers a solution for high-speed data processing, management, and control applications. The module contains an MC68010 MPU, MC68451 MMU, logical instruction cache, a serial debug terminal port, programmable tick timer, private ROM, and a large dynamic dual port RAM that may be loaded externally via the VMEbus. The MPU module, when used with a system controller module (such as the MVME025 or MVME050 system controllers or the MVME110-1 CPU module), satisfies system requirements ranging from simple, high-speed single processor applications to complex multiprocessor system architectures.

Four configurations are available for the MVME120 family of microprocessor modules. The configurations are listed below:

MVME120	10 MHz MC68010, 128Kb dynamic RAM, cache, MMU
MVME121	10 MHz MC68010, 512Kb dynamic RAM, cache, MMU
MVME122	12.5 MHz MC68010, 128Kb dynamic RAM, no cache, no MMU
MVME123	12.5 MHz MC68010, 512Kb dynamic RAM, cache, no MMU

2.4.1 MVME120 Series Standard VERSAdos Configurations

Figures 2-4 through 2-7 illustrate the jumper locations and settings required to allow VERSAdos to operate using the MVME120 series of microprocessor modules. Tables 2-7 through 2-10 list the jumpers and their required settings. Refer to the MVME120/MVME121/MVME122/MVME123 VMEbus Microprocessor Module User's Manual (MVME120) to compare these settings with the factory-set configurations.

2.4.2 MVME120 Debug Monitor

The MVME120 family of microprocessor modules make use of the MVME120 Debug Monitor for debugging and diagnostic purposes. The MVME120 Debug Monitor commands are listed in Table 2-11.

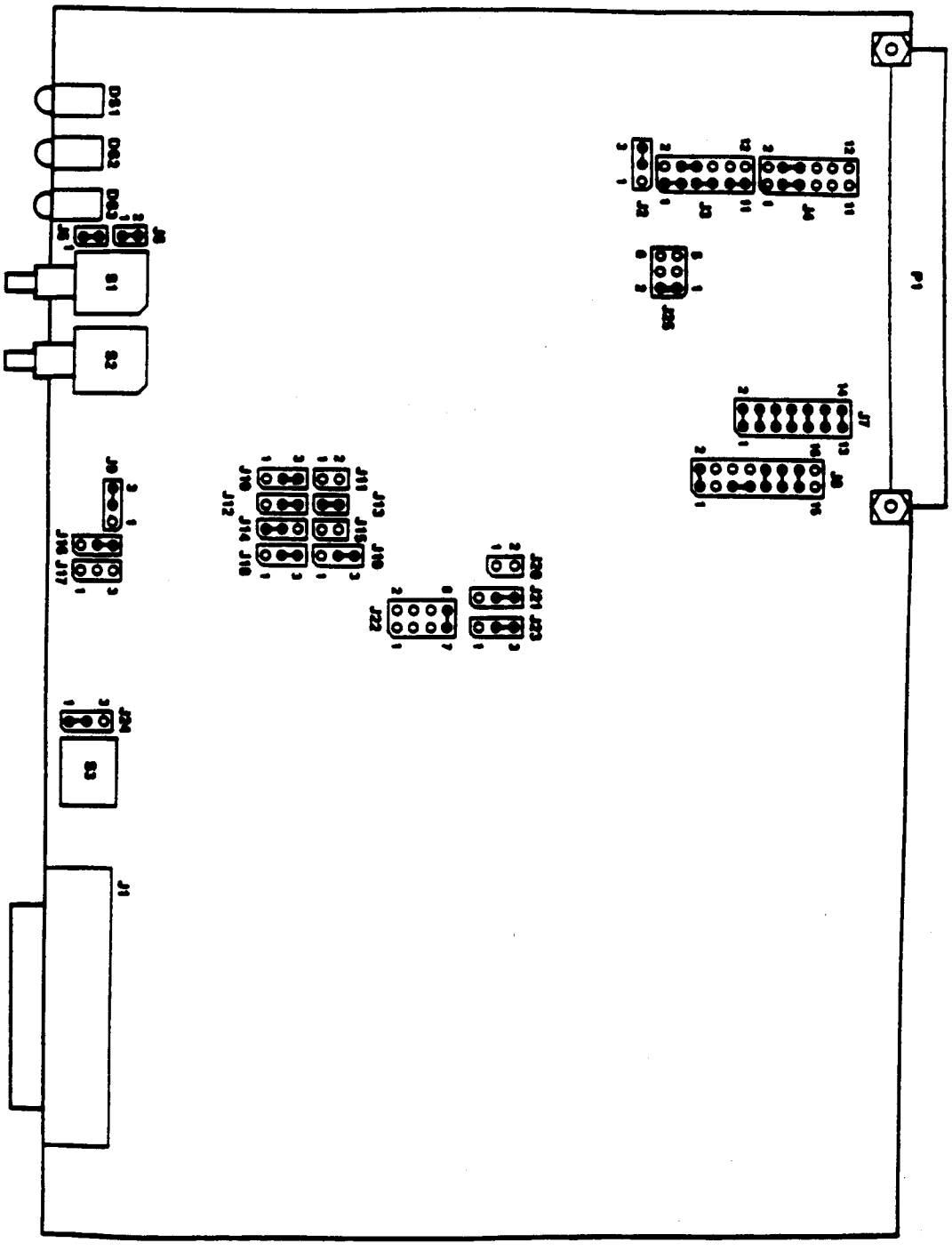


FIGURE 2-4. MVME120 Jumper Header Locations

2

TABLE 2-7. MVME120 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME120 VERSAdos JUMPER SETTINGS	
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	2-3
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		1-2
J16		2-3
J18		2-3
J19		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	2-3
J24	RESET VECTOR FETCH MODE SELECT	1-2
J25	LOCAL TIME-OUT SELECT	1-2

TABLE 2-7. MVME120 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME120 VERSAdos SWITCH SETTINGS	
SWITCH NUMBER		
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
	MVME120 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

2

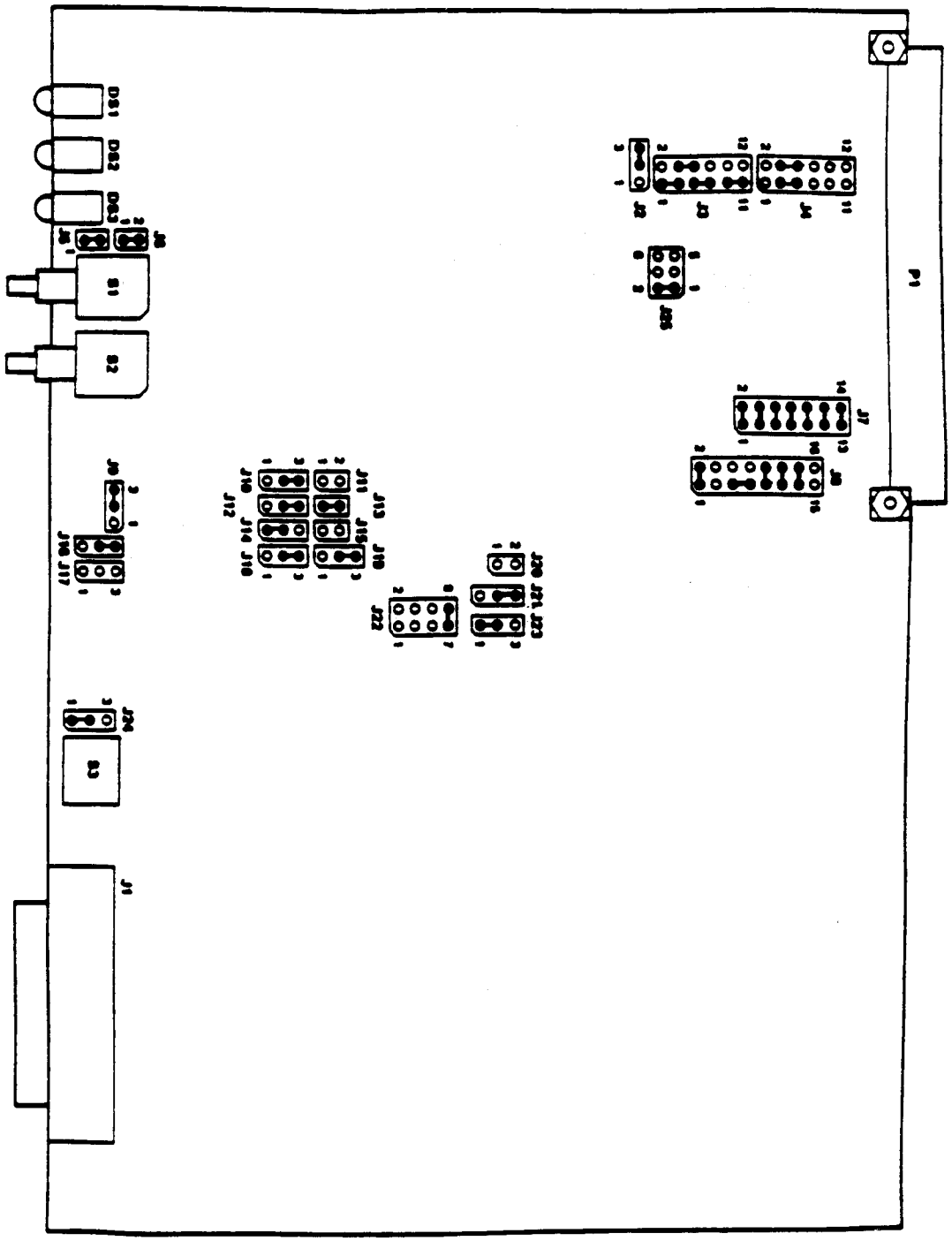


FIGURE 2-5. MVME121 Jumper Header Locations

TABLE 2-8. MVME121 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME121 VERSAdos JUMPER SETTINGS	
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	2-3
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		1-2
J16		2-3
J18		2-3
J19		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

2

TABLE 2-8. MVME121 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME121 VERSAdos SWITCH SETTINGS	
SWITCH NUMBER		
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
	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

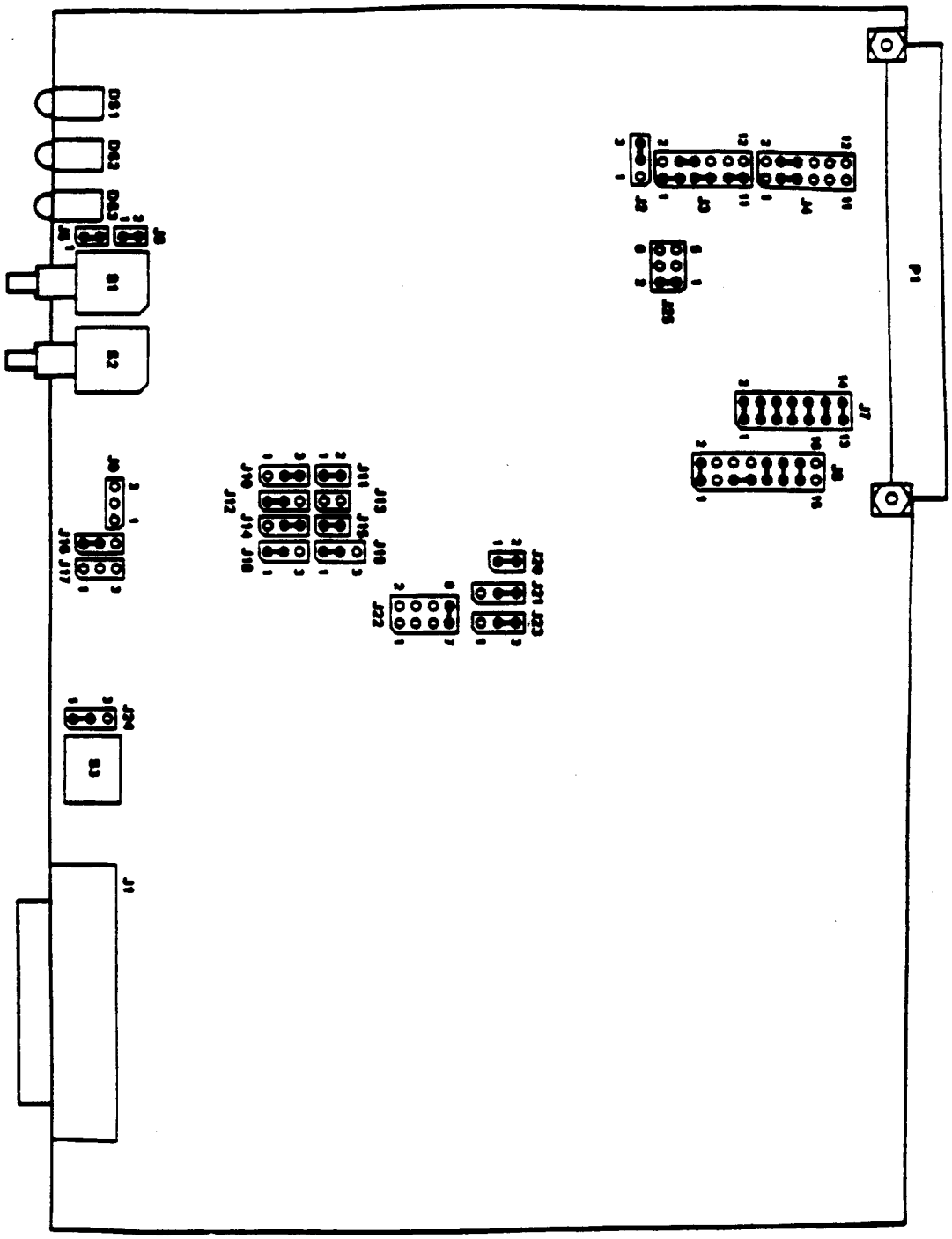


FIGURE 2-6. MVME122 Jumper Header Locations

2

TABLE 2-9. MVME122 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME122 VERSAdos JUMPER SETTINGS	
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	NO JUMPER
J17	CACHE CONFIGURATION SELECT	NO JUMPER
J10	MMU STROBE SELECT	2-3
J11	MMU ADDRESS STROBE SELECT	1-2
J12	MAP DECODE TIME SELECT	1-2
J13	RAM CYCLE TIMING SELECT	NO JUMPER
J14		2-3
J16		1-2
J18		1-2
J19		1-2
J15	REFRESH PERIOD SELECT	1-2
J20	MSR BIT 1 SELECT	1-2
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

TABLE 2-9. MVME122 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME122 VERSAdos SWITCH SETTINGS	
SWITCH NUMBER		
S3-1	CACHE DISABLE	CLOSED
S3-2	BAUD RATE SELECT (12.5-10 MHz)	OPEN
S3-3	RESET VECTOR FETCH SELECT	CLOSED
S3-4	RESET VECTOR FETCH SELECT	OPEN
	MVME122 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

2

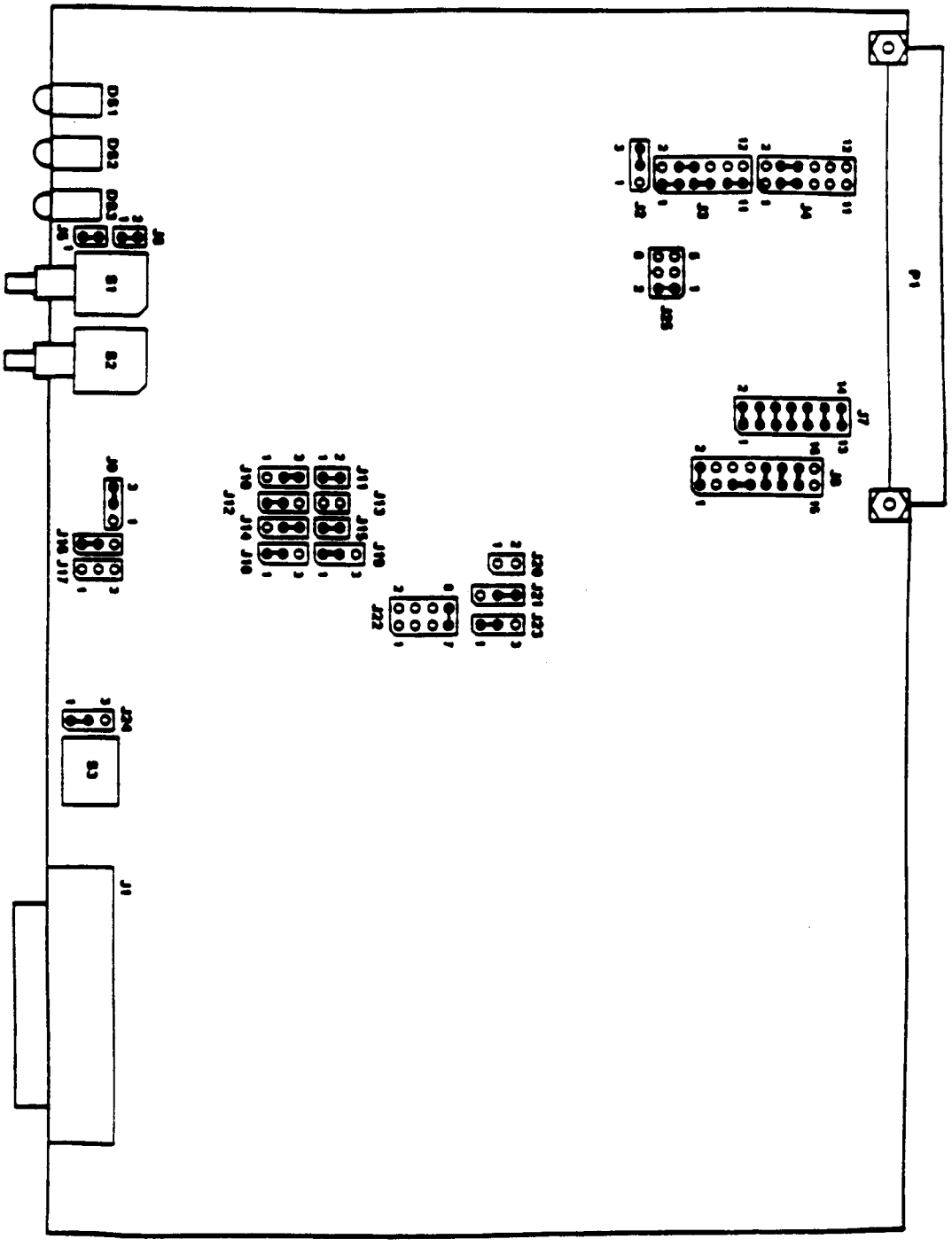


FIGURE 2-7. MVME123 Jumper Header Locations

TABLE 2-10. MVME123 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME123 VERSAdos JUMPER SETTINGS	
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	2-3
J17	CACHE CONFIGURATION SELECT	NO JUMPERS
J10	MMU STROBE SELECT	2-3
J11	MMU ADDRESS STROBE SELECT	1-2
J12	MAP DECODE TIME SELECT	1-2
J13	RAM CYCLE TIMING SELECT	NO JUMPERS
J14		2-3
J16		1-2
J18		1-2
J19		1-2
J15	REFRESH PERIOD SELECT	1-2
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

2

TABLE 2-10. MVME123 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME123 VERSAdos SWITCH SETTINGS	
SWITCH NUMBER		
S3-1	AUTOBOOT SELECT	OPEN
S3-2	BAUD RATE SELECT (12.5-10 MHz)	OPEN
S3-3	RESET VECTOR FETCH SELECT	CLOSED
S3-4	RESET VECTOR FETCH SELECT	OPEN
	MVME123 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 2-11. MVME120 Debug Monitor Command and Option Summary

COMMAND	DESCRIPTION
BD [<device>][,<controller>]	Bootstrap dump
BF <address1> <address2> <pattern>	Block fill
BH [<device>][,<controller>]	Boot and halt
BI <address1> <address2>	Block initialize
BM <address1> <address2> <address3>	Block move
BO [<device>][,<controller>][,<string>]	Boot operating system
[NO]BR [<address>[;<count>]]...	Set and remove breakpoints
BS <address1> <address2> '<literal string>'	Block search; options ;B ;W ;L
<address1> <address2> <data>[<mask>][;<option>]	
BT <address1> <address2>	Block test
CACHE	Manipulate MVME120 cache
DC <expression>	Data conversion/evaluation
DF	Display formatted registers
DU[<port number>] <address1> <address2> [<text>]	Dump memory (S-records)
GD [<address>]	Go direct
G[O] [<address>]	Install breakpoints and go
GT <temporary breakpoint address>	Go until address
HE	Display commands/registers
IOC	Issue disk command
IOP	Issue physical read/write
IOT [<device>][,<controller>]	Teach disk controller a configuration
LO<port number> [;<options>] =<text>	Load (S-records)
MD[<port number>] <address> [<count>][;<options>]	Memory display; options ;DI ;S
M[M] <address>[;<options>]	Memory modify; options ;W ;L ;O ;V ;N ;DI
MP <address>	Map memory for disk
MS <address> <data>	Memory set (also ASCII)
OF	Offset register display
[NO]PA<port number>	Printer attach/detach
PF<port number>	Port format
ST	Self-test
TA[<port number>]	Terminal attach
TM <port number> [<exit character> [<trailing character>]]	Transparent mode
T[R] [<count>]	Trace
TT <breakpoint address>	Trace until address
VE<port number> [;<options>] =<text>	Verify (S-records)
VI	Vector initialize

TABLE 2-11. MVME120 Debug Monitor Command and Option Summary (cont'd)

COMMAND	DESCRIPTION
.A0 - .A7 [<expression>]	Display/set address register
.D0 - .D7 [<expression>]	Display/set data register
.DFC [<expression>]	Display/set destination function code
.PC [<expression>]	Display/set program counter
.R0 - .R7 [<expression>]	Display/set relative offset register
.SFC [<expression>]	Display/set source function code
.SR [<expression>]	Display/set status register
.SSP [<expression>]	Display/set supervisor stack pointer
.USP [<expression>]	Display/set user stack pointer
.VBR [<expression>]	Display/set vector base register
Key Functions:	
(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

2.5 MVME025 SYSTEM CONTROLLER MODULE

The MVME025 System Controller module offers the bus arbitration, monitor, and utility functions usually required in a VMEbus system. This module combines clock generation, AC fail and system reset control, bus time-out and error control, and bus arbitration capabilities thus eliminating the need for these functions on other modules. The MVME025 is especially useful in multiprocessor and intelligent Direct Memory Access (DMA) applications, and is recommended in applications requiring the MVME115M Monoboard Microcomputer.

2.5.1 MVME025 Standard VERSAdos Configuration

Figure 2-8 illustrates the jumper locations and settings required for VERSAdos operation using the MVME025. Table 2-12 lists the jumpers and the required settings. Refer to the MVME025 VMEbus System Controller User's Manual (MVME025) to compare these settings with the factory-set configuration.

2

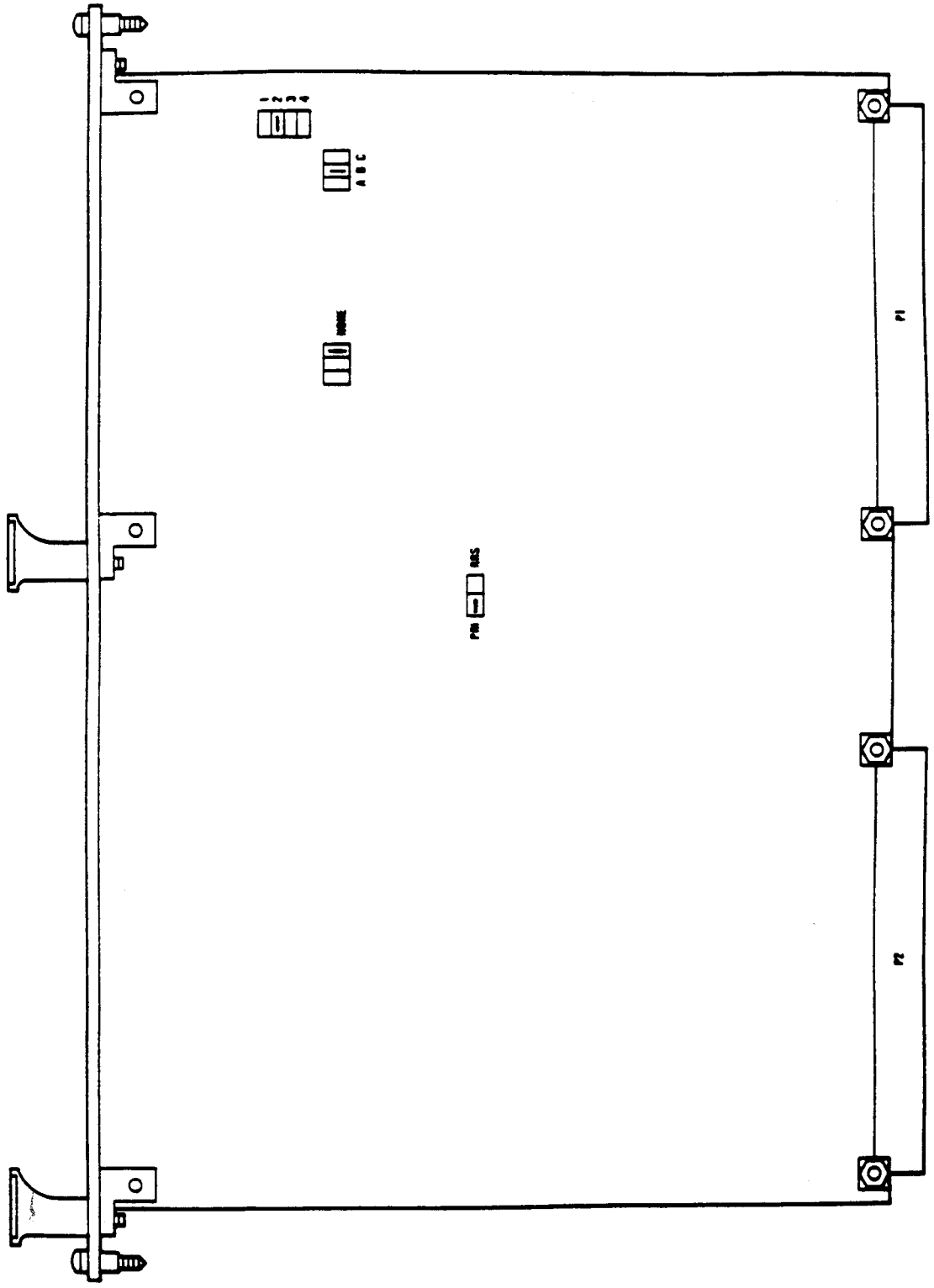


FIGURE 2-8. MVME025 Jumper Header Locations

TABLE 2-12. MVME025 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME025 VERSAdos JUMPER SETTINGS	
	AC - FAIL POLARITY	NONE
	BUS TIME-OUT	B,2
	BUS ARBITRATION	PRI
	MVME025 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

2.6 MVME050 SYSTEM CONTROLLER AND MVME701 I/O TRANSITION MODULES

The MVME050 System Controller module is a combination system controller, system utility, and debug/diagnostic module. The MVME050 is designed to implement functions that are only required once in a multiprocessor system such as bus arbitration, bus time-out, system clock generation, and time-of-day clock. This capability eliminates the need to duplicate these functions on every processor in the system (although only one can be enabled). The MVME050 is typically used in an MVME120-based system.

The MVME701 I/O Transition module is an optional, Euro-format, double-high companion board to the MVME050. The MVME701 may be attached via a ribbon cable to the MVME050 P2 connector, providing the connectors for the system controller's serial and parallel ports, and optionally, batteries for back-up of the MVME050 time-of-day clock. There are no active components on the MVME701.

2.6.1 MVME050/701 Standard VERSAdos Configuration

Figures 2-9 and 2-10 illustrate the jumper locations and settings for the MVME050 and MVME701 respectively. Tables 2-13 and 2-14 list the jumpers and their settings for these modules. Refer to the MVME050 System Controller/MVME701 I/O Transition Module User's Manual (MVME050) to compare these settings with the factory-set configurations.

2

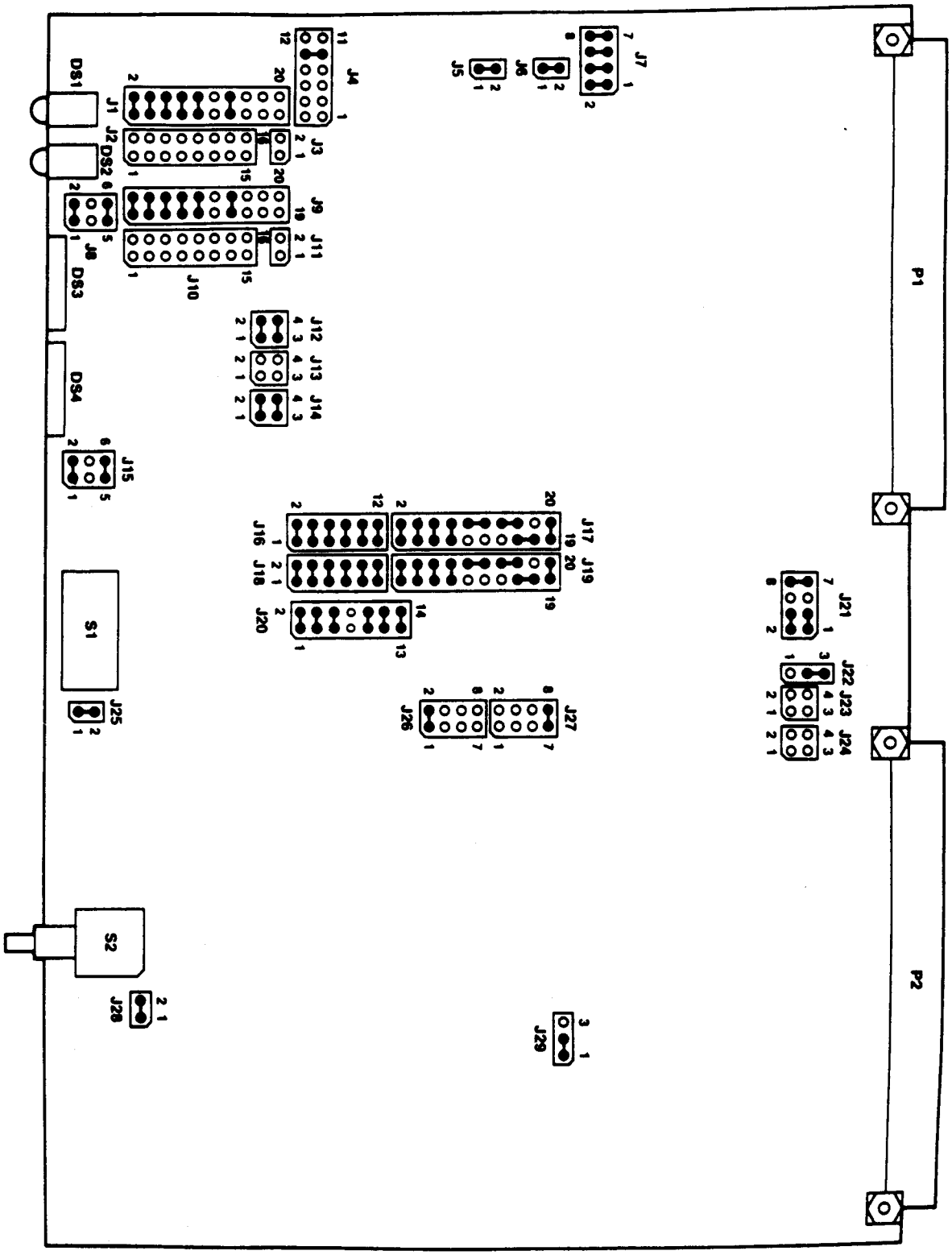


FIGURE 2-9. MVME050 Jumper Header Locations

TABLE 2-13. MVME050 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME050 VERSAdos JUMPER SETTINGS	
J1	EPROM BASE ADDRESS SELECT (A14-A23)	USER-CONFIGURED
J2	EPROM BASE ADDRESS SELECT (A24-A31)	USER-CONFIGURED
J8	EPROM SIZE SELECT	USER-CONFIGURED
J13	EPROM QUAD ENABLE SELECT	USER-CONFIGURED
J27	EPROM ACCESS TIME SELECT	USER-CONFIGURED
J9	RAM BASE ADDRESS SELECT (A14-A23)	USER-CONFIGURED
J10	RAM BASE ADDRESS SELECT (A24-A31)	USER-CONFIGURED
J26	RAM ACCESS TIME SELECT	USER-CONFIGURED
J14	RAM QUAD ENABLE SELECT	USER-CONFIGURED
J15	RAM SIZE SELECT	USER-CONFIGURED
J12	AM1E ENABLE SELECT (EPROM QUAD 1) AM16 ENABLE SELECT (EPROM QUAD 2)	USER-CONFIGURED
J3,J16,J17	EPROM/RAM CONFIGURATION SELECT (EPROM/RAM QUAD 1)	USER-CONFIGURED
J11,J18,J19	EPROM/RAM CONFIGURATION SELECT (EPROM/RAM QUAD 2)	USER-CONFIGURED
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, 7-8 9-10, 11-12, 13-14
J21	TIME-OF-DAY CLOCK POWER AND BATTERY CHARGE SELECT	1-2, 2-4, 7-8
J22	SYSTEM FAIL OR GROUND FOR INTERRUPT SOURCE SELECT	2-3
J23	INTERNAL/EXTERNAL TRANSMIT CLOCK SERIAL PORT 1 SELECT	NO JUMPER
J24	INTERNAL/EXTERNAL TRANSMIT CLOCK SERIAL PORT 2 SELECT	NO JUMPER

2

TABLE 2-13. MVME050 VERSAdos-Supported Jumper Settings (cont'd)

JUMPER NUMBER	DESCRIPTION	SETTING
J25	DISPLAY BLANKING ENABLE SELECT	1-2
J28	RESET SWITCH DISABLE SELECT	1-2
J29	PRINTER ACKNOWLEDGE EDGE SELECT	1-2
	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

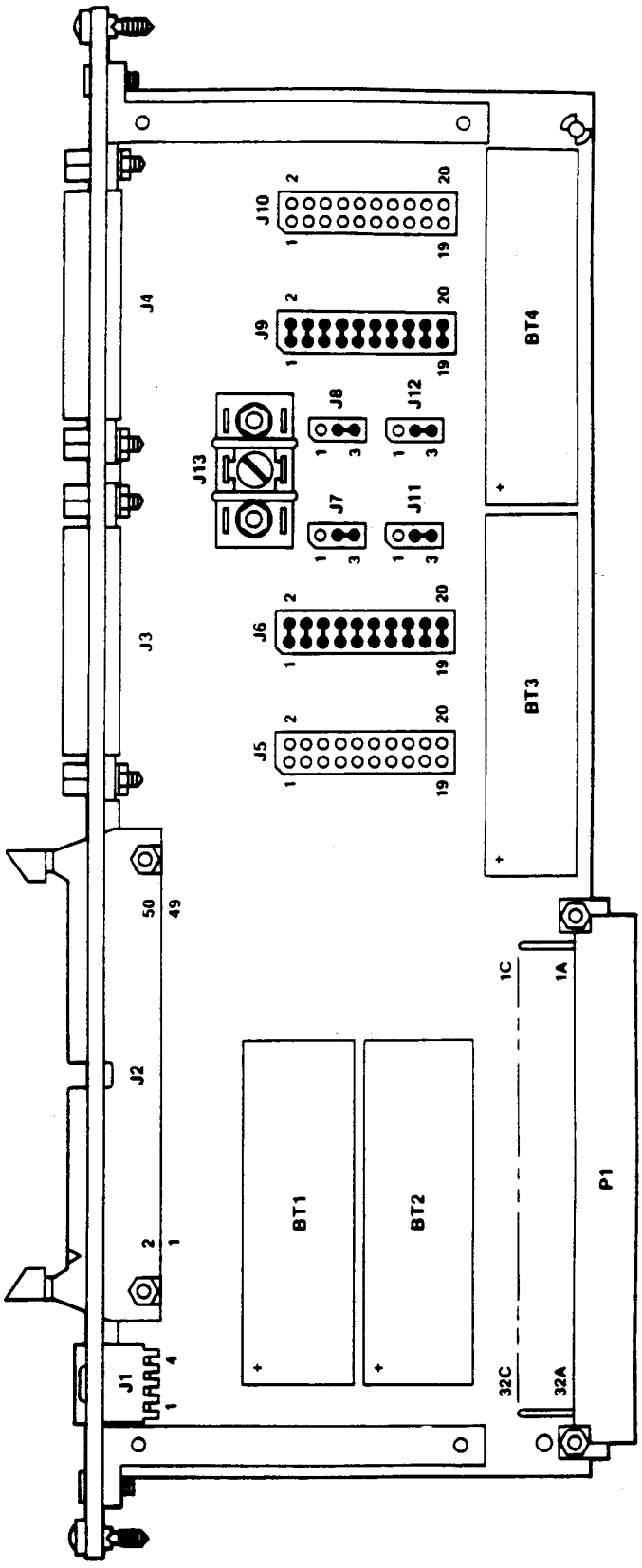


FIGURE 2-10. MVME701 Jumper Header Locations

2

TABLE 2-14. MVME701 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME701 VERSAdos JUMPER SETTINGS	
J5	PORT 1 TO TERMINAL SELECT	NO JUMPERS
J6	PORT 1 TO MODEM SELECT	1-2, 3-4, 5-6, 7-8 9-10,11-12, 13-14 15-16, 17-18, 19-20
J11	PORT 1 CTS SELECT	2-3
J7	PORT 1 DSR SELECT	2-3
J9	PORT 2 TO TERMINAL SELECT	1-2, 3-4, 5-6, 7-8 9-10, 11-12, 13-14 15-16, 17-18, 19-20
J10	PORT 2 TO MODEM SELECT	NO JUMPERS
J8	PORT 2 DSR SELECT	2-3
J12	PORT 2 CTS SELECT	2-3

CHAPTER 3

DYNAMIC MEMORY MODULES

3.1 MVME200/201 64K/256K BYTE DYNAMIC MEMORY MODULES

The MVME200 and MVME201 Dynamic RAM VMEmodules are designed to provide additional global RAM capacity in a VMEbus-based microcomputer system. Memory is addressable in byte or 16-bit word format. The modules feature odd parity error detection and asynchronous refresh circuitry. In addition, both modules allow memory block base addresses to be jumper-selectable and unused memory blocks to be jumper disabled.

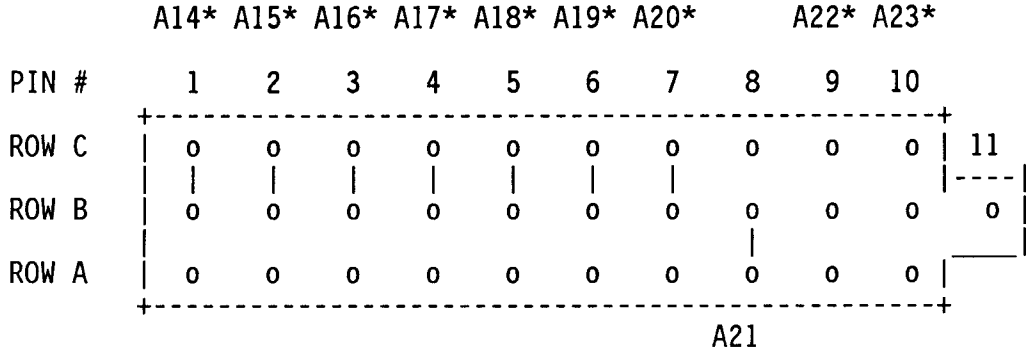
The MVME200 utilizes 16Kb dynamic RAM devices organized into four 16Kb blocks to create a total memory capacity of 64Kb. The MVME201 employs 64Kb devices in four 64Kb blocks to achieve 256Kb of memory.

3.1.1 Standard VERSAdos Configuration

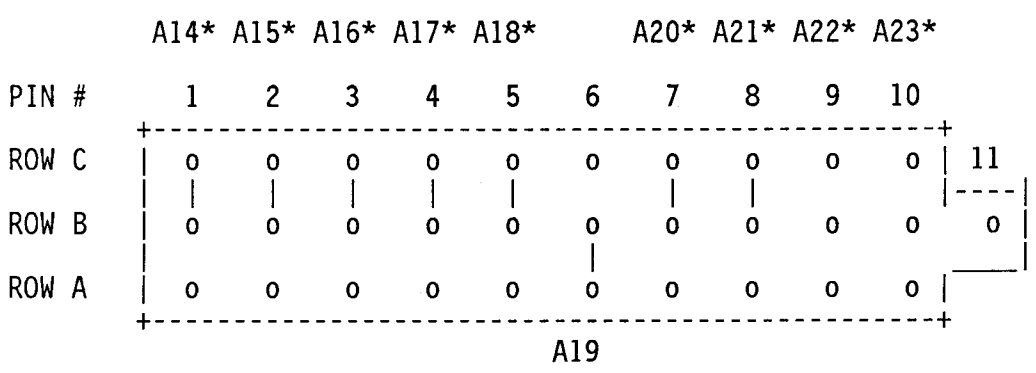
Figure 3-1 depicts the jumper locations for the MVME200 and MVME201 memory modules. Table 3-1 lists the jumper settings required to operate VERSAdos using the modules. Refer to the MVME200/201 64K/256K Byte Dynamic Memory Module User's Manual (MVME200) to compare these settings with the factory-set jumper configurations.

3.1.2 MVME201 Memory Address Settings

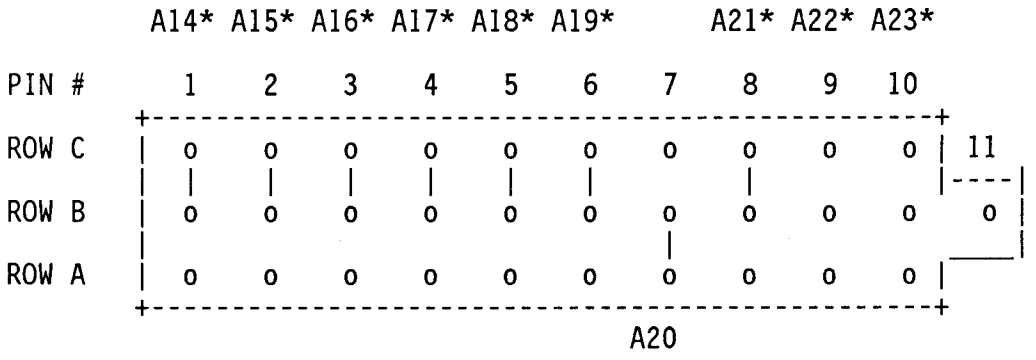
The MVME201 module is organized into four 64Kb blocks of memory. Each block is addressable on any 64Kb boundary within the 16Mb address space. The following diagrams illustrate these blocks for various VMEmodules.



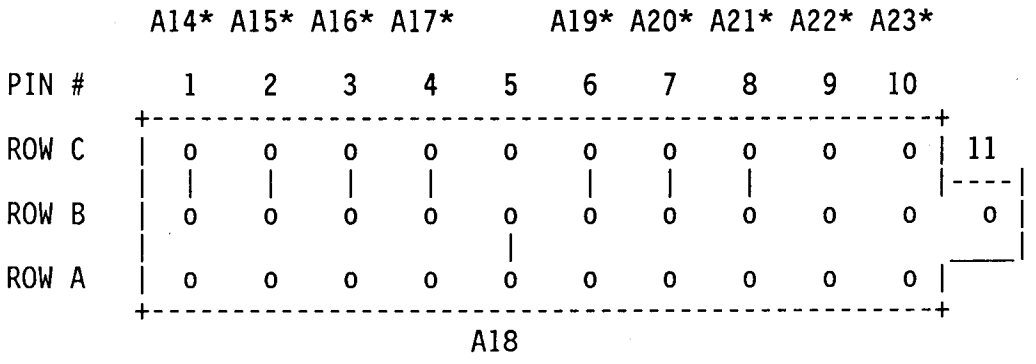
Block Addressed at $10000 (MVME101)



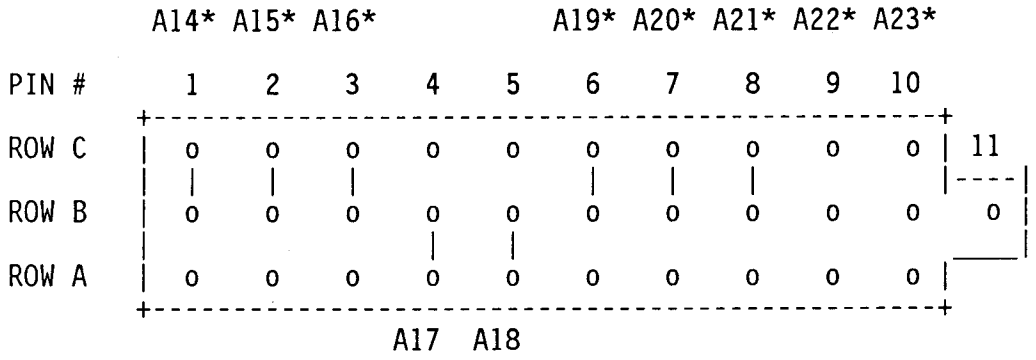
Block Addressed at $40000 (MVME110-1)



Block Addressed at $20000 (MVME120, MVME122)



Block Addressed at $80000 (MVME121, MVME123)



Block Addressed at $180000 (VME/10)

3.2 MVME202/222-1/222-2 512K/1M/2M BYTE DYNAMIC RAM MODULES

The MVME202, MVME222-1, and MVME222-2 are dynamic RAM VMEmodules used in VMEbus-based microcomputer systems to increase global memory. The modules use an 18-bit row organization to implement both byte and word accessing and upper or lower byte parity generation and checking.

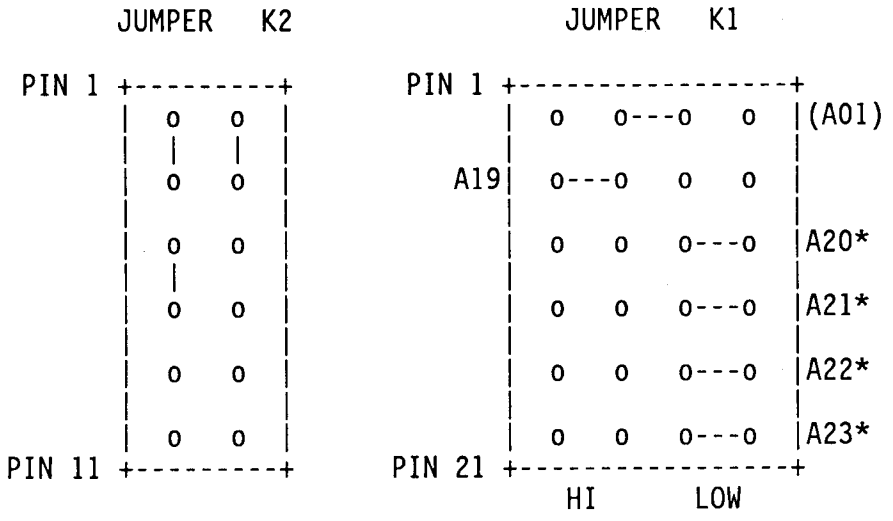
The four 18-device rows on the MVME202 hold 64K-bit memories providing a capacity of 512Kb. On the MVME222-1, two 18-device rows are filled with 256K-bit memories providing 1Mb of memory. The four 18-device rows on the MVME222-2 are filled with 256K-bit memories to achieve 2Mb of memory.

3.2.1 Standard VERSAdos Configuration

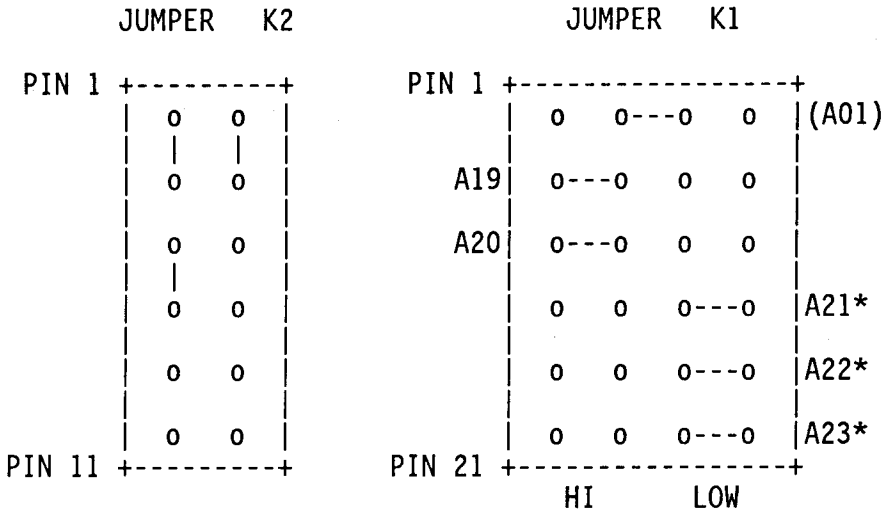
The standard VERSAdos-supported jumper configuration is shown in Figure 3-2. The jumper settings are listed in Table 3-2. Refer to the MVME202/222-1/222-2 512K/1M/2M Byte Dynamic RAM User's Manual (MVME202) to compare these settings with the factory-set configurations.

3.2.2 MVME202/222 Memory Address Settings

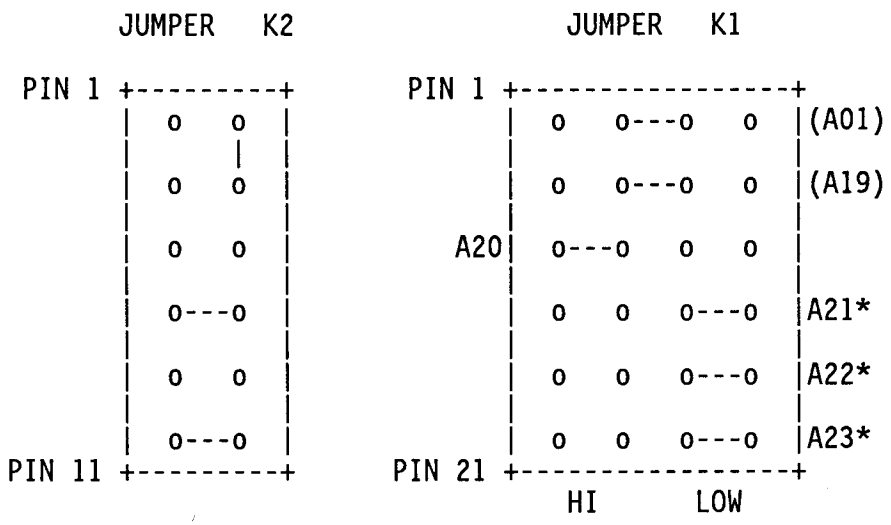
The 512Kb MVME202 memory module is organized as one 512Kb block of memory addressable on any 512Kb boundary within the 16Mb address space. The 1Mb MVME222-1 memory module is organized as one 1Mb block of memory addressable on any 1Mb boundary within the 16Mb address space. Similarly, the 2Mb MVME222-2 memory module is organized as one 2Mb block of memory addressable on any 2Mb boundary within the 16Mb address space. The following diagrams illustrate these blocks for various VMEmodules.



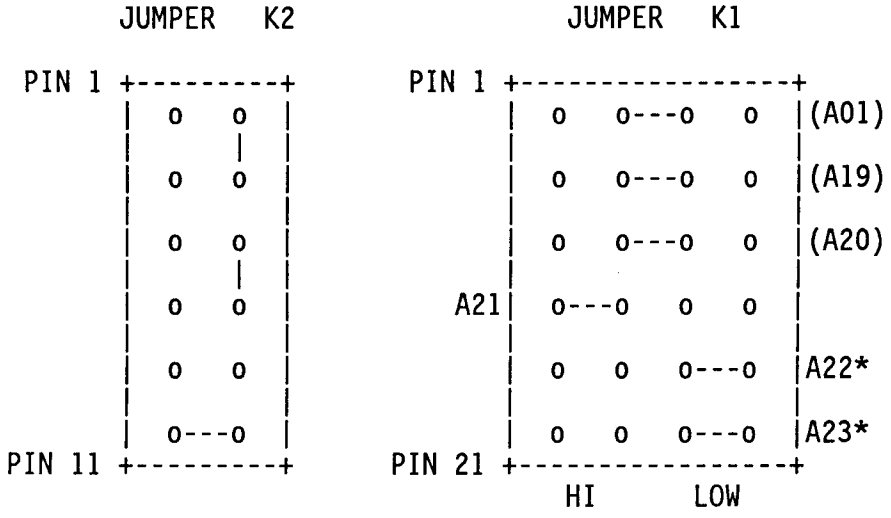
MVME202 Block Addressed at $80000 (MVME121, MVME123)



MVME202 Block Addressed at $180000 (VME/10)



MVME222-1 Block Addressed at $100000 (MVME121, MVME123)



MVME222-2 Block Addressed at $200000

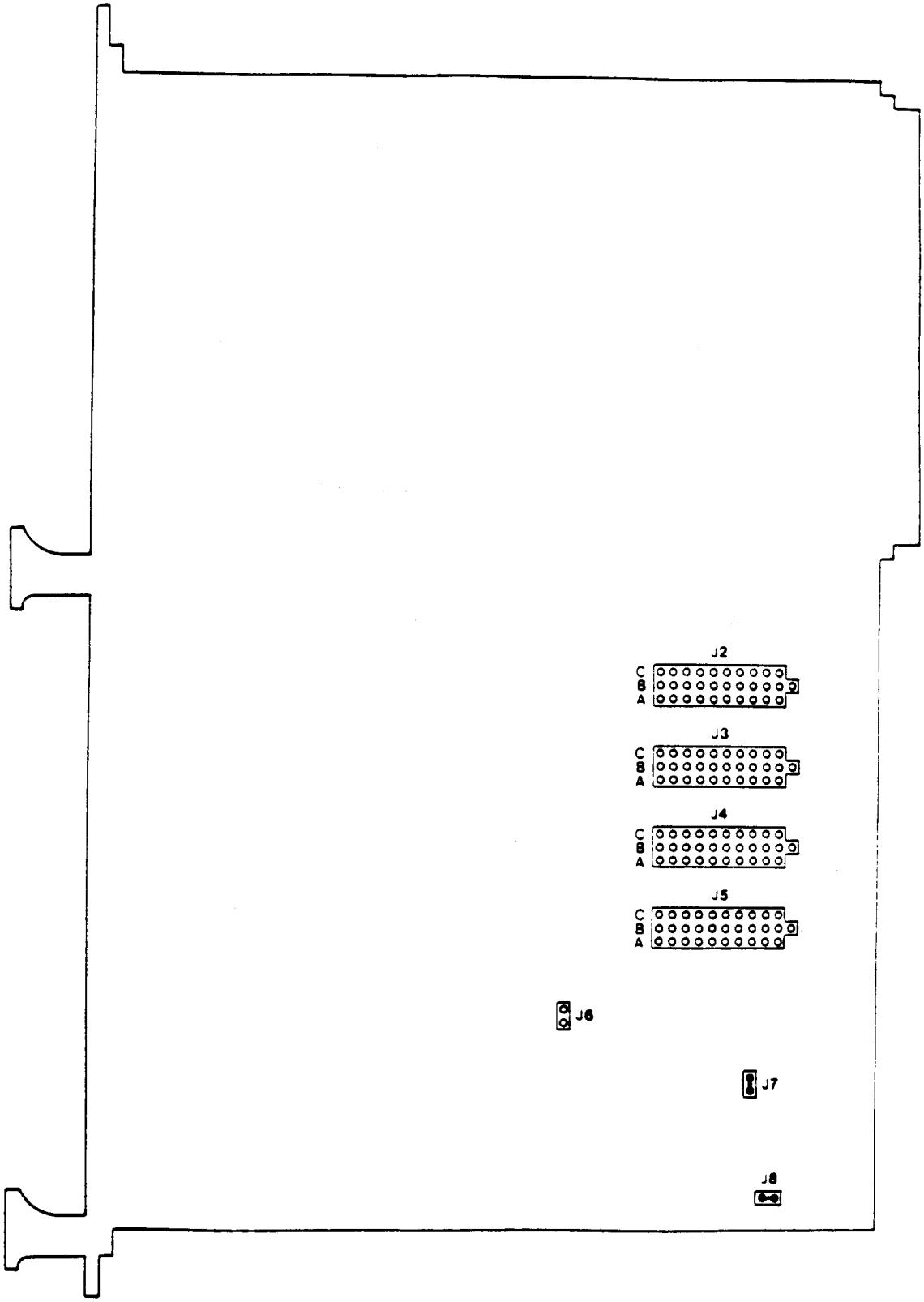


FIGURE 3-1. MVME200/201 Jumper Header Locations

TABLE 3-1. MVME200/201 VERSAdos Jumper and Backplane Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME201 VERSAdos JUMPER SETTINGS	
J2	BLOCK 1 BASE ADDRESS SELECT	USER-CONFIGURED
J3	BLOCK 2 BASE ADDRESS SELECT	USER-CONFIGURED
J4	BLOCK 3 BASE ADDRESS SELECT	USER-CONFIGURED
J5	BLOCK 4 BASE ADDRESS SELECT	USER-CONFIGURED
J6	LONGWORD DISABLE SELECT	NO JUMPER
J7	BUS ERROR DISABLE SELECT	1-2
J8	REFRESH DISABLE SELECT	1-2
	MVME201 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

3

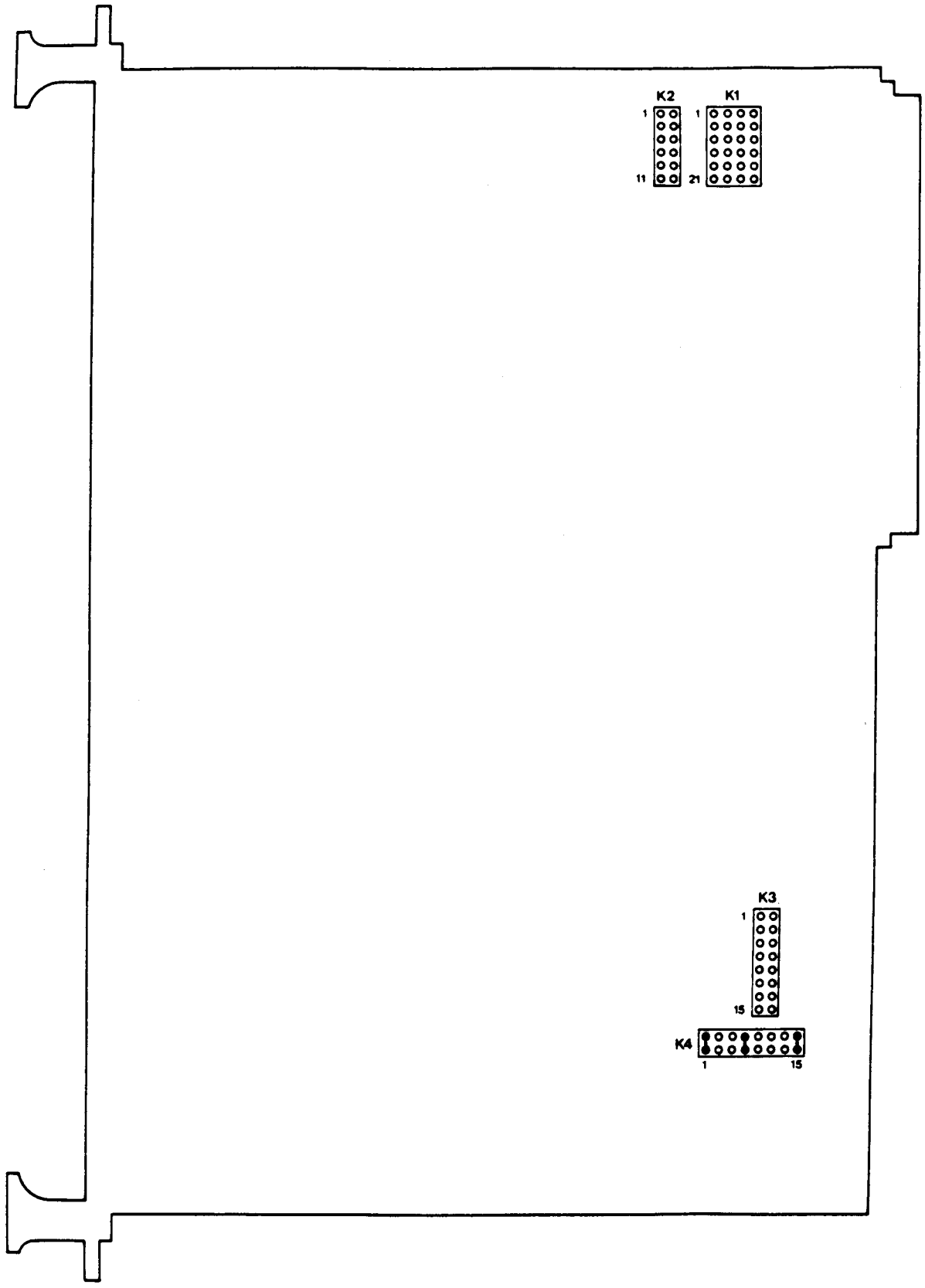


FIGURE 3-2. MVME202/222 Jumper Header Locations

TABLE 3-2. MVME202/222 VERSAdos Jumper and Backplane Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME202 VERSAdos JUMPER SETTINGS	
K1	BASE ADDRESS SELECT	USER-CONFIGURED
K2	MEMORY CAPACITY AND ADDRESSING MODE SELECT	USER-CONFIGURED
K3	TEST CONNECTOR	NO JUMPERS
K4	ACCESS TIME SELECT	1-2, 7-8, 15-16
	MVME202 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

3

3.3 MVME210 RAM/ROM/EPROM MEMORY MODULE

The MVME210 RAM/ROM/EPROM Memory Module provides users with a VMEbus-compatible board containing sixteen 28-pin sockets for installation of up to 128Kb of various types of byte-wide memory devices. Module sockets are organized into four word-oriented memory blocks and are capable of being populated with 1Kb, 2Kb, 4Kb or 8Kb RAM, ROM, or EPROM devices.

The MVME210 is jumper-selectable for device type and device access time. In addition, the module is capable of jumper disabling for unused memory blocks.

3.3.1 Standard VERSAdos Configuration

Figure 3-3 illustrates the jumper header locations for the MVME210 RAM/ROM/EPROM Memory Module. Table 3-3 lists the jumpers and their settings required to operate VERSAdos. Refer to the MVME210 RAM/ROM/EPROM Memory Module User's Manual (MVME210) to compare these settings with the factory-set configuration.

3

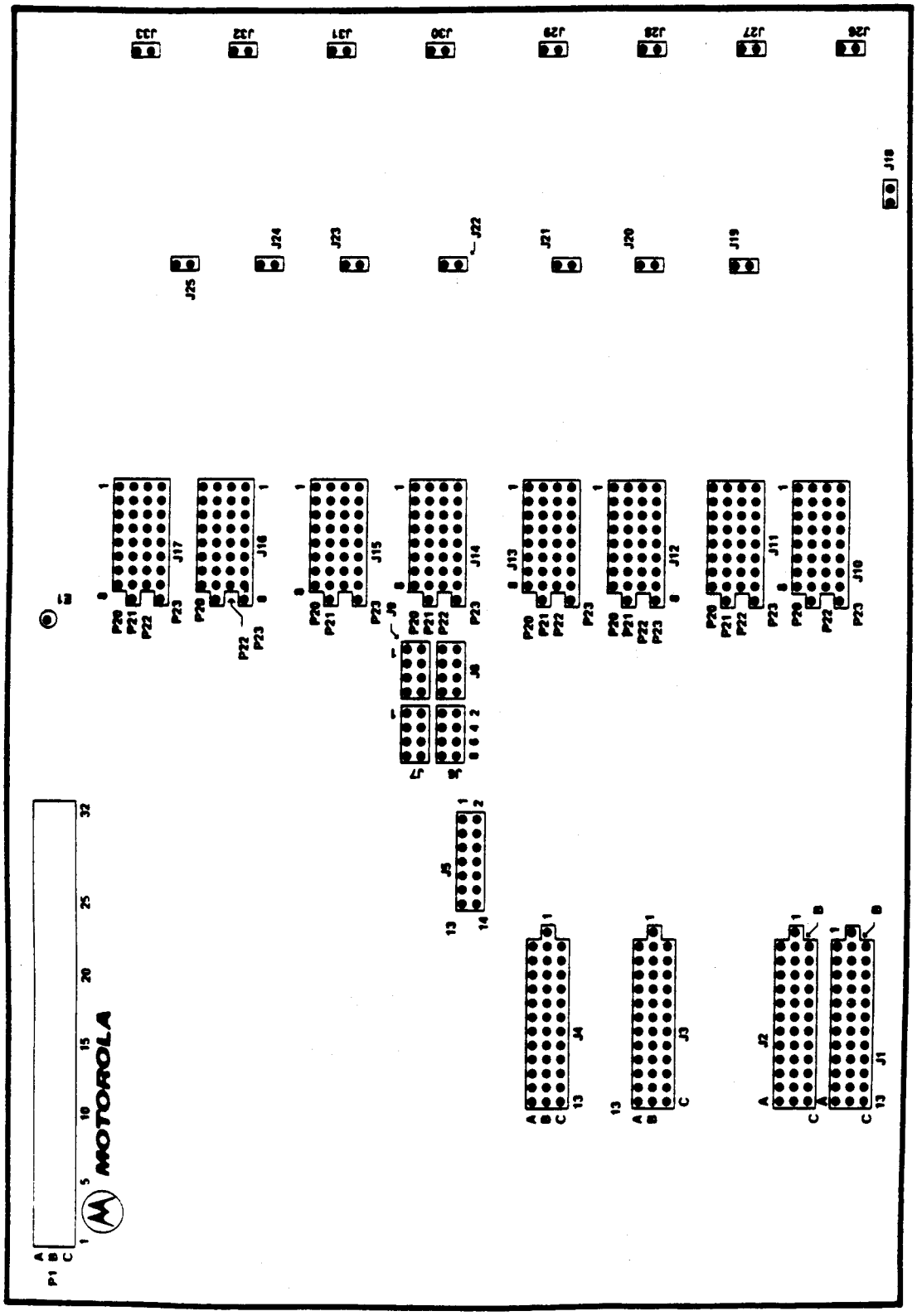


FIGURE 3-3. MVME210 Jumper Header Locations

TABLE 3-3. MVME210 VERSAdos Jumper and Backplane Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME210 VERSAdos JUMPER SETTINGS	
J1	BLOCK 4 BASE ADDRESS SELECT	USER-CONFIGURED
J9	BLOCK 4 SIDE SELECT	USER-CONFIGURED
J11	BLOCK 4 LOW WORD DEVICE SELECT	USER-CONFIGURED
J10	BLOCK 4 HIGH WORD DEVICE SELECT	USER-CONFIGURED
J3	BLOCK 3 BASE ADDRESS SELECT	USER-CONFIGURED
J8	BLOCK 3 SIDE SELECT	USER-CONFIGURED
J13	BLOCK 3 LOW WORD DEVICE SELECT	USER-CONFIGURED
J12	BLOCK 3 HIGH WORD DEVICE SELECT	USER-CONFIGURED
J2	BLOCK 2 BASE ADDRESS SELECT	USER-CONFIGURED
J6	BLOCK 2 SIDE SELECT	USER-CONFIGURED
J15	BLOCK 2 LOW WORD DEVICE SELECT	USER-CONFIGURED
J14	BLOCK 2 HIGH WORD DEVICE SELECT	USER-CONFIGURED
J4	BLOCK 1 BASE ADDRESS SELECT	USER-CONFIGURED
J7	BLOCK 1 SIDE SELECT	USER-CONFIGURED
J17	BLOCK 1 LOW WORD DEVICE SELECT	USER-CONFIGURED
J16	BLOCK 1 HIGH WORD DEVICE SELECT	USER-CONFIGURED
J5	ACCESS TIME SELECT	USER-CONFIGURED
J18-J33	RAM OR ROM SELECT	USER-CONFIGURED
	MVME210 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

3.4 MVME211 RAM/ROM/EPROM MEMORY MODULE

The MVME211 RAM/ROM/EPROM Memory Module provides users with a VMEbus-compatible board containing sixteen 28-pin sockets for installation of up to 1Mb of various types of byte-wide memory devices. Module sockets are organized into two word-oriented memory blocks and are capable of being populated with 2Kb, 4Kb, 8Kb, 16Kb, 32Kb, or 64Kb JEDEC-standard RAM, ROM, or EPROM devices.

The MVME211 is jumper-selectable for device type, device access time, and battery back-up standby power. The module also features jumper disabling for unused memory blocks.

3.4.1 Standard VERSAdos Configuration

Figure 3-4 illustrates the jumper header locations of the MVME210 RAM/ROM/EPROM Memory Module. Table 3-4 lists the jumpers and the settings required to boot VERSAdos on a system incorporating the module.

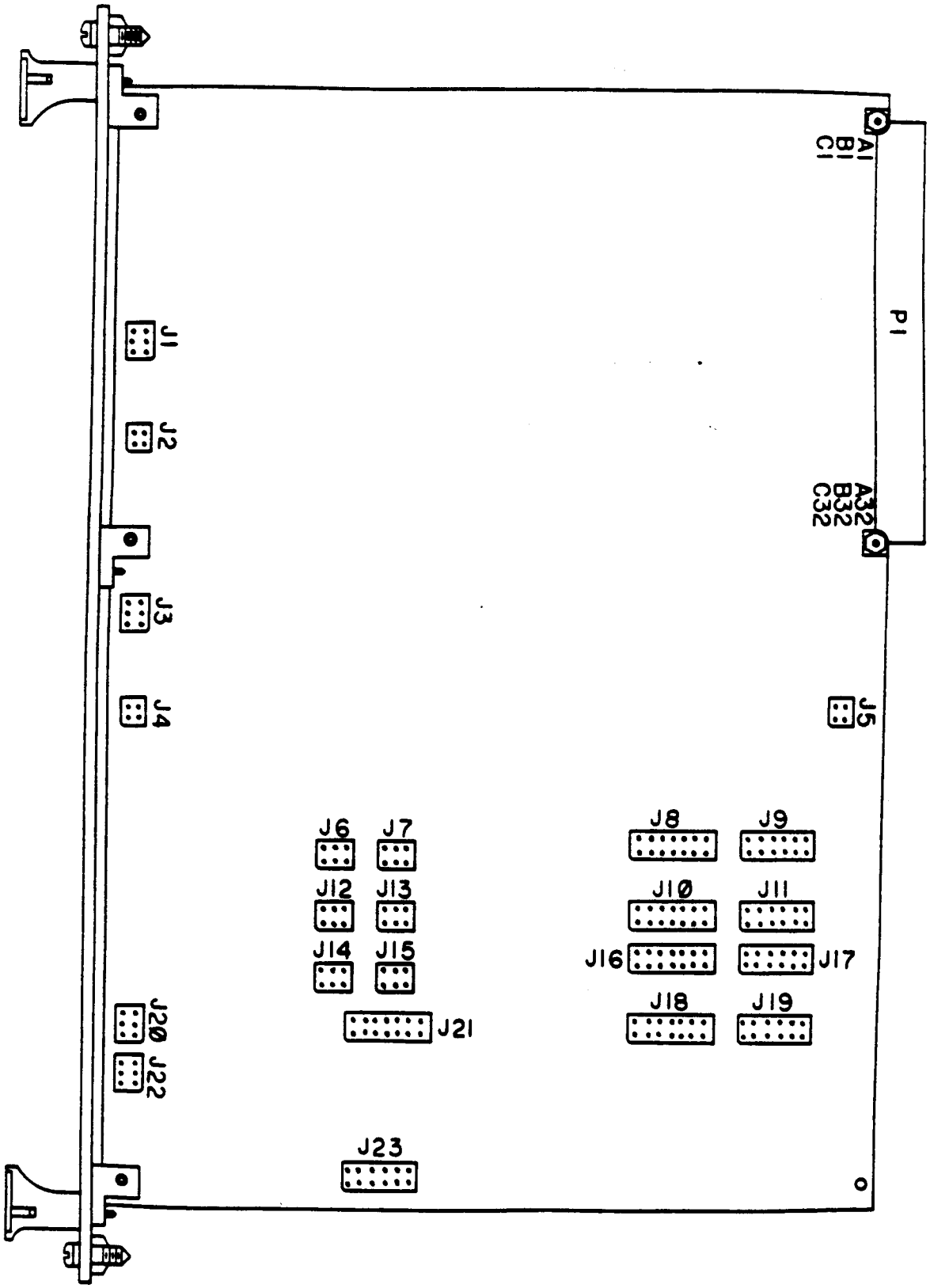


FIGURE 3-4. MVME211 Jumper Header Locations

TABLE 3-4. MVME211 VERSAdos Jumper and Backplane Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME211 VERSAdos JUMPER SETTINGS	
J1,J2,J6 J12,J14	BLOCK 1 SIGNAL SELECT	USER-CONFIGURED
J3,J4,J7 J13,J15	BLOCK 2 SIGNAL SELECT	USER-CONFIGURED
J5	STANDBY POWER BATTERY BACKUP SELECT	USER-CONFIGURED
J8-J11	BLOCK 1 BASE ADDRESS SELECT	USER-CONFIGURED
J16-J19	BLOCK 2 BASE ADDRESS SELECT	USER-CONFIGURED
J22	BLOCK 1 CHIP SIZE SELECT	USER-CONFIGURED
J20	BLOCK 2 CHIP SIZE SELECT	USER-CONFIGURED
J21,J23	ACCESS TIME SELECT	USER-CONFIGURED
	MVME211 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

CHAPTER 4

MASS STORAGE CONTROLLERS

4.1 MVME315 INTELLIGENT DISK CONTROLLER MODULE

The MVME315 Intelligent Disk Controller Module combines in one double-high Eurocard VMEmodule a multiple floppy disk controller, a SASI bus interface. The module is designed for use in applications having intensive I/O or multiprocessor structures to reduce VMEbus traffic and to increase system throughput and mass storage capability.

The MVME315 controls up to four 8-inch or 5-1/4 inch floppy disk drives in any mixed configuration and serves as an intelligent interface to the SASI bus. The Universal Intelligent Peripheral Controller (UIPC) is equipped with an MC68121 Intelligent Peripheral Controller, a Direct Memory Access (DMA) controller, 32Kb of dual-ported RAM for program data and storage of up to 12Kb of floppy and hard disk I/O data, and a full VMEbus interface featuring a VMEbus request and interrupt controller.

4.1.1 MVME315 Standard VERSAdos Configuration

Figure 4.1 illustrates the jumper header locations of the MVME315 module. The jumper settings required to support VERSAdos are listed in Table 4.1. Refer to the MVME315 Intelligent Disk Controller Module User's Manual (MVME315) to compare these jumper settings with the factory-set configuration.

4.2 MVME320 DISK CONTROLLER MODULE

The MVME320 Disk Controller Module is a double-high Eurocard VMEmodule for adding mass storage capability to a VMEbus-based system. The module provides high-performance DMA channels between system memory and Winchester hard disk drives and/or floppy disk drives. The MVME320 is intended for applications having intensive real-time disk I/O or multiprocessing structures designed to reduce VMEbus traffic and increase system throughput.

The MVME320 controls mixed 5-1/4 inch and 8-inch disk drives. The module can simultaneously control up to two Winchester hard disks and two floppy disks or up to four floppy disk drives. The MVME320 supports single- and double-sided (single- and double-density) disks. All drives connected to the MVME320 must be soft-sectored.

VERSAdos is released on double-sided, single-density, Motorola formatted 8-inch diskettes which are incompatible with the IBM format supported by the MVME320. Software purchased on 8-inch diskettes must be converted to a single-sided, single-density format before it can be read by a MVME320 Disk Controller.

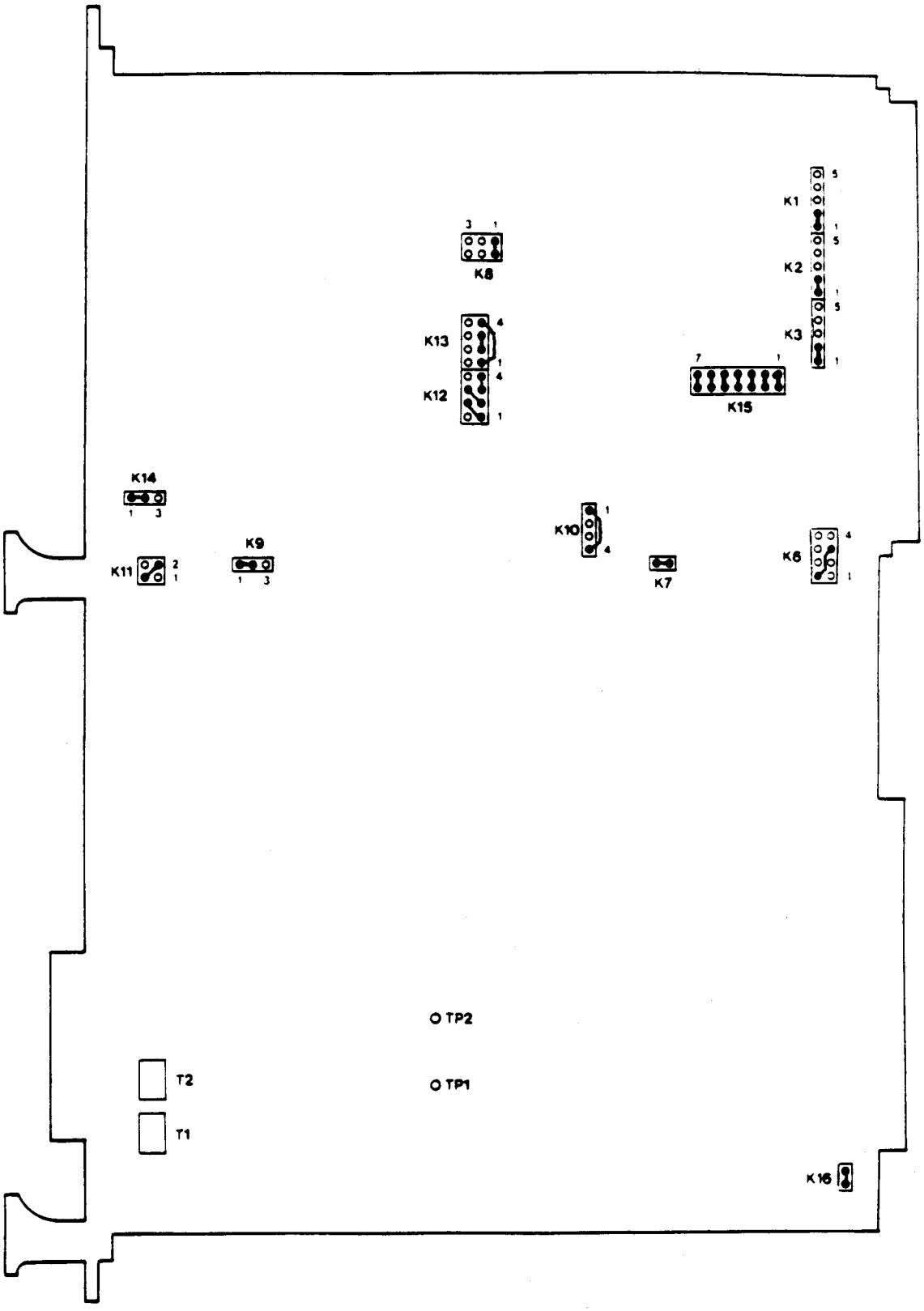


FIGURE 4-1. MVME315 Jumper Header Locations

TABLE 4-1. MVME315 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME315 VERSAdos JUMPER SETTINGS	
K1	VMEBUS GRANT IN SELECT	1-2
K2	VMEBUS GRANT OUT SELECT	1-2
K3	VMEBUS REQUEST LEVEL SELECT	1-2
K6	VMEBUS INTERRUPT REQUEST LEVEL SELECT	3-8
K8	VMEBUS INTERRUPT REQUEST LEVEL SELECT	1-6
K15	COMMAND CHANNEL BASE ADDRESS SELECT	1-14, 2-13, 3-12, 4-11, 5-10, 6-9, 7-8
K7	SYSTEM FAIL ENABLE SELECT	1-2
K16	DRIVE 3 SELECT ENABLE SELECT	1-2
K10	INTERRUPT CONTROLLER CONFIGURATION	1-4
K9	LOCAL RAM CONFIGURATION	1-2
K11	LOCAL RAM CONFIGURATION	2-4
K12	LOCAL ROM CONFIGURATION	1-7, 2-6, 3-4
K13	LOCAL ROM CONFIGURATION	1-4, 2-3
K14	LOCAL ROM CONFIGURATION	1-2
	MVME315 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

4.2.1 MVME320 Standard VERSAdos Configuration

Figure 4-2 details the jumper header locations of the MVME320 Disk Controller module. Table 4-2 lists the jumper settings required to operate VERSAdos in a system employing the MVME320. Refer to the MVME320 Disk Controller User's Manual (MVME320) to compare these settings with the factory-set configuration.

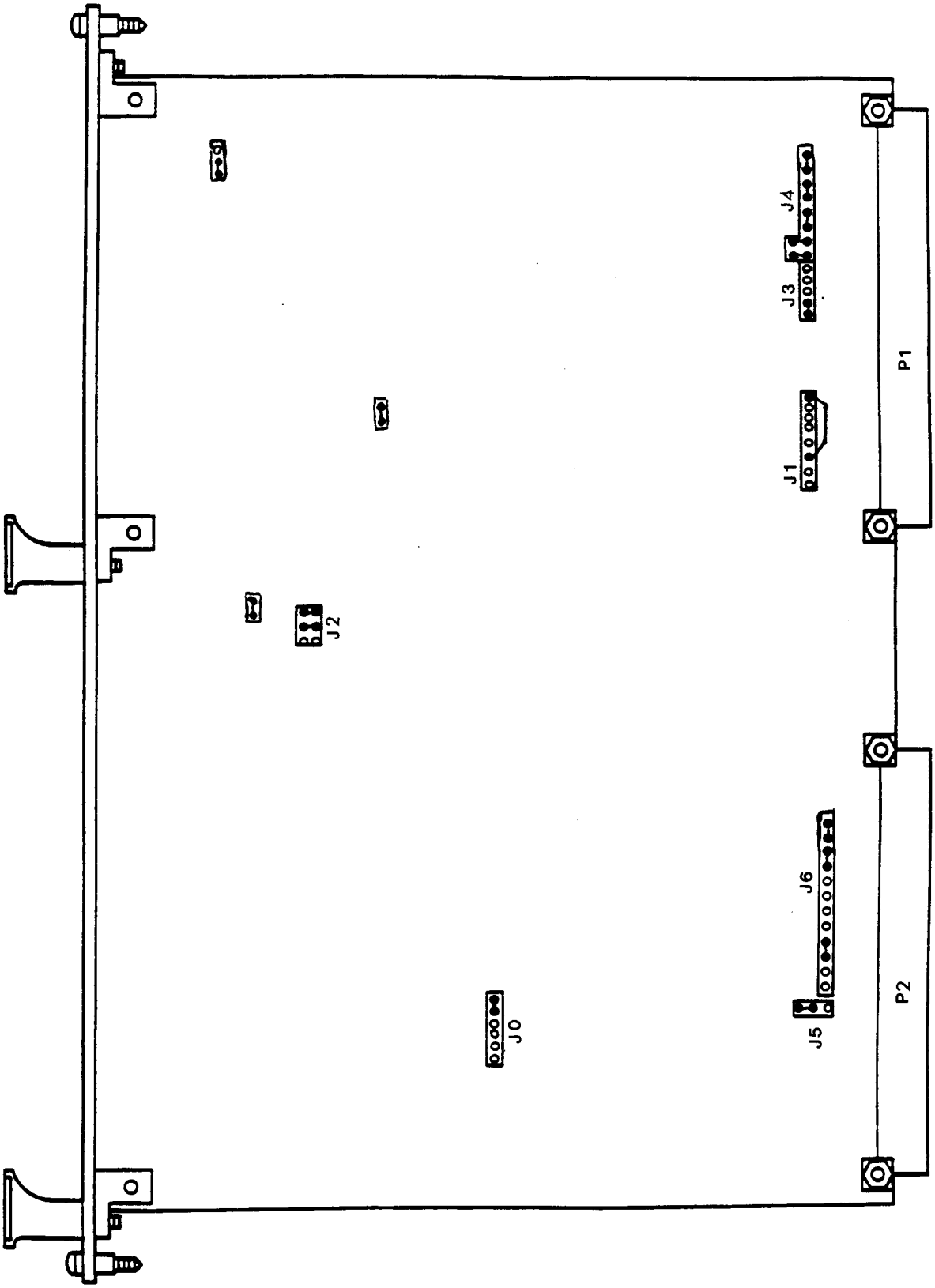


FIGURE 4-2. MVME320 Jumper Header Locations

TABLE 4-2. MVME320 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME320 VERSAdos JUMPER SETTINGS	
JW0	PROM SIZE SELECT	5-6
JW1	INTERRUPT REQUEST LEVEL SELECT	3-8
JW2	INTERRUPT ACKNOWLEDGE LEVEL SELECT	2-5, 3-6
JW3	BUS REQUEST LEVEL SELECT	1-2
JW4	BUS GRANT LEVEL SELECT	1-9, 2-10, 3-4 5-6, 7-8
JW5	ADDRESS MODIFIER SELECT	2-3
JW6	MEMORY ADDRESS SELECT	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

4.2.2 MVME702 Disk Interface Module

The MVME702 Disk Interface Module is a double-high VMEmodule that is designed to function as an extension of the MVME320 Disk Controller Module. The MVME702 performs signal switching and provides standard disk cable connections for different drives.

4.2.2.1 MVME702 Standard VERSAdos Configuration. Figure 4-3 illustrates the jumper header locations of the MVME702 Disk Interface Module. Table 4-3 lists the MVME702 jumper settings required to operate VERSAdos with the MVME702. Refer to the MVME702 Disk Interface Module User's Manual (MVME702) to compare these settings with the factory-set configuration.

4

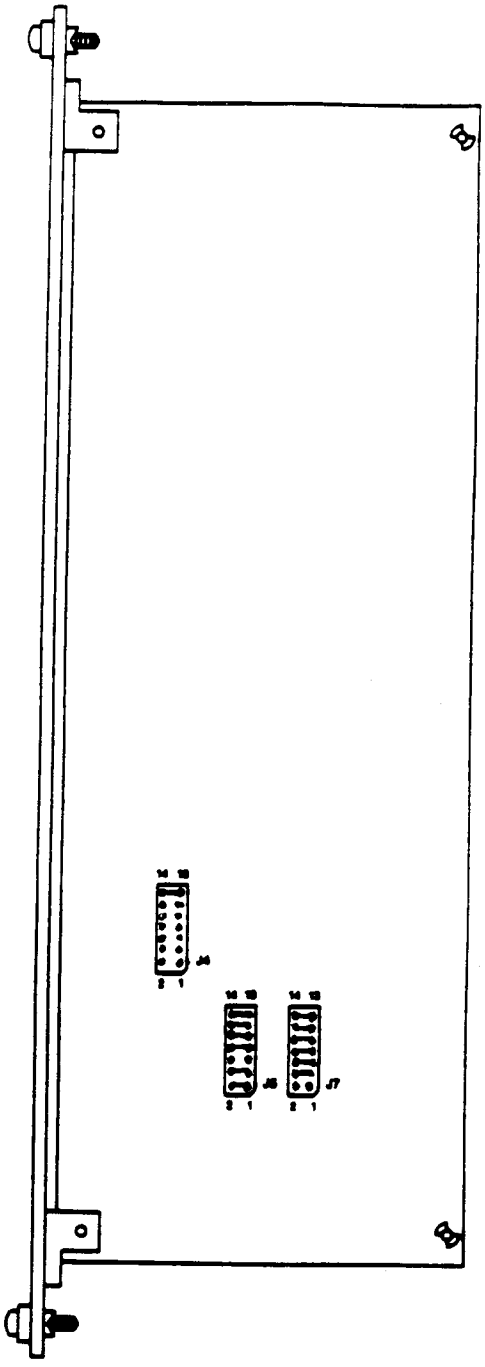


FIGURE 4-3. MVME702 Jumper Header Locations

TABLE 4-3. MVME702 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME702 VERSAdos JUMPER SETTINGS	
J4	SIGNAL TERMINATION SELECT	13-14
J5	SIGNAL TERMINATION SELECT	1-2, 3-4, 7-8, 9-10, 11-12, 13-14
J6	SIGNAL TERMINATION SELECT	3-4, 5-6, 7-8, 9-10, 11-12, 13-14

4.3 MVME420 SASI PERIPHERAL ADAPTER

The MVME420 SASI Peripheral Adapter provides an interface between a microcomputer I/O Channel and a Shugart Associates SASI bus. The SASI bus supports the single-host, non-arbitrating, SA1400 disk controller implementation. The MVME420 is a single Eurocard VMEmodule that appears as eight 1-byte read/write registers on the I/O Channel address map. The base address is jumper-selectable. In addition, a maskable interrupt is jumper-selectable to any of three I/O Channel interrupt priority levels. Diagnostic read/write capability is provided to verify I/O Channel data paths.

The MVME420 supports both 5-1/4 inch and 8-inch diskette media. However, VERSAdos is released on double-sided, single-density, Motorola-formatted 8-inch diskettes which are incompatible with the IBM format supported by the MVME420. Software purchased on 8-inch diskettes must be converted to a single-sided, single-density format before it can be used with an MVME420 module.

4.3.1 MVME420 Standard VERSAdos Configuration

Figure 4-4 illustrates the jumper header locations of the MVME420 SASI Adapter Module. Table 4-4 lists the jumper settings required to operate VERSAdos using the MVME420. Refer to the MVME420 SASI Peripheral Adapter User's Manual (MVME420) to compare these settings with the factory-set configuration.

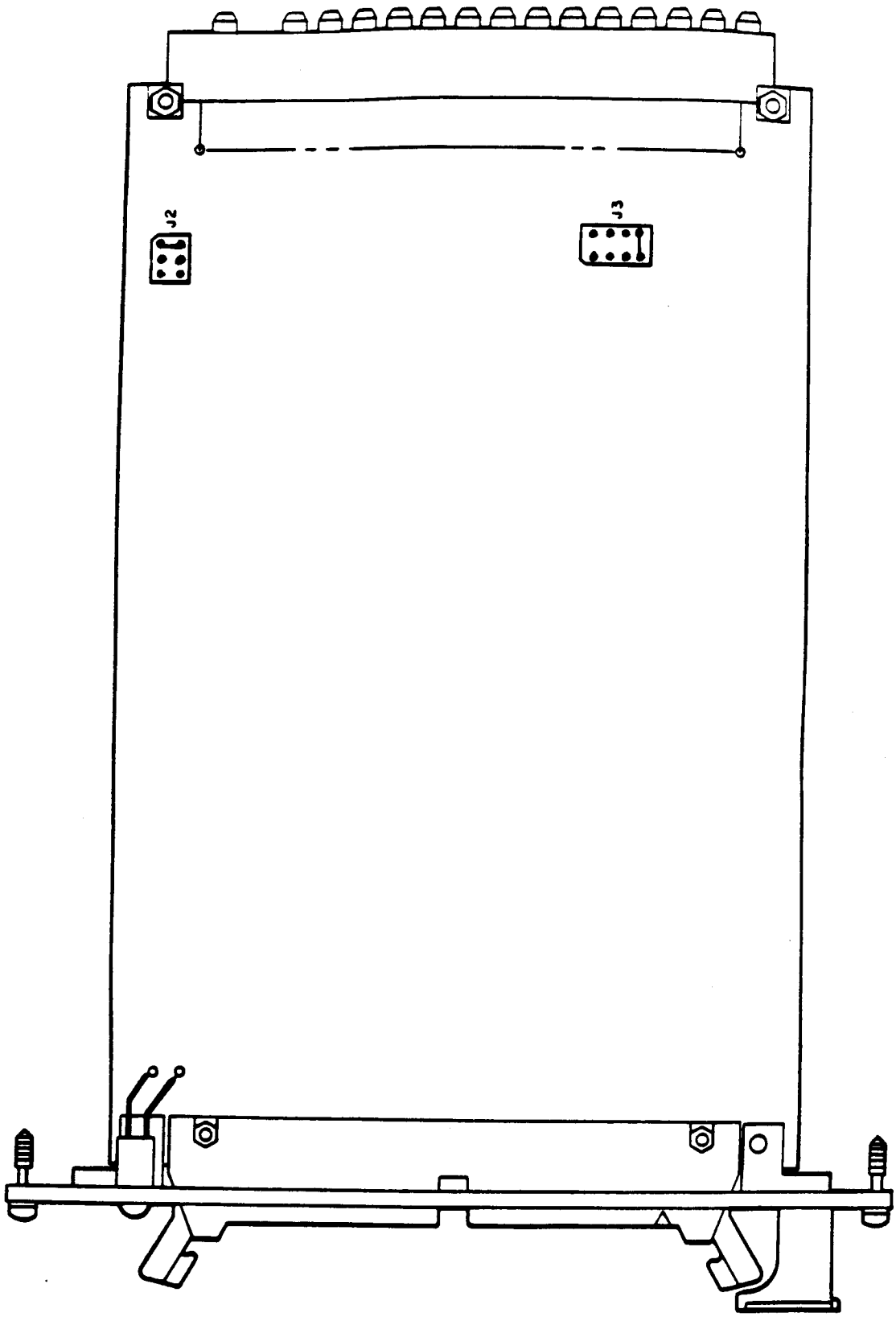


FIGURE 4-4. MVME420 Jumper Header Locations

TABLE 4-4. MVME420 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME420 VERSAdos JUMPER SETTINGS	
J2	INTERRUPT LEVEL SELECT	1-2
J3	BASE ADDRESS SELECT	7-8

4.4 M68RWIN1 WINCHESTER DISK CONTROLLER

The M68RWIN1 Winchester Disk Controller is an intelligent interface between the Motorola I/O Channel and up to two Winchester disk drives and two floppy disk drives. The module is designed to operate with any host which uses an I/O Channel bus. The M68RWIN1 is available in two configurations -- one for 8-inch (SA1000-compatible) drives and one for 5-1/4 inch (ST500-compatible) drives.

The M68RWIN1 performs high-level commands such as read/write data, format track, verify data, etc., and includes implied seeks, automatic track/head switching, alternate sectoring, and error checking. Data is sent between the host and the M68RWIN1 through interrupt-initiated block transfers, interrupt-initiated byte transfers, or by processor polling.

4.4.1 M68RWIN1 Standard VERSAdos Configuration

Figure 4-5 illustrates the jumper header locations of the M68RWIN1 Winchester Disk Controller module. Table 4-5 lists the jumper settings of the M68RWIN1 required to operate VERSAdos. Refer to the M68RWIN1 Winchester Disk Controller User's Manual (M68RWIN1) to compare these settings with the factory-set configuration.

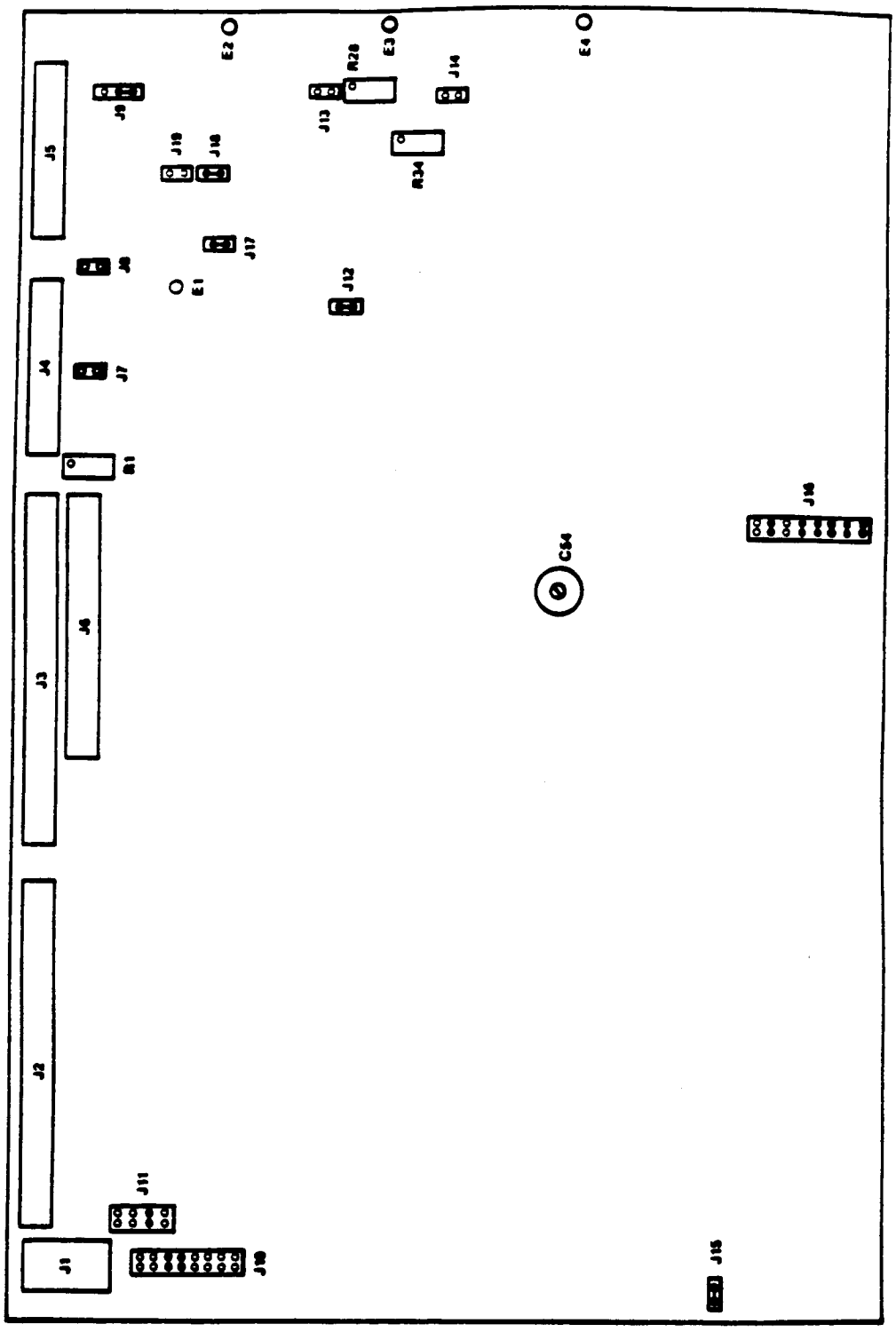


FIGURE 4-5. M68RWIN1 Jumper Header Locations

TABLE 4-5. M68RWIN1 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	RWIN VERSAdos JUMPER SETTINGS	
J7	WINCHESTER DATA TERMINATION SELECT	NO JUMPER
J8		NO JUMPER
J9	POWER CONNECTION SELECT	2-3
J10	I/O CHANNEL INTERRUPT SELECT	5-6, 7-8
J11	I/O CHANNEL ADDRESS SELECT	5-6
J16	WINCHESTER & FLOPPY DISK DRIVE PARAMETERS SELECT	3-4, 7-8, 9-10, 11-12, 13-14, 15-16
J12		1-2
J13		NO JUMPER
J14		NO JUMPER
J15		1-2
J17		1-2
J18		1-2
J19		NO JUMPER

4

4

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 5

I/O CHANNEL MODULES

5.1 MVME316 VMEBUS TO I/O CHANNEL INTERFACE

The MVME316 VMEbus to I/O Channel Interface processes address, data, interrupt, and control signals to expand the resources of the VMEbus and provide a complete interface from the VMEbus to the I/O Channel. The MVME316 is designed to operate with any I/O Channel module available from Motorola Microsystems.

The module features programmable interrupt vectors and VMEbus time-out capability. Base addresses for the interrupter and the I/O Channel are jumper-selectable. In addition, the MVME316 supports daisy-chained interrupt acknowledgements and a software-controlled I/O Channel reset mechanism.

Two types of operations are possible through the MVME316 to the I/O Channel -- normal polling and interrupt handling routines. Data transfer may be done using a normal polling operation to an I/O Channel access. Read or write instructions access odd bytes of the VMEbus memory map dedicated to the I/O Channel.

5.1.1 MVME316 Standard VERSAdos Configuration

Figure 5-1 depicts the jumper header locations of the MVME316 module. Table 5-1 lists the jumper settings required to operate VERSAdos in a system employing the MVME316. Refer to the VMEbus to I/O Channel Interface User's Manual (MVME316) to compare these settings with the factory-set configuration.

5.2 MVME400 DUAL RS-232C SERIAL PORT MODULE

The MVME400 Dual RS-232C Serial Port Module is an I/O Channel-compatible dual RS-232C serial interface module designed to expand the resources of an I/O Channel master to include one or two additional RS-232C serial data ports. The module conforms to a single 64-pin DIN 41612 standard connector; however, because of the wide front panel, the MVME400 occupies two module card slots.

5.2.1 Standard VERSAdos Configuration

Figure 5-2 illustrates the jumper header locations of the MVME400. The jumper settings required to operate VERSAdos in a standard configuration are listed in Table 5-2. Refer to the Dual RS-232C Serial Port Module User's Manual (MVME400) to compare these settings with the factory-set configuration.

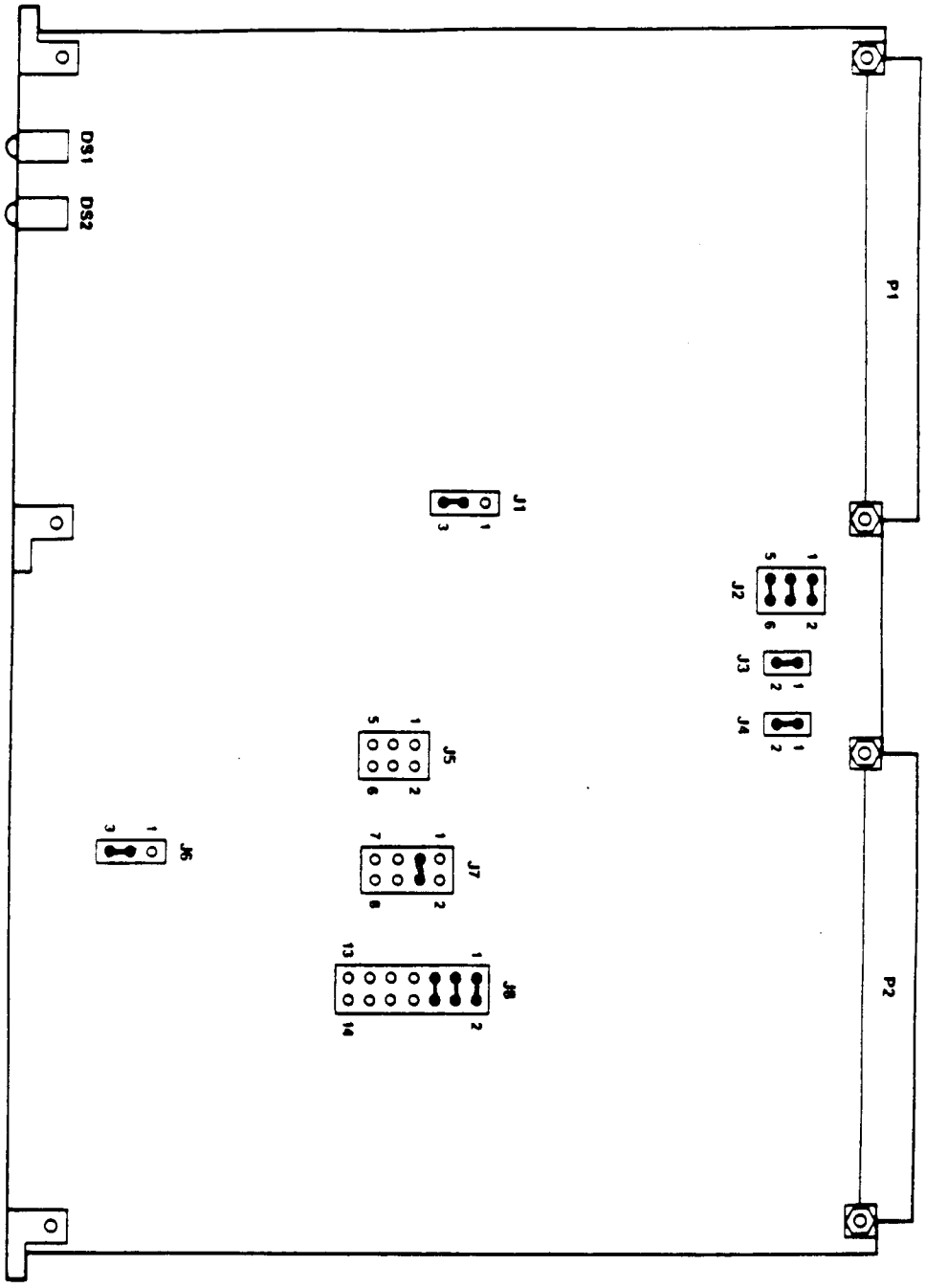


FIGURE 5-1. MVME316 Jumper Header Locations

TABLE 5-1. MVME316 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME316 VERSAdos JUMPER SETTINGS	
J1	VMEBUS TIME-OUT ENABLE SELECT	2-3
J2	+5VDC EXTENSION TO I/O CHANNEL SELECT	1-2, 3-4, 5-6
J3	+12VDC EXTENSION TO I/O CHANNEL SELECT	1-2
J4	-12VDC EXTENSION TO I/O CHANNEL SELECT	1-2
J5	I/O CHANNEL BASE ADDRESS SELECT	NO JUMPERS
J6	SOFTWARE RESET ENABLE SELECT	2-3
J7	BIM AND SOFTWARE RESET ADDRESS SELECT (MSB)	3-4
J8	BIM AND SOFTWARE RESET ADDRESS SELECT (LSB)	1-2, 3-4, 5-6
	MVME316 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

5

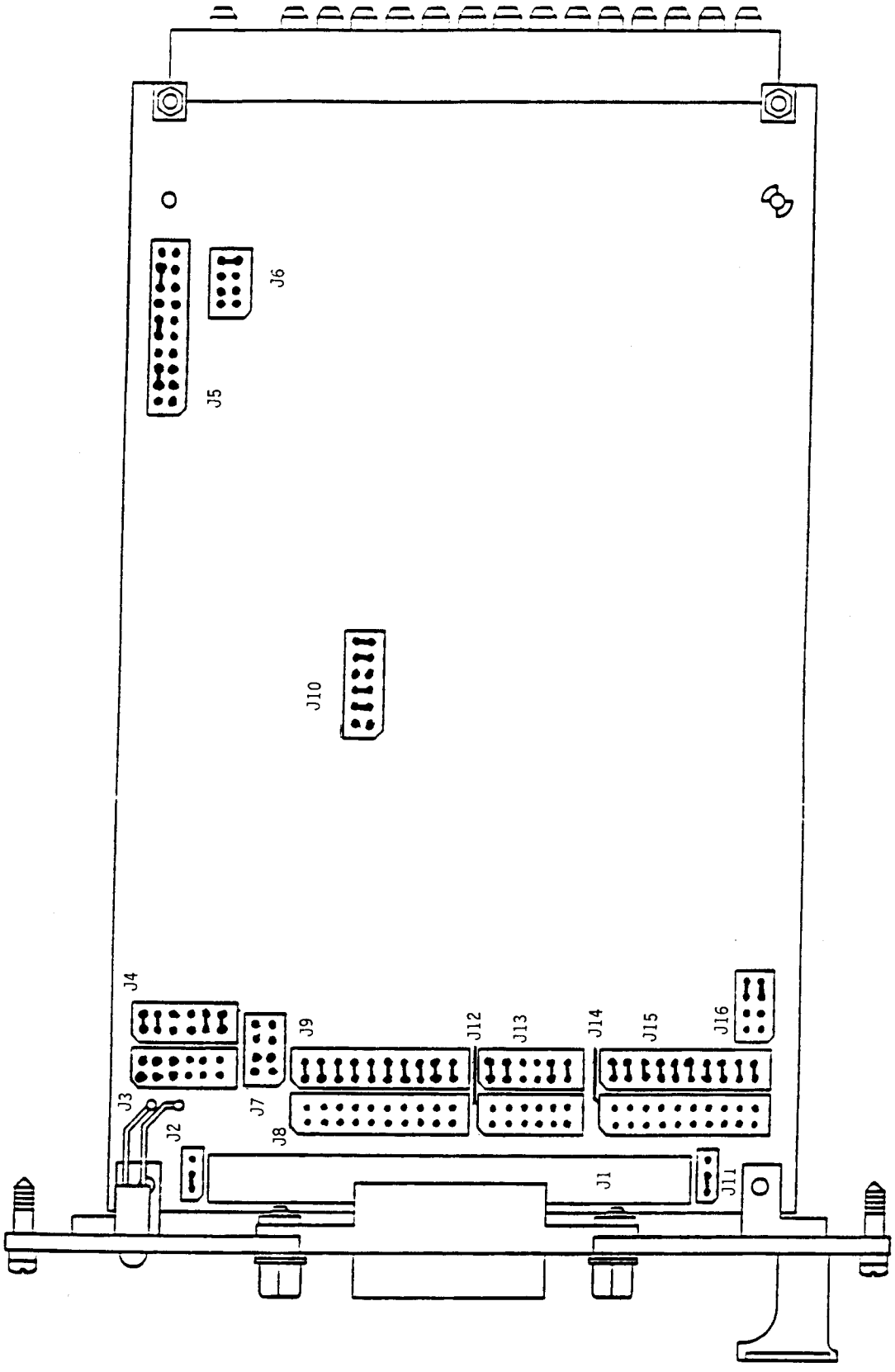


FIGURE 5-2. MVME400 Jumper Header Locations

TABLE 5-2. MVME400 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME400 VERSAdos JUMPER SETTINGS	
J2	PORT 2 TXC SELECT	1-2
J3	PORT 2 EXTERNAL CLOCK SELECT	NO JUMPERS
J4	PORT 2 INTERNAL CLOCK SELECT	1-2, 3-4, 9-10, 11-12
J5	INTERRUPT LEVEL SELECT	3-5, 9-11, 15-17
J6	BASE ADDRESS SELECT	7-8
J7	PORT 2 CTS FLOW CONTROL	5-7, 6-8
J8	PORT 2 TO MODEM SELECT	NO JUMPERS
J9	PORT 2 TO TERMINAL SELECT	1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16, 17-18, 19-20
J10	BAUD RATE PORT 1 & 2 SELECT	3-4, 5-6, 9-10, 11-12
J11	PORT 1 TXC SELECT	1-2
J12	PORT 1 EXTERNAL CLOCK SELECT	NO JUMPERS
J13	PORT 1 INTERNAL CLOCK SELECT	1-2, 3-4, 9-10, 11-12
J14	PORT 1 TO MODEM SELECT	NO JUMPERS
J15	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
J16	PORT 1 CTS FLOW CONTROL	5-7, 6-8

5

5.3 MVME410 DUAL PARALLEL PORT MODULE

The MVME410 Dual Parallel Port Module is an I/O Channel-compatible dual printer interface module. The module is used to allow an I/O Channel master to include two printer interfaces. These interfaces may consist of one printer interface and a general-purpose 16-bit parallel data port, or two general-purpose 16-bit parallel data I/O ports. The MVME410 conforms to the single-wide VMEmodule form factor and connects to the I/O Channel via a 64-pin DIN standard connector.

The MVME410 utilizes two MC6821 Peripheral Interface Adapter buffered ports. Each port is capable of driving two of four I/O Channel interrupt lines.

5.3.1 MVME410 Standard VERSAdos Configuration

Figure 5-3 depicts the jumper locations of the MVME410 module. Table 5-3 lists the proper jumper settings required to operate the module in a standard configuration using VERSAdos. Refer to the Dual Parallel Port Module User's Manual (MVME410) to compare these settings with the factory-set configuration.

5.4 M68RAD1 REMOTE INTELLIGENT ANALOG-TO-DIGITAL CONVERSION MODULE

The M68RAD1 Remote Intelligent Analog-to-Digital Conversion Module measures fast changing analog voltages with 12 bits of accuracy (unipolar operation). Four programmable full-scale voltage input ranges are available: 0-10 Vdc, 0-5 Vdc, 0-2.5 Vdc, and 0-1 Vdc. Analog current inputs may also be measured by installing appropriately sized voltage dropping (sensing) resistors. Conversion time is 33 us for a selected range. Short cycle operation is programmable for faster operation. An onboard precision reference power supply automatically checks the calibration of the module. An external trigger input is also provided to allow the user to measure waveforms at a particular point in time.

The M68RAD1 is controlled by an onboard MC68B09 microprocessor. The module provides a choice of remote parallel or serial operation and 32 single-ended/16 differential A/D channels. I/O Channel address and serial communication port identification is selectable for the application of multiple boards on the I/O Channel and/or a serial I/O network.

5.4.1 M68RAD1 Standard VERSAdos Configuration

Figure 5-4 illustrates the jumper header locations of the M68RAD1 module. Table 5-4 lists the jumper settings required to operate VERSAdos using a system employing the M68RAD1 module. Refer to the Remote Intelligent Analog-to-Digital Conversion Module User's Manual (M68RAD1) to compare these settings with the factory-set configuration.

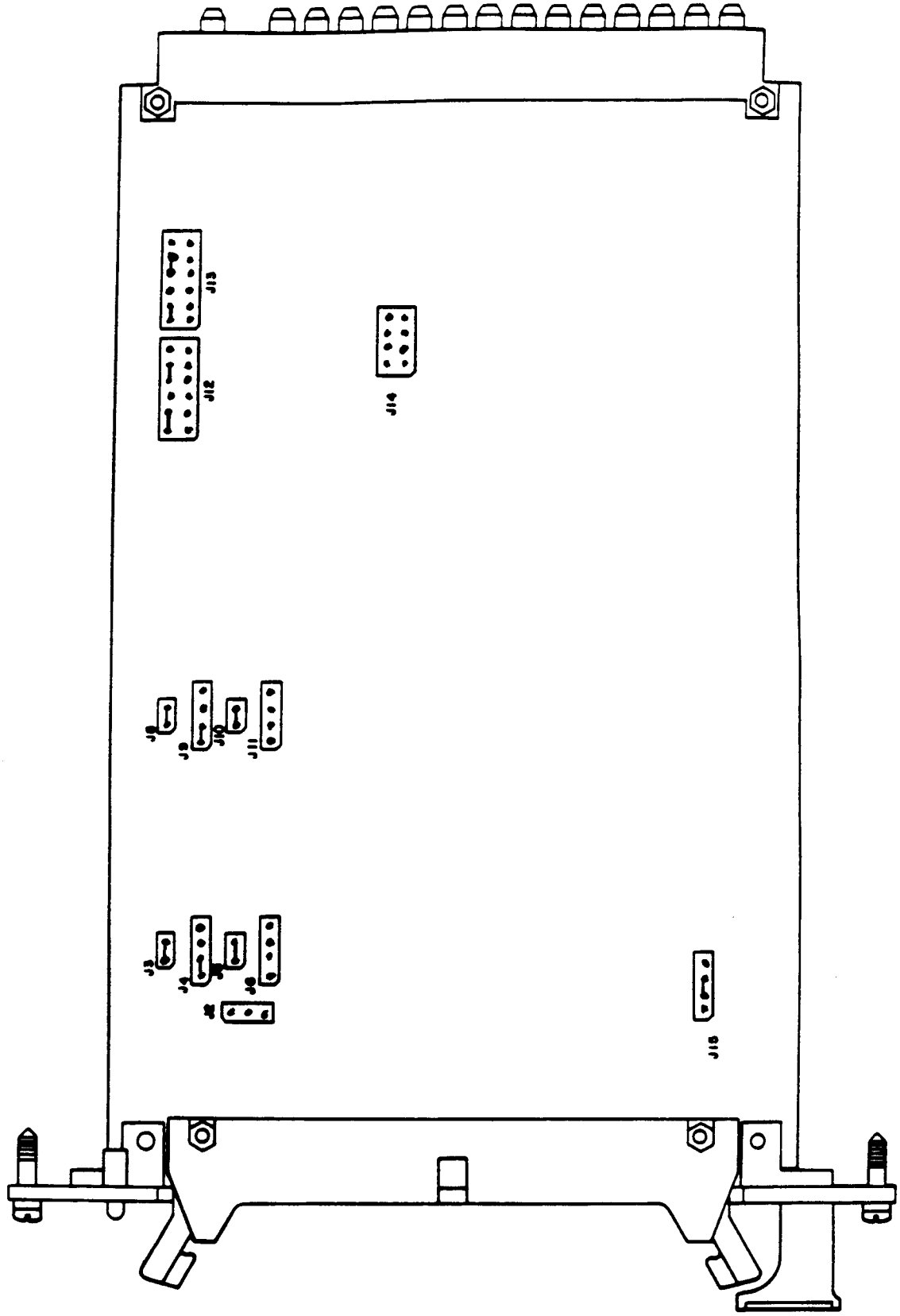


FIGURE 5-3. MVME410 Jumper Header Locations

TABLE 5-3. MVME410 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME410 VERSAdos JUMPER SETTINGS	
J2	LED MONITOR	2-3
J3	PORT 1 (P1CA2 P1PA0-P1PA7) DIRECTION	1-2
J4		1-2
J5	PORT 1 (P1CB2 P1PB0-P1PB7) DIRECTION	1-2
J6		NO JUMPERS
J8	PORT 2 (P2CA2 P2PA0-P2PA7) DIRECTION	1-2
J9		1-2
J10	PORT 2 (P2CB2 P2PB0-P2PB7) DIRECTION	1-2
J11		NO JUMPERS
J12	INTERRUPT SELECT (PORT 1)	2-4, 8-10
J13	INTERRUPT SELECT (PORT 2)	2-4,8-10
J14	BASE ADDRESS SELECT	NO JUMPERS
J15	LED CONTROL SELECT	2-3

5

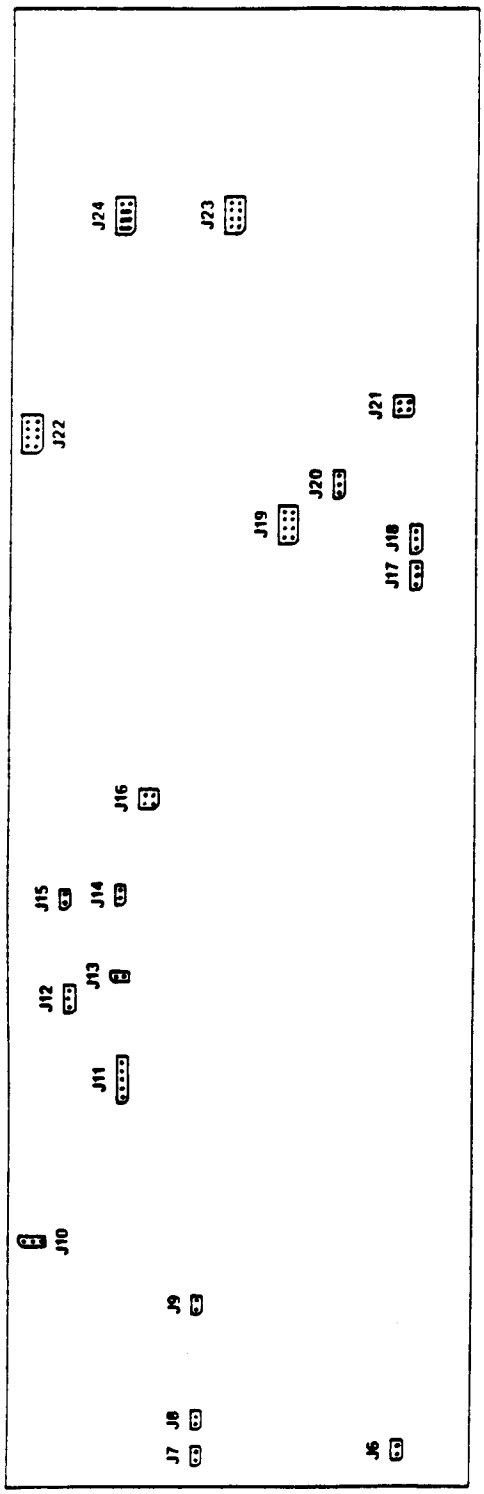


FIGURE 5-4. M68RAD1 Jumper Header Locations

TABLE 5-4. M68RAD1 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	RAD VERSAdos JUMPER SETTINGS	
J6 J8 J9 J7 J15	SINGLE ENDED/DIFFERENTIAL MEASUREMENT SELECT	
J10 J11 J12 J13	OPERATION POLARITY SELECT	
J14	16/32 CHANNEL SELECT	
J24	I/O CHANNEL BASE ADDRESS SELECT	1-2, 3-4, 5-6
J22	I/O CHANNEL INTERRUPT LEVEL SELECT	
J23	SERIAL ADDRESS IDENTIFIER OPTIONS	
J17 J18 J20 J21	SERIAL INTERFACE DRIVER OPTIONS	
J19	SERIAL BAUD RATE SELECT	
J16	MPU INTERRUPT - E.O.C. SELECT	

5

5.5 M68RIO1 REMOTE INPUT/OUTPUT MODULE

The M68RIO1 Remote Input/Output Module provides for remote parallel operation at distances up to 12 feet from an I/O Channel Master. This module accepts any mix of up to 16 user-supplied Crydom Series 6, Opto-22, or other compatible solid state relay input and output modules. The M68RIO1 enables an I/O Channel interrupt to be generated simultaneously by up to eight solid state input modules. Block select jumpering allows the M68RIO1 to be configured for operation with any of 15 other I/O Channel-compatible modules.

5.5.1 M68RIO1 Standard VERSAdos Configuration

Figure 5-5 illustrates the jumper header locations of the M68RIO1 module. Table 5-5 lists the jumper settings required to operate VERSAdos on a system employing the M68RIO1. Refer to the Remote Input/Output Module User's Manual (M68RIO1) to compare these settings with the factory-set configuration.

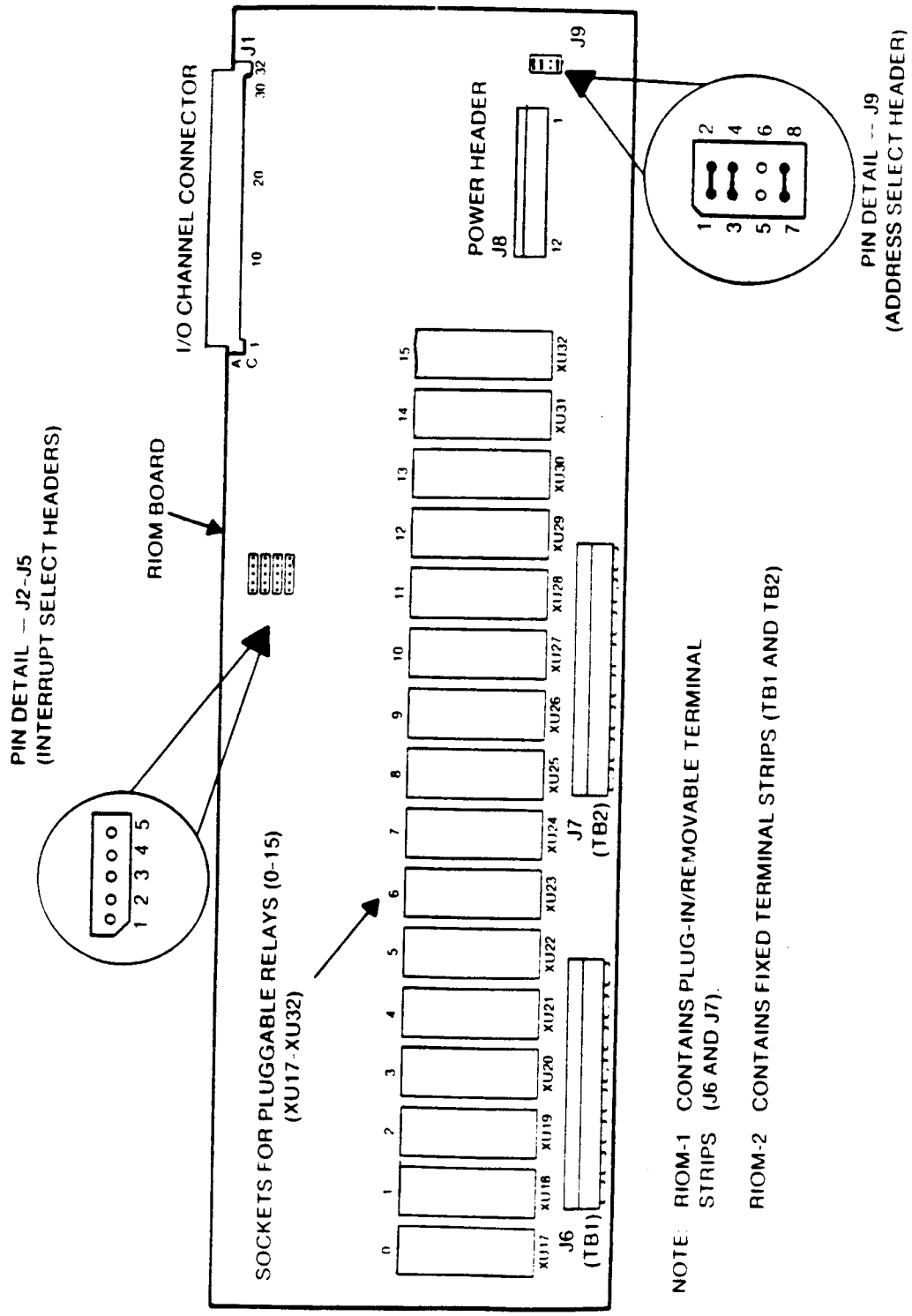


FIGURE 5-5. M68RIO1 Jumper Header Locations

TABLE 5-5. M68RIO1 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	RIO VERSAdos JUMPER SETTINGS	
J2	I/O CHANNEL INTERRUPT REQUEST LEVEL SELECT (IRQ1A)	1-2
J3	I/O CHANNEL INTERRUPT REQUEST LEVEL SELECT (IRQ1B)	1-2
J4	I/O CHANNEL INTERRUPT REQUEST LEVEL SELECT (IRQ2A)	1-2
J5	I/O CHANNEL INTERRUPT REQUEST LEVEL SELECT (IRQ2B)	1-2
J9	BASE ADDRESS SELECT	1-2, 3-4, 7-8

5

5.6 I/O CHANNEL ADDRESS SPACE

Tables 5-6 through 5-11 consists of the I/O Channel address maps for several modules which utilize the I/O Channel including:

M68RAD1 Remote Intelligent A/D Conversion Module
M68RIO1 Remote Input/Output Module
M68RWIN1 Winchester Disk Controller
MVME400 Dual RS-232C Serial Port Module
MVME410 Dual Parallel Port Module
MVME420 SASI Adapter Module
MVME435 Buffered 9-Track Magnetic Tape Adapter Module
MVME600 Analog Input Module
MVME605 Analog Output Module
MVME610 AC Input Module
MVME615 AC Output Module
MVME616 AC Output Module
MVME620 DC Input Module
MVME625 DC Output Module

These tables describe the configuration of up to four of each module without conflicts in addresses. The SYSGEN file CSYSTEMS.SYSTEM.CI contains the parameter that defines the I/O Channel base address. IOCBASE=$FE6000, for an MVME110-1 system. Appendix C contains a list of all I/O Channel addresses.

TABLE 5-6. Top Level I/O Channel Address Map

I/O ADDRESS	DESCRIPTION OF USAGE
000 - 0FF	Refer to Table 5-7
100 - 1FF	Refer to Table 5-8
200 - 2FF	Reserved
300 - 3FF	Reserved
400 - 4FF	Reserved (RAD #4)
500 - 5FF	Reserved (RAD #3)
600 - 6FF	RAD #2
700 - 7FF	RAD #1
800 - 8FF	Reserved (VME605 #4)
900 - 9FF	Reserved (VME605 #3)
A00 - AFF	VME605 #2
B00 - BFF	VME605 #1
C00 - CFF	Reserved (VME600 #4)
D00 - DFF	Reserved (VME600 #3)
E00 - EFF	VME600 #2 (with up to 5 VME601's)
F00 - FFF	VME600 #1 (with up to 5 VME601's)

5

TABLE 5-7. Map of I/O Channel Addresses 000 - 0FF

I/O ADDRESS	DESCRIPTION OF USAGE
000 - 00F	Refer to Table 5-11
010 - 01F	Refer to Table 5-10
020 - 02F	Refer to Table 5-9
030 - 03F	Refer to Table 5-9
040 - 04F	Refer to Table 5-9
050 - 05F	Refer to Table 5-9
060 - 06F	Refer to Table 5-9
070 - 07F	Refer to Table 5-9
080 - 08F	Reserved (VME410 #2)
090 - 09F	Reserved (VME410 #3)
0A0 - 0AF	Reserved (VME410 #4)
0B0 - 0BF	Reserved (VME400 #4)
0C0 - 0CF	Reserved (VME400 #3)
0D0 - 0DF	VME400 #2
0E0 - 0EF	VME400 #1
0F0 - 0FF	VME410 #1

TABLE 5-8. Map of I/O Channel Addresses 100 - 1FF

I/O ADDRESS	DESCRIPTION OF USAGE
100 - 11F	Reserved
120 - 13F	Reserved
140 - 15F	Reserved
160 - 17F	Reserved
180 - 19F	Reserved (VME435 #4)
1A0 - 1BF	Reserved (VME435 #3)
1C0 - 1DF	VME435 #2
1E0 - 1FF	VME435 #1

TABLE 5-9. Map of I/O Channel Addresses 000 - 07F

I/O ADDRESS	DESCRIPTION OF USAGE
000 - 007	Refer to Table 5-11
008 - 00F	Refer to Table 5-11
010 - 017	Refer to Table 5-10
018 - 01F	Refer to Table 5-10
020 - 027	Reserved
028 - 02F	Reserved
030 - 037	Reserved
038 - 03F	Reserved
040 - 047	Reserved (RIO #4)
048 - 04F	Reserved (RIO #3)
050 - 057	RIO #2
058 - 05F	RIO #1
060 - 067	VME420 #2
068 - 06F	RWIN #1
070 - 077	RWIN #2
078 - 07F	VME420 #1

TABLE 5-10. Map of I/O Channel Addresses 000 - 01F

I/O ADDRESS	DESCRIPTION OF USAGE
000 - 001	Refer to Table 5-11
002 - 003	Refer to Table 5-11
004 - 005	Refer to Table 5-11
006 - 007	Refer to Table 5-11
008 - 009	Refer to Table 5-11
00A - 00B	Refer to Table 5-11
00C - 00D	Refer to Table 5-11
00E - 00F	Refer to Table 5-11
010 - 011	Reserved (VME620 #4)
012 - 013	Reserved (VME620 #3)
014 - 015	(VME620 #2)
016 - 017	(VME620 #1)
018 - 019	Reserved (VME610 #4)
01A - 01B	Reserved (VME610 #3)
01C - 01D	VME610 #2
01E - 01F	VME610 #1

5

TABLE 5-11. Map of I/O Channel Addresses 000 - 00F

I/O ADDRESS	DESCRIPTION OF USAGE
000	Reserved
001	Reserved
002	Reserved
003	Reserved
004	Reserved (VME625 #4)
005	Reserved (VME625 #3)
006	VME625 #2
007	VME625 #1
008	Reserved (VME616 #4)
009	Reserved (VME616 #3)
00A	VME616 #2
00B	VME616 #1
00C	Reserved (VME615 #4)
00D	Reserved (VME615 #3)
00E	VME615 #2
00F	VME615 #1

5

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 6

MISCELLANEOUS VMEmodules

6.1 MVME300 GPIB CONTROLLER WITH DMA

The MVME300 General Purpose Interface Bus (GPIB) Controller with Direct Memory Access (DMA) provides the user with a complete IEEE-488 bus listener/talker/controller interface for use with a VMEbus system. To a host on the VMEbus, the MVME300 appears as 32 adjacent 16-bit read/write registers with which data transactions are performed using the odd bytes only. Several MVME300 modules may be used as GPIB controllers in a system, each with its own IEEE-488 bus but all controlled by the VMEbus system controller.

6.1.1 MVME300 Standard VERSAdos Configuration

Figure 6-1 illustrates the jumper header locations of the MVME300 module. Table 6-1 lists the jumper settings required to operate VERSAdos on a system employing the MVME300. Refer to the GPIB Controller with DMA User's Manual (MVME300) to compare these settings with the factory-set configuration.

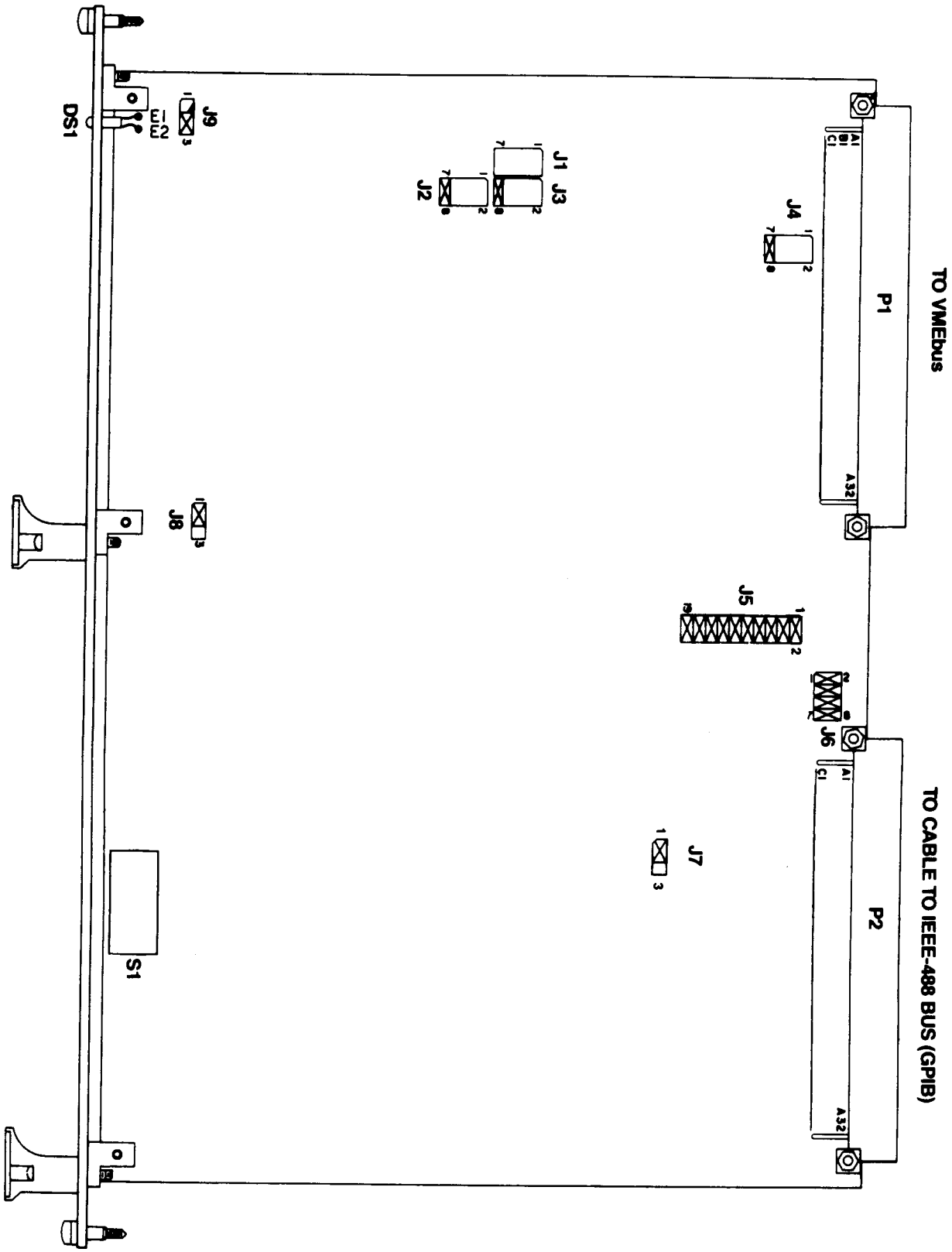


FIGURE 6-1. MVME300 Jumper Header Locations

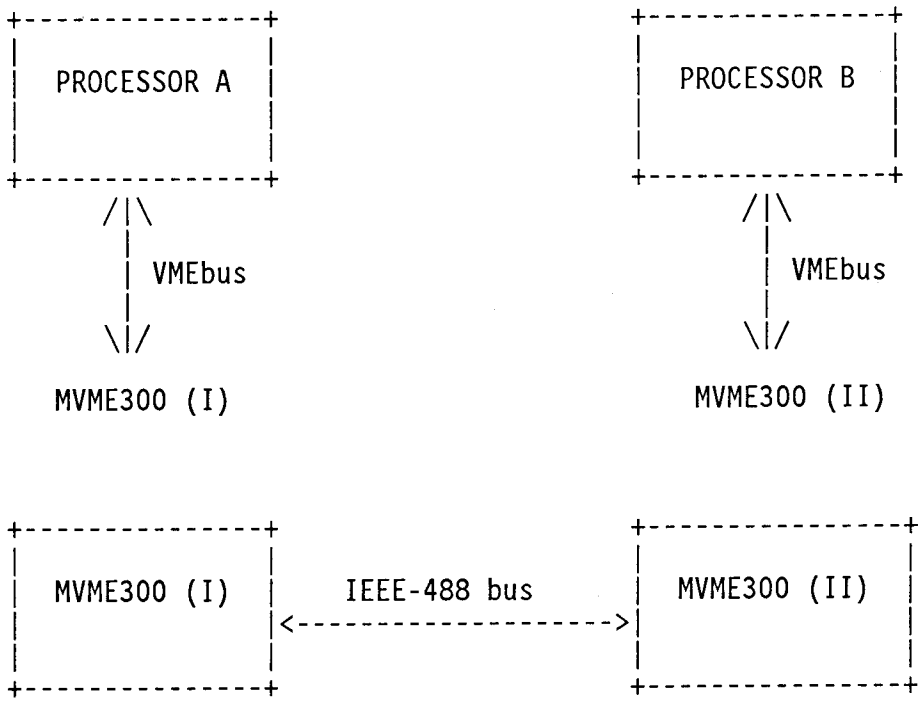
TABLE 6-1. MVME300 VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME300 VERSAdos JUMPER SETTINGS	
J1	BGXIN* TO BGXOUT* BYPASS SELECT	NO JUMPERS
J2	BGXOUT* ENABLE SELECT	7-8
J3	BGXIN* ENABLE SELECT	7-8
J4	BRX* OUTPUT ENABLE SELECT	7-8
J5	BASE ADDRESS SELECT	1-2, 3-4, 5-6, 7-8, 11-12, 13-14, 15-16, 17-18, 19-20
J6	VMEBUS RELEASE COUNT REGISTER COUNT	1-2, 3-4, 5-6, 7-8
J7	TRI-STATE OR O.C. BUFFER SELECT	1-2
J8	FIFO RAM CONTROLLER MEMORY SIZE SELECT	1-2
J9	BCLR* ENABLE	2-3
	MVME300 VERSAdos JUMPER SETTINGS	
SWITCH NUMBER		
S1-1	SC	OPEN
S1-2	DAL	OPEN
S1-3	DAT	OPEN
S1-4	A4	OPEN
S1-5	A3	OPEN
S1-6	A2	OPEN
S1-7	A1	OPEN
S1-8	A0	OPEN
	MVME300 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

6

6.1.2 MVME300 Communication Between Processors (MVME110-1 or VME/10)

The following is an example of using the MVME300 module to communicate between microprocessor-based systems (MVME110-1 or VME/10):



Configuration Information

	MVME300 (I)	MVME300 (II)
VERSAdos Processor A	"BUSA" Bus is not shareable ATW $1F (system controller) Primary address: $01 Board address: $FF0400	"BA0A" Device is not shareable ATW $03 (talker/listener) Primary address: $10 Board address: $FF0400
VERSAdos Processor B	N/A	"BUSA" Bus is not shareable ATW $03 (talker/listener) Primary address: $10 Board address: $FF0400

Jumper Configurations

J1 -- BGXIN* TO BGXOUT* Bypass Header

J2, J3, J4 -- Bus Grant and Bus Request Headers

	MVME300 (I)	MVME300 (II)
J1	none	1-2, 3-4, 5-6
J2	7-8	7-8
J3	7-8	7-8
J4	7-8	7-8

J5 -- VMEbus Board Base Address Header

J5	all but 9, 10	all but 9, 10

J6 -- VMEbus Release Count Register (VBRCR) Count Header

J6	1-2, 3-4, 5-6, 7-8	1-2, 3-4, 5-6, 7-8

J7 -- Three-State or Open-Collector Buffer Selection Header

J7	1-2	1-2

J8 -- FIFO RAM Controller Memory Size Header

J8	1-2	1-2

J9 -- Bus Clear BCLR* Enable Header

J9	2-3	2-3

DIP Switch

	MVME300 (I)	MVME300 (II)
sc	closed	open
dat	closed	closed
dal	closed	closed
A4	open	closed
A3	open	open
A2	open	open
A1	open	open
A0	closed	open

NOTE: When configuring an MVME300 module that serves as a bus, the MVME300 driver reads the DIP switch to obtain the requested primary address. After the MVME300 driver reads the DIP switch, it takes the 1's complement of the number read, and writes that 1's complement into the TMS9914A address register. Thus, A4 closed and A3-A0 open as above will result in a primary address of $10.

Also works with:

J7	2-3
J8	2-3
J9	1-2

Processor A: WriteBA0A
Processor B: ReadBUSA

NOTE: When installing the MVME300 module in a VMEbus system, it is important to remember that whenever there are empty slots between modules in the VMEmodule chassis, jumpers must be installed on the VMEmodule backplane at the empty slot locations to jumper the bus arbitration signals and acknowledge signals across the empty slot(s). Refer to the appropriate hardware manual for further information.

If this procedure is not followed, VERSAdos will not boot if the system where the MVME300 module has system controller capabilities.

6.2 MVME435A BUFFERED 9-TRACK MAGNETIC TAPE ADAPTER

The MVME435A Buffered 9-Track Magnetic Tape Adapter is designed to add magnetic tape data storage capability to microcomputer system resources on the I/O Channel. The module provides the interface for one or two dual-density, 1/2-inch magnetic tape formatters allowing the addition of up to eight compatible tape drivers operating at speeds from 12.5 to 125 inches/second in the start/stop mode.

The MVME435A provides software-selectable direct or buffered data transfer modes. In the buffered mode, direct data transfers of 4Kb per operation may occur between drives.

6.2.1 MVME435A Standard VERSAdos Configuration

Figure 6-2 illustrates the jumper header locations of the MVME435A module. Table 6-1 lists the jumper settings required to operate VERSAdos on a system employing the MVME435A. Refer to the Buffered 9-Track Magnetic Tape Adapter User's Manual (MVME435A) to compare these settings with the factory-set configuration.

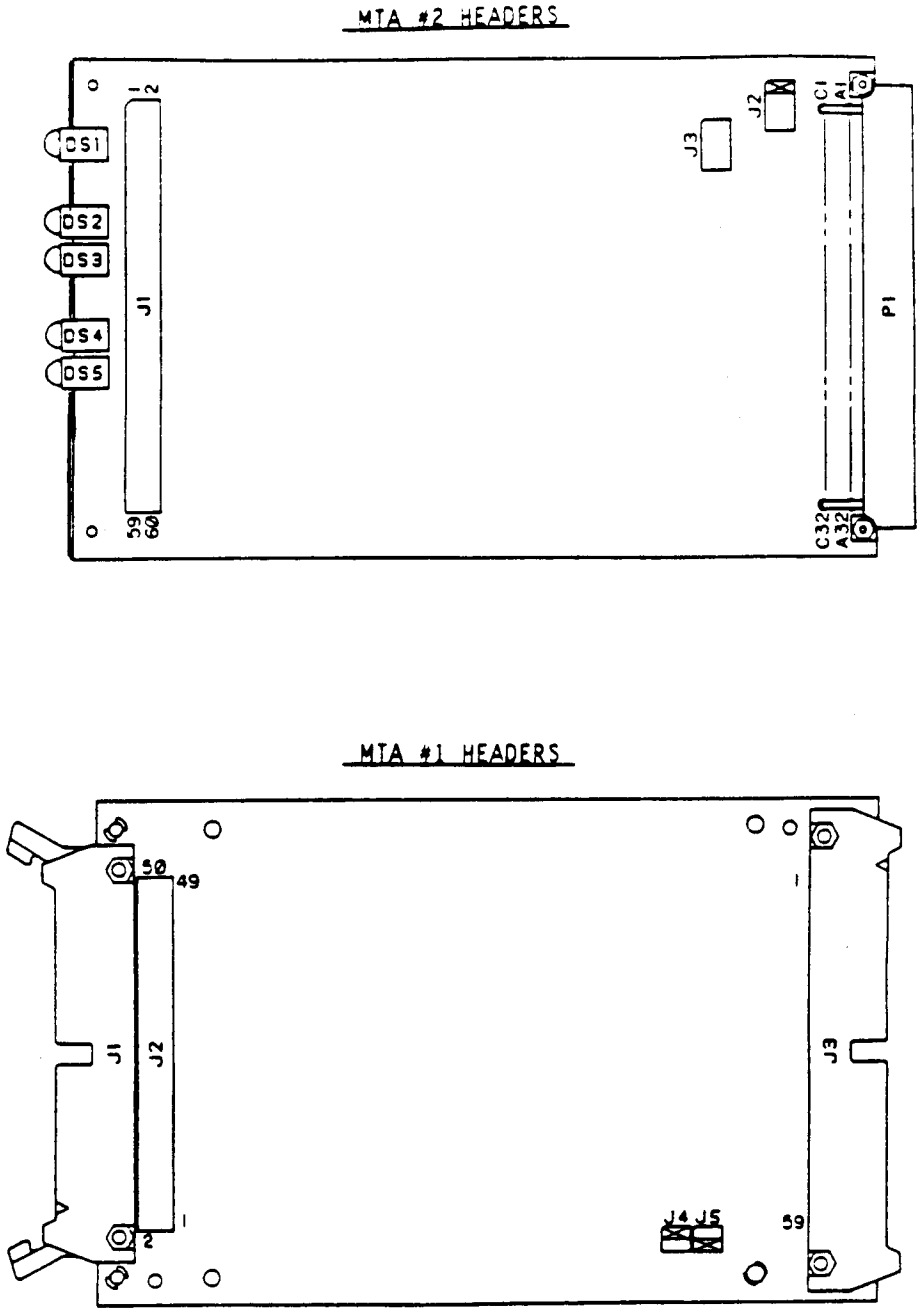


FIGURE 6-2. MVME435A Jumper Header Locations

TABLE 6-2. MVME435A VERSAdos-Supported Jumper Settings

JUMPER NUMBER	DESCRIPTION	SETTING
	MVME435A VERSAdos JUMPER SETTINGS	
J2	I/O CHANNEL INTERRUPT LEVEL SELECT	1-2
J3	BASE ADDRESS SELECT	NO JUMPERS
J4	DNBSY INTERRUPT SELECT	1-2
J5	INTERRUPT ENABLE SELECT	3-4

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 7

VME CHASSIS/CARD CAGES AND BACKPLANES

7.1 INTRODUCTION

As a reference, the features of Motorola's chassis/card cages and backplanes for the VMEsystem are listed in this chapter. Figures 7-1 and 7-2 depict typical chassis and backplanes used in a VMEsystem.

7.2 VME CHASSIS/CARD CAGES

7.2.1 MVME940-1 Chassis/Card Cage

The MVME940-1 Chassis/Card Cage is designed to help convert individual VMEmodule and I/Omodule components into a complete, compact, self-contained microcomputer system. Features of the MVME940-1 assembly include:

- 19-inch rack-mounting card rack
- Card slots and hardware for up to seven (double), and up to ten (single) modules.
- One MVME921 9-slot VMEbus backplane (motherboard)
- Two MVME922 5-slot I/O Channel backplanes (motherboards)
- MVME910-3 plug-in power supply
- DC wiring harness
- I/O Channel signal cables

7.2.2 MVME941 Chassis/Card Cage

The MVME941 is a fully-assembled rack-mountable card cage chassis designed to support a variety of VMEmodules and I/Omodules. This chassis is supplied with all necessary hardware, connectors, and backplanes. The user must provide a power supply and cables. Features of the MVME941 include:

- 19-inch rack-mountable card cage
- Card guides and hardware for up to nine double, and ten single modules
- One MVME921 9-slot VMEbus backplane (motherboard)
- Two MVME922 5-slot I/O Channel backplanes (motherboards)
- DC power cable assembly
- I/O Channel cable assembly

7.2.3 MVME942 Chassis/Card Cage

The MVME942 is a fully-assembled rack-mountable card cage chassis which supports up to 20 VMEmodules. Features of the MVME942 include:

- 19-inch rack-mountable card cage
- Card guides and hardware for up to 20 double VMEmodules
- One MVME920 20-slot VMEbus backplane (motherboard)

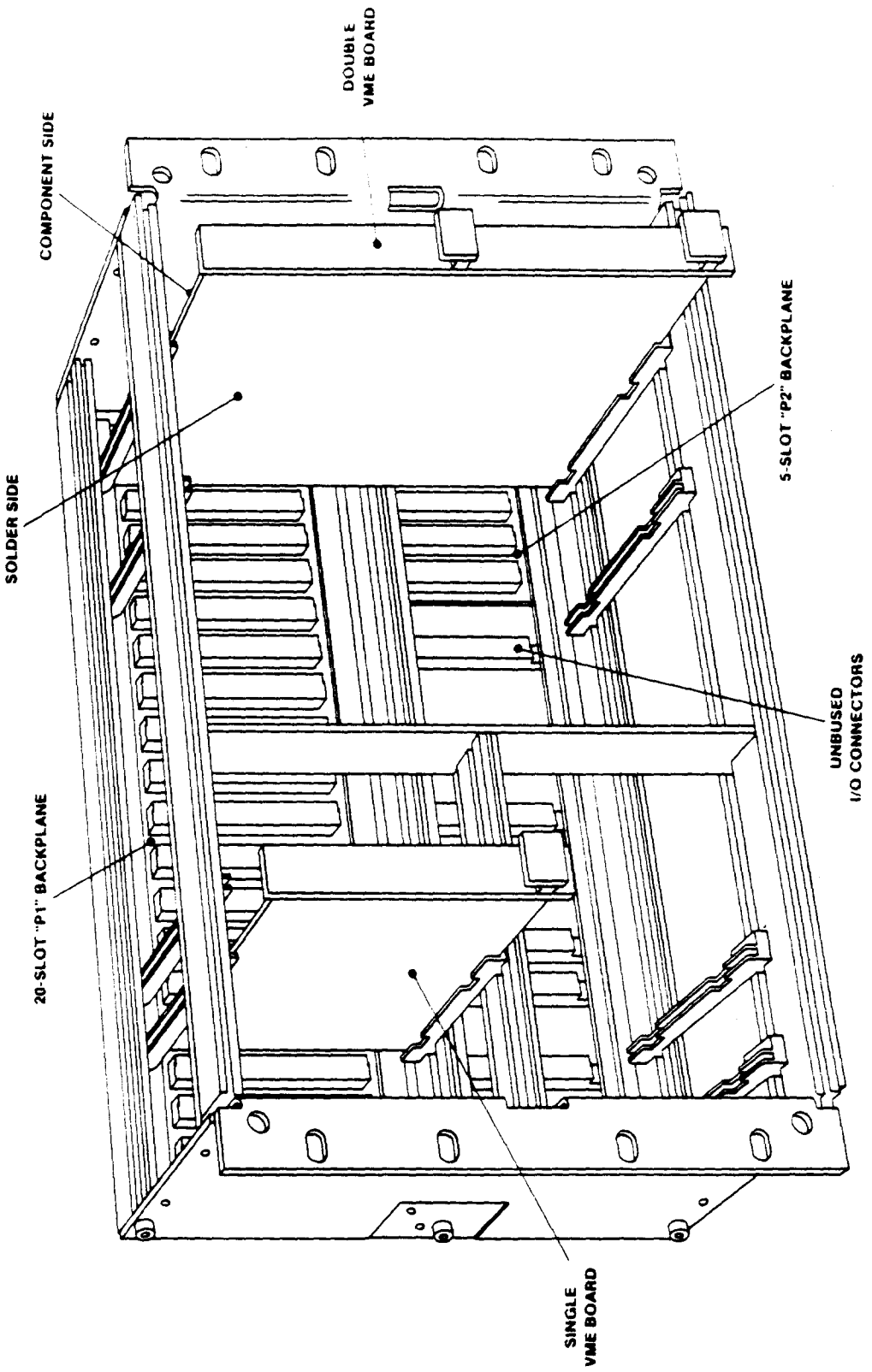


FIGURE 7-1. VME Chassis, Typical Configuration

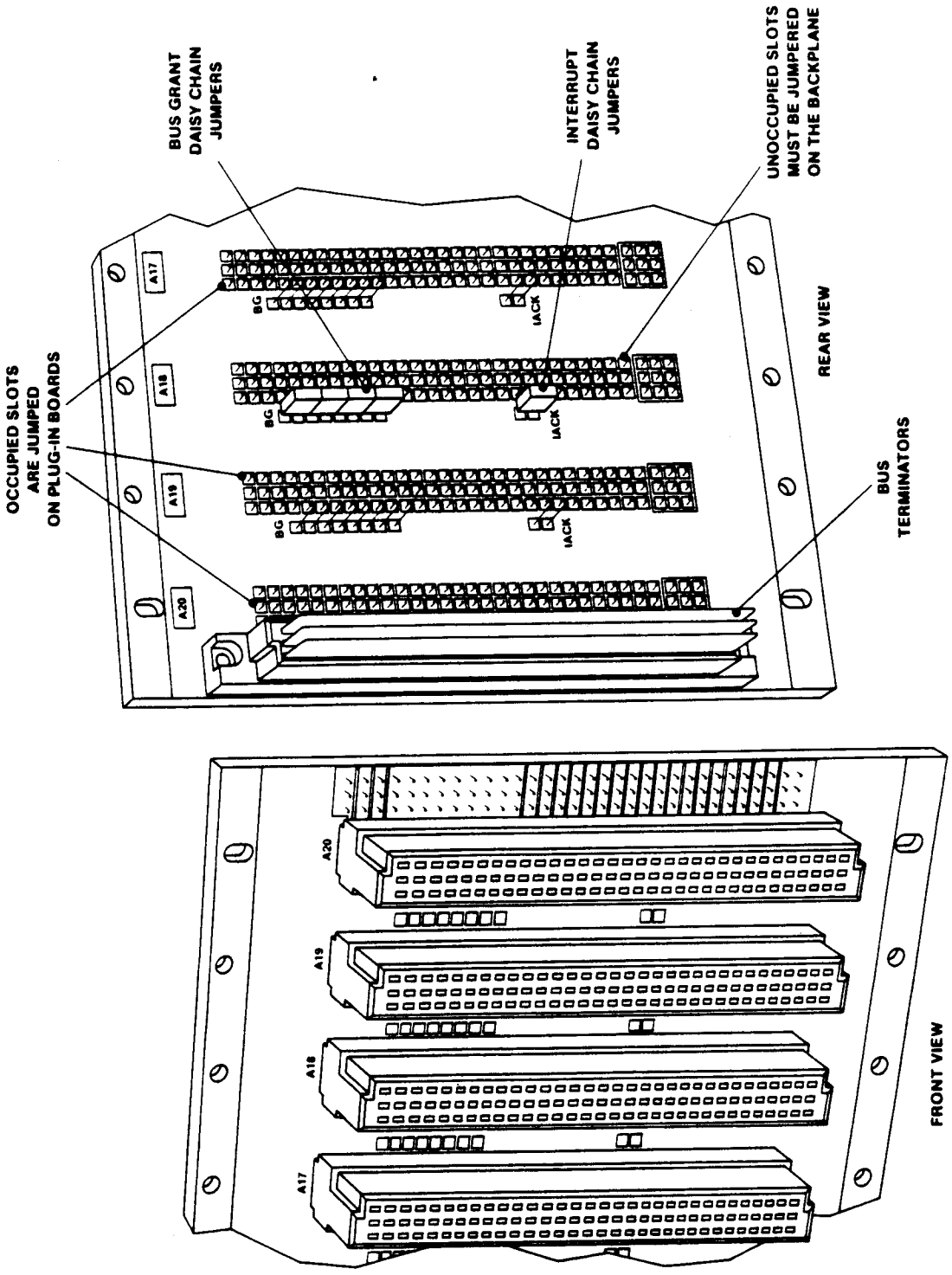


FIGURE 7-2. VME Backplane

7.2.4 MVME943 Chassis

The MVME943 is designed to accommodate a variety of VME components in an efficient housing which can be tailored to a specific user's needs. The MVME943 assembly features:

- 19-inch rack-mountable chassis
- 400-watt power supply
- AC filter/circuit breaker module
- Power switch module, with DC voltage indicators (LEDs)
- AC wiring harness
- DC power terminal strip
- DC wiring harness
- MVME924 3-slot I/O Channel backplanes (2)
- MVME921 9-slot VMEbus backplane
- VMEbus backplane terminating modules (2)
- I/O Channel signal cable
- Provision for mass storage unit (user-provided)

7.2.5 MVME944 Chassis

The MVME944 is designed to accommodate a variety of options geared to meet the specific needs of a wide range of users. The MVME944 assembly features:

- 19-inch rack-mountable chassis
- 400-watt power supply
- AC filter/circuit breaker module
- Power switch module, with DC voltage indicators (LEDs)
- AC wiring harness
- DC wiring harness
- DC power terminal strip
- MVME920 20-slot VMEbus backplane, with terminator modules

7

7.3 VME BACKPLANES

7.3.1 MVME920 20-Slot VMEbus Backplane

The MVME920 20-slot VMEbus Backplane is designed for use with MVME944 chassis. Features of the MVME920 include:

- Suitable for operation at up to 30 MHz
- Provides all signal line and power line connections
- Faston power, sense, and ground connections
- Jumper areas for daisy-chain lines
- Plug-in bus termination, with active negative spike suppression
- DIN 41612C 96-pin sockets for VMEmodules

7.3.2 MVME921 9-Slot VMEbus Backplane

The MVME921 9-slot VMEbus Backplane is intended for use with a variety of VMEmodules and related components. Features of the MVME921 include:

- Suitable operation at up to 20 MHz
- Provides all signal line and power line connections
- Faston power and ground connections
- Jumper areas for daisy-chain lines
- Bus termination networks on both ends
- Test connector for bus signal access
- DIN 41612C 96-pin sockets

7.3.3 MVME922 5-Slot I/O Channel Backplane

The MVME922 is designed to house I/Omodules in a VME chassis. This component includes:

- All signal and power connections
- Faston power and ground connectors
- DIN 41612C sockets
- Termination for all signal lines -- except INT 1-4* and XACK*
- Test connector for signal access

7.3.4 MVME924 3-Slot I/O Channel Backplane

The MVME924 3-Slot I/O Channel Backplane is intended to house I/Omodules in a VME chassis configuration. Features of the MVME924 include:

- All signal line and power line connections
- Faston power and ground connections
- Bus termination, using 330-ohm/470-ohm SIP resistor packs
- DIN 41612C 96-pin sockets for I/Omodules

THIS PAGE INTENTIONALLY LEFT BLANK.

CHAPTER 8

STANDARD SYSTEM CONFIGURATIONS

8.1 INTRODUCTION

This chapter describes the standard VERSAdos configuration for five different VMEsystems. Each system is shown via a block diagram of the components supported in a standard VERSAdos system.

8.2 MVME101 SYSTEM

Table 8-1 lists the SYSGEN map for the standard VERSAdos-supported system based on the MVME101 Monoboard Microcomputer. Figure 8-1 illustrates the standard MVME101 system.

TABLE 8-1. MVME101 Standard System SYSGEN Map

```
================================================================================
FILENAME          TASK  PROC  SEG    ADDR      TCB
================================================================================
RMS.LO                  RMS   RMS0   $010000
                              RMS2   $010100
DRVLIB.LO               DRVL  DRVL   $014E00
TERMLIB.LO              TERM  TERM   $015000
M315DRV.LO              M315  M315   $016300
EPCIDRV.LO              EPCI  EPCI   $017000
PIADRV.LO               PIAD  PIAD   $017400
FHS.LO            .FHS        .FHS   $017A00   $018E00
IOS.LO            .IOS        .IOS   $019000   $01A900
FMS.LO            .FMS        .FMS   $01AB00   $01FD00
EET.LO            &EET        .EET   $01FF00   $024200
                              .STT   $020200
                              &EET   $020500
LDR.LO            &LDR        &LDR   $024400   $025900
IOI.LO            .IOI        IOSG   $025B00   $026C00
                              .IOI   $026700
SYSINIT.LO              SYSI  .INT   $026E00

- FINAL PC VALUE = $027800
- START-UP ADDRESS = $026E00
TOTAL NUMBER OF USER DEFINED SYMBOLS = 210

0 ERRORS ENCOUNTERED
================================================================================
```

To create the above output, execute the VME101.COPYSGEN.CF file to a blank user number followed by STD.SYSGEN.CF.

```
= USE SYS=9100.VME101
= SYS:9998.VME101  COPYSGEN.CF  SYS,SYS,9110
= STD.SYSGEN.CF
```

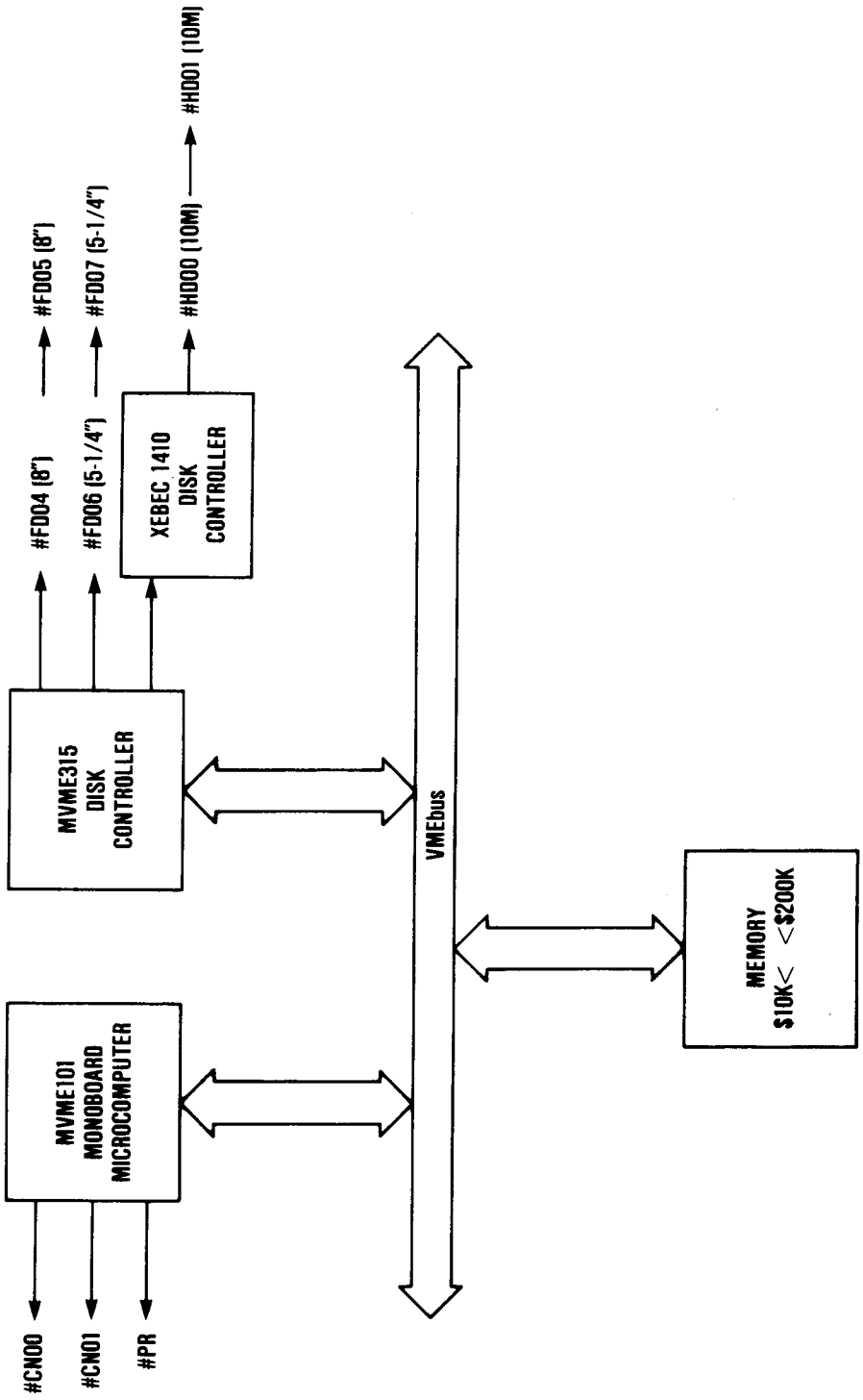


FIGURE 8-1. Standard MVME101 System

8.3 MVME110-1 SYSTEM

Table 8-2 lists the SYSGEN map for the standard VERSAdos-supported system based on the MVME110-1 Monoboard Microcomputer. Figure 8-2 illustrates the standard MVME110-1 system.

TABLE 8-2. MVME110-1 Standard SYSGEN Map

```
================================================================================
FILENAME        TASK    PROC  SEG    ADDR        TCB
================================================================================

RMS.LO                  RMS   RMS0   $040000
                              RMS2   $040100
DRVLIB.LO               DRVL  DRVL   $044E00
TERMLIB.LO              TERM  TERM   $045000
RWINDRV.LO              RWIN  RWIN   $046300
M420DRV.LO              M420  M420   $046E00
M315DRV.LO              M315  M315   $047700
M320DRV.LO              M320  M320   $048400
ACIADRV.LO              ACIA  ACIA   $049100
MPSCSPR.LO              MPSC  MPSC   $049400
MPSCDRV.LO              MPSC  MPSC   $049600
PIADRV.LO               PIAD  PIAD   $049D00
FHS.LO          .FHS          .FHS   $04A300     $04B700
IOS.LO          .IOS          .IOS   $04B900     $04D300
FMS.LO          .FMS          .FMS   $04D500     $052700
EET.LO          &EET          .EET   $052900     $056D00
                              .STT   $052C00
                              &EET   $052F00
LDR.LO          &LDR          &LDR   $056F00     $058400
IOI.LO          .IOI          IOSG   $058600     $05A300
                              .IOI   $059D00
SYSINIT.LO              SYSI  .INT   $05A500

- FINAL PC VALUE = $05AF00
- START-UP ADDRESS = $05A500
TOTAL NUMBER OF USER DEFINED SYMBOLS =   290

   0  ERRORS ENCOUNTERED
================================================================================
```

8

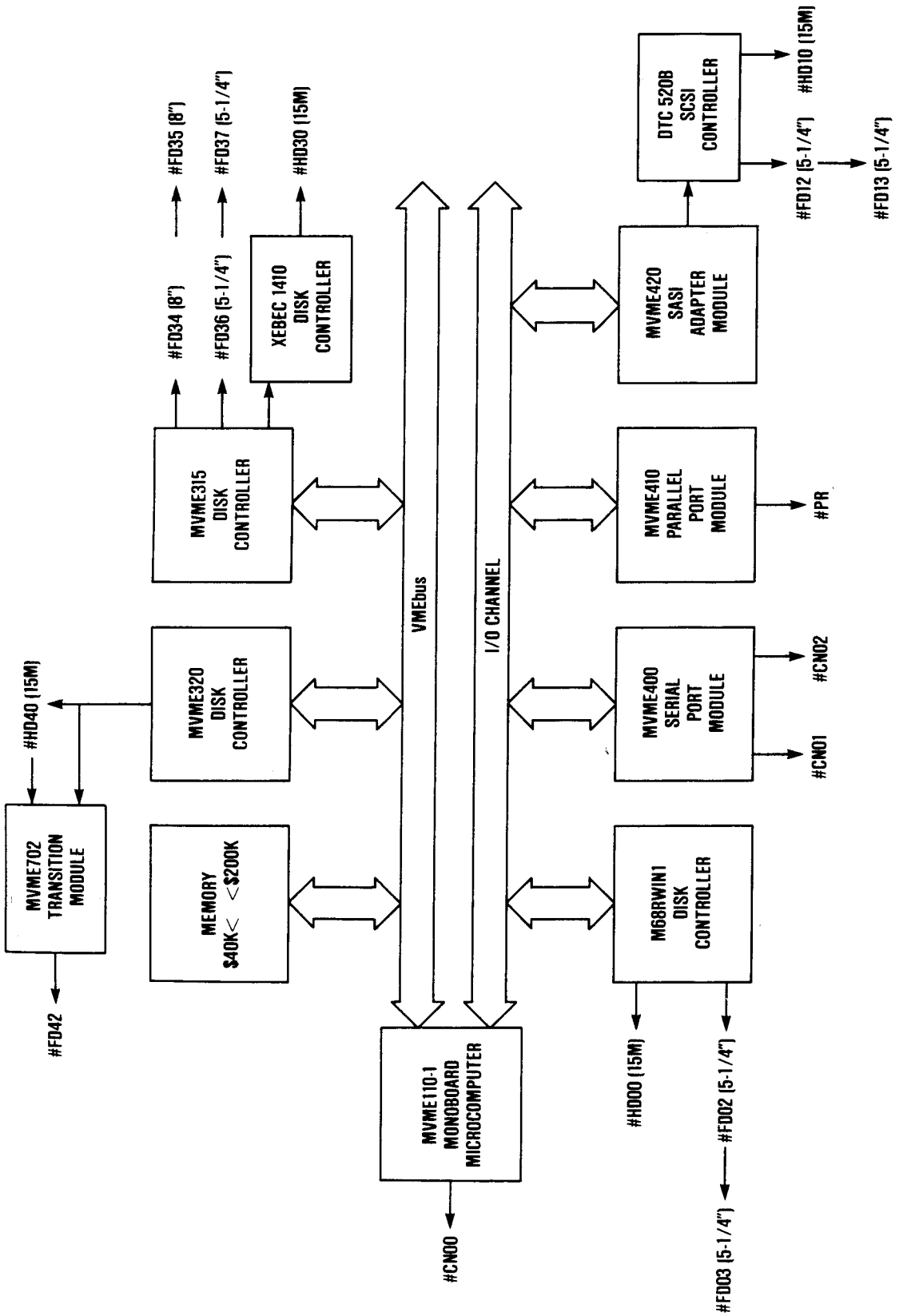


FIGURE 8-2. Standard MVME110-1 System

8.4 MVME115M SYSTEM

Table 8-3 lists the SYSGEN map for the standard VERSAdos-supported system based on the MVME115M Monoboard Microcomputer. Figure 8-3 is a block diagram of the standard MVME115M system.

TABLE 8-3. MVME115M Standard SYSGEN Map

FILENAME	TASK	PROC	SEG	ADDR	TCB
RMS.LO		RMS	RMS0	$010000	
			RMS2	$010100	
DRVLIB.LO		DRVL	DRVL	$015200	
TERMLIB.LO		TERM	TERM	$015400	
M320DRV.LO		M320	M320	$016700	
RWINDRV.LO		RWIN	RWIN	$017400	
DARTSPR.LO		DART	DART	$017F00	
DARTDRV.LO		DART	DART	$018100	
P115DRV.LO		P115	P115	$018600	
FHS.LO	.FHS		.FHS	$018D00	$01A100
IOS.LO	.IOS		.IOS	$01A300	$01BD00
FMS.LO	.FMS		.FMS	$01BF00	$021100
EET.LO	&EET		.EET	$021300	$025700
			.STT	$021600	
			&EET	$021900	
LDR.LO	&LDR		&LDR	$025900	$026E00
IOI.LO	.IOI		IOSG	$027000	$028200
			.IOI	$027D00	
SYSINIT.LO		SYSI	.INT	$028400	

- FINAL PC VALUE = $028F00
- START-UP ADDRESS = $028400
TOTAL NUMBER OF USER DEFINED SYMBOLS = 248

0 ERRORS ENCOUNTERED

8

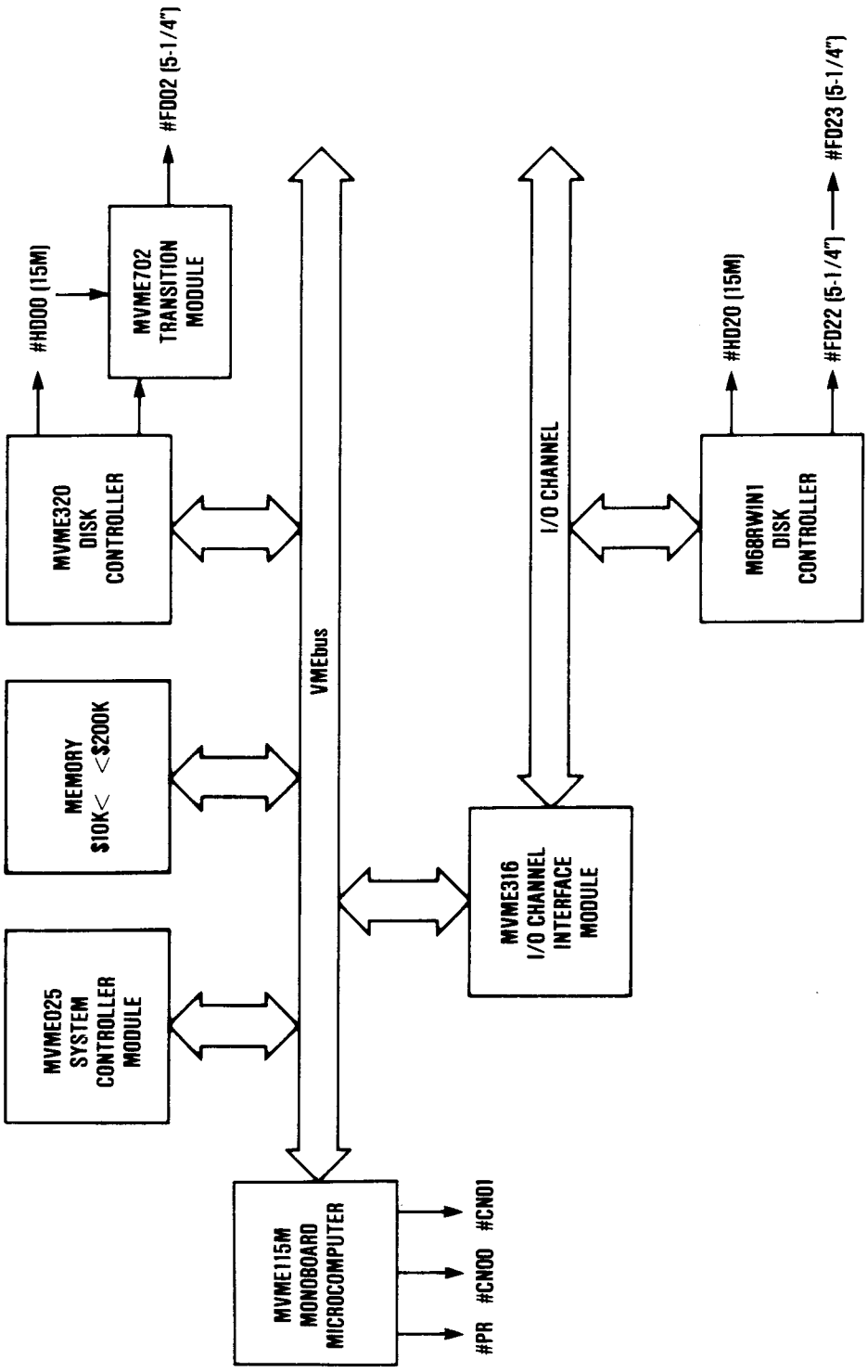


FIGURE 8-3. Standard MVME115M System

8.5 MVME120 FAMILY SYSTEMS

Table 8-4 lists the SYSGEN map of the standard VERSAdos-supported systems based on the MVME120, MVME121, and MVME123 Microprocessor Modules. Table 8-5 describes the SYSGEN map for the MVME122 module. Figure 8-4 is a block diagram of the standard system based on all four modules.

TABLE 8-4. MVME120 Family Standard SYSGEN Map

FILENAME	TASK	PROC	SEG	ADDR	TCB
RMS.LO		RMS	RMS0	$001000	
			RMS2	$001100	
DRVLIB.LO		DRVL	DRVL	$006200	
TERMLIB.LO		TERM	TERM	$006400	
M320DRV.LO		M320	M320	$007700	
M315DRV.LO		M315	M315	$008400	
RWINDRV.LO		RWIN	RWIN	$009100	
MFPDRV.LO		MFPD	MFP	$009C00	
MPCCDRV.LO		MPCC	MPCC	$00A000	
P050DRV.LO		P050	PRT5	$00A400	
FHS.LO	.FHS		.FHS	$00AB00	$00BF00
IOS.LO	.IOS		.IOS	$00C100	$00DB00
FMS.LO	.FMS		.FMS	$00DD00	$012F00
EET.LO	&EET		.EET	$013100	$017500
			.STT	$013400	
			&EET	$013700	
LDR.LO	&LDR		&LDR	$017700	$018C00
IOI.LO	.IOI		IOSG	$018E00	$01B500
			.IOI	$01B000	
SYSINIT.LO		SYSI	.INT	$01B700	

```
- FINAL PC VALUE = $01C200
- START-UP ADDRESS = $01B700
TOTAL NUMBER OF USER DEFINED SYMBOLS =   268

   0  ERRORS ENCOUNTERED
```

8

TABLE 8-5. MVME122 Standard SYSGEN Map

```
================================================================================
FILENAME          TASK    PROC   SEG    ADDR         TCB
================================================================================

RMS.LO                    RMS    RMS0   $001000
                                 RMS2   $001100
DRVLIB.LO                 DRVL   DRVL   $006300
TERMLIB.LO                TERM   TERM   $006500
RWINDRV.LO                RWIN   RWIN   $007800
TERMDRV.LO                TERM   TERM   $008300
MPSCSPR.LO                MPSC   MPSC   $00BB00
MPSCDRV.LO                MPSC   MPSC   $00BD00
PIADRV.LO                 PIAD   PIAD   $00C400
FHS.LO           .FHS            .FHS   $00CA00     $00DE00
IOS.LO           .IOS            .IOS   $00E000     $00FA00
FMS.LO           .FMS            .FMS   $00FC00     $014E00
EET.LO           &EET            .EET   $015000     $019400
                                 .STT   $015300
                                 &EET   $015600
LDR.LO           &LDR            &LDR   $019600     $01AB00
IOI.LO           .IOI            IOSG   $01AD00     $01BF00
                                 .IOI   $01BA00
SYSINIT.LO                SYSI   .INT   $01C100

- FINAL PC VALUE = $01CD00
- START-UP ADDRESS = $01C100
TOTAL NUMBER OF USER DEFINED SYMBOLS =   237

   0  ERRORS ENCOUNTERED
================================================================================
```

8

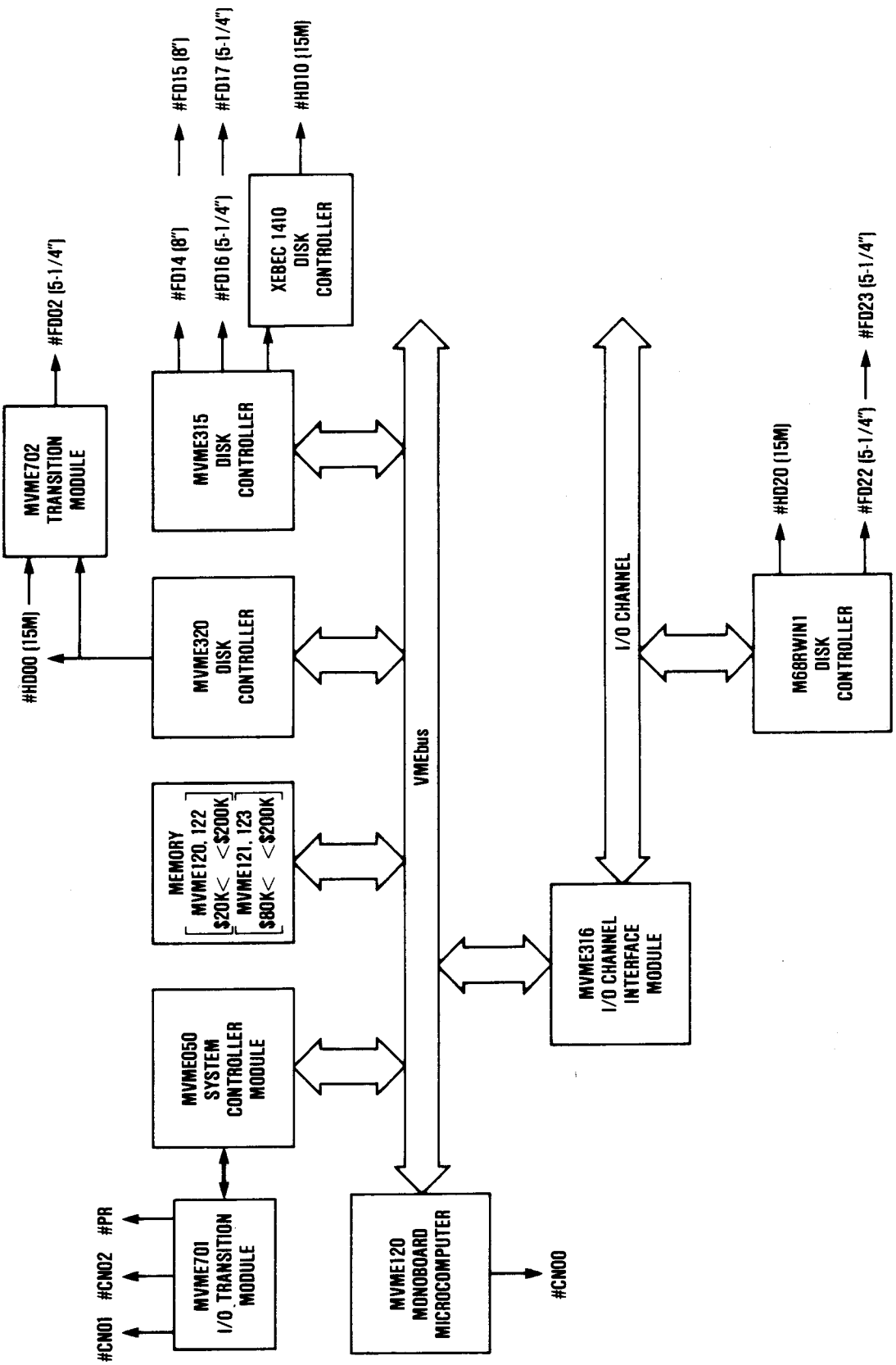


FIGURE 8-4. Standard MVME120 Family System

8

8.6 VME/10 SYSTEM

Table 8-6 lists the SYSGEN map for the standard VERSAdos-supported system based on the VME/10 Development System. Figure 8-5 is a block diagram of the standard VERSAdos-supported VME/10 system.

TABLE 8-6. VME/10 Standard System SYSGEN Map

FILENAME	TASK	PROC	SEG	ADDR	TCB
RMS.LO		RMS	RMS0	$001000	
			RMS2	$001100	
DRVLIB.LO		DRVL	DRVL	$006000	
TERMLIB.LO		TERM	TERM	$006200	
M320DRV.LO		M320	M320	$007500	
M315DRV.LO		M315	M315	$008200	
RWINDRV.LO		RWIN	RWIN	$008F00	
MFPDRV.LO		MFPD	MFP	$009A00	
MPCCDRV.LO		MPCC	MPCC	$009E00	
P050DRV.LO		P050	PRT5	$00A200	
FHS.LO	.FHS		.FHS	$00A900	$00BD00
IOS.LO	.IOS		.IOS	$00BF00	$00D900
FMS.LO	.FMS		.FMS	$00DB00	$012D00
EET.LO	&EET		.EET	$012F00	$017300
			.STT	$013200	
			&EET	$013500	
LDR.LO	&LDR		&LDR	$017500	$018A00
IOI.LO	.IOI		IOSG	$018C00	$01B300
			.IOI	$01AE00	
SYSINIT.LO		SYSI	.INT	$01B500	

- FINAL PC VALUE = $01C000
- START-UP ADDRESS = $01B500
TOTAL NUMBER OF USER DEFINED SYMBOLS = 268

0 ERRORS ENCOUNTERED

8

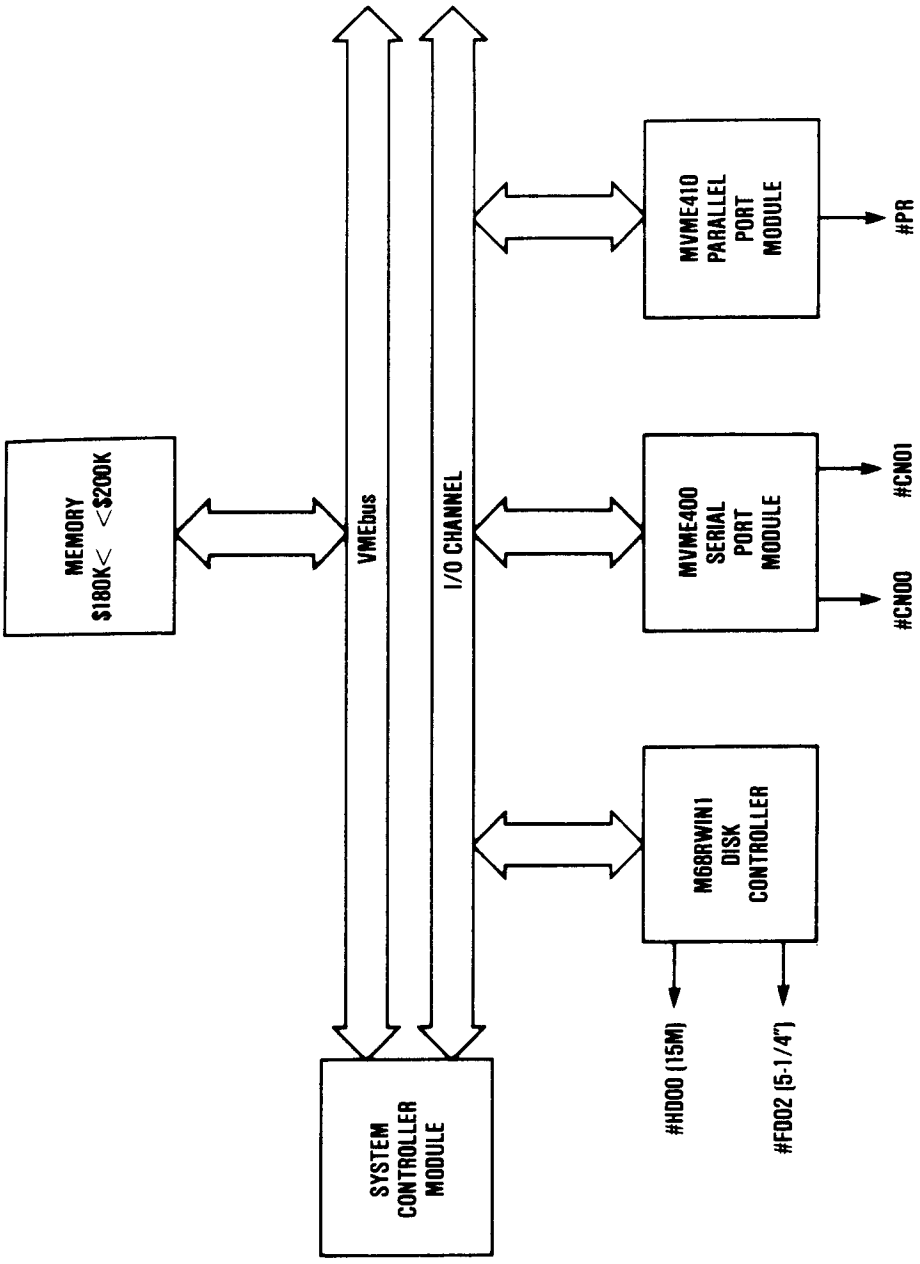


FIGURE 8-5. Standard VME/10 System

8.7 STANDARD CONFIGURATION PROCEDURE

The following step-by-step instructions describe the procedure necessary to properly configure a specific set of hardware. Once the hardware is set and verified as being functional, the user will be able to boot the VERSAdos 4.4 operating system standard boot file.

It is noted here that while the following set of instructions pertain specifically to configuring an MVME120-based system, these guidelines can be applied to other VMEsystem configurations.

STEP 1

Clear a suitable working area large enough to comfortably set-up the hardware. Allocated space should be based on the number of boards, size of chassis, and size of periphals.

STEP 2

Set-up the VME chassis and power supply. Plug chassis into suitable A/C outlet and verify with VOM that voltages are correct.

STEP 3

Locate the MVME050 and MVME120 boards, then verify the jumpers are in the correct positions according to paragraphs 2.4 and 2.6 in this manual.

REMOVE POWER FROM THE VME CHASSIS

STEP 4

Plug the MVME050 into slot 1 of the VME chassis and remove all backplane jumpers in this slot.

STEP 5

Plug the MVME120 into slot 2 of the VME chassis and remove all backplane jumpers in this slot.

STEP 6

Connect a terminal to the RS-232C connector on the front of the MVME120.

STEP 7

Apply power to the VME chassis and turn the terminal ON. Depress the reset button on the MVME050. This should result in the Debug Monitor prompt appearing on the screen:

VME120 1.1>

Test the local memory using the BI, BT or MD functions. Test the proper operation of all local peripherals such as the local serial port (in use at this point). The MVME050 serial ports and the MVME050 printer port can be tested by using the TA and PA commands.

DISCONNECT POWER FROM VME CHASSIS

STEP 8

Install the memory board(s) making sure they are jumpered somewhere in the address space from the end of MVME120 onboard memory to $200000. It is generally more efficient to have VMEbus memory boards jumpered contiguous to the onboard memory of the processor.

STEP 9

Apply power to VME chassis and test the memory boards just installed using the MVME120 Debug Monitor commands BI, BT, or MD.

DISCONNECT POWER FROM THE VME CHASSIS

STEP 10

Install the MVME320 disk controller. If empty slots are left between the MVME120 and the MVME320, then the user must jumper the daisy-chained Bus Grant and IACK pins on the VME backplane. Connect the control cable to the front of the MVME320 board J3 to the disk chassis. It is assumed that the user has some kind of box to contain the disk drives and power supply. If the MVME702 transistion board is being used, review Figure 4-2. Connect the hard disk data cable(s) to the MVME320 J1 and J2 connectors.

STEP 11

Apply power to the VME chassis and then apply power to the disk drive box. If the hard disk drive has been initialized by a MVME320 controller, the MVME120 Debug Monitor command IOP may be used to read sectors to verify that the drive and controller are working properly. If the hard disk has not been initialized, install a known working 5-1/4 inch floppy disk in the floppy disk drive. Use the MVME120 Debug Monitor command IOP to read sectors from the floppy to ensure that the disk and controller are working properly.

STEP 12

Locate the VERSAdos 4.4 package of 5-1/4 inch floppy disks. Install the disk that is MVME120-bootable into the 5-1/4 inch drive. Reset the MVME050 controller and answer "Y" to the MVME120 Debug Monitor initialize memory prompt. At the Debug Monitor prompt, enter the command:

```
VME120 1.1>BO 2,0,VME120.VERSADOS.SY
```

STEP 13

VERSAdos will clear the terminal screen and request a DEFAULT VOLUME and USER. Enter the name of the floppy disk that was booted from, then enter the date and time when requested.

STEP 14

Initialize the hard disk and commence copying the release floppy disk to the hard disk. The system is ready to operate.

CHAPTER 9

SYSTEM MODIFICATIONS

9.1 INTRODUCTION

This Chapter deals with modifications to the standard system configurations defined in Chapter 8.

In order to make the task of configuring a VMEsystem using VERSAdos an orderly one, several questions must be answered by the user. Topics that need to be addressed include:

- Is memory located where it is supposed to be and does it function properly?
- Can the rotating disk media be read?
- Does the information in sectors 0 and 1 appear reasonable? If so, is the BH command in the Debug Monitor functioning?

Once these issues are resolved, the Debug Monitor in use will read the Initial Program Loader (IPL) into memory. The user will then be able to boot the file VMExxx.VERSADOS.SY in order to receive the log-on message.

This chapter deals with modifications to the standard system configurations defined in Chapter 8. In doing so, the information provided can be used t answer the above questions so that the user may design a non-standard VERSAd system.There are only two files necessary to edit for most changes to stand VERSAdos-supported products. They are VERSADOS.CD and <system>.CNFGDRVR.C where <system> is the catalog name of the board or system being used. For example, the file VME110.CNFGDRVR.CI is used for any MVME110-1 based system, and the file VME120.CNFGDRVR.CI is used for any MVME120-based system.

9.2 ADDING MEMORY

Most of the standard system configurations support additional memory simply by jumpering the memory boards and plugging them into the VMEbus. There may be occasions when the standard VERSAdos does not support the desired target configuration. This section describes the methods used to implement various memory configurations. Each example describes the memory requirements of a particular application and the necessary steps required to meet the needs of the application.

Example: Default VERSAdos for the MVME110-1 supports one memory partition up to $200000. Target system requires memory from 0 to $100000.
Solution: Jumper memory boards, plug into VMEbus. No SYSGEN required.

Example: Default VERSAdos for the MVME110-1 supports one memory partition up to $200000. Target system requires memory from 0 to $280000.
Solution: Change parameter MEMEND3 from $200000 to $280000. Perform SYSGEN.

Example: Default VERSAdos for the MVME101 supports one memory partition up to $200000. Target system requires memory from $10000 to $80000, and from $100000 to $180000.
Solution: Jumper memory boards, plug into VMEbus. No SYSGEN required.

Example: Default VERSAdos for the MVME110-1 supports one memory partition up to $200000. Target system requires memory from 0 to $30000 for operating system memory as partition zero, $30000 to $80000 for user task memory as partition one, and $80000 to $C0000 as special memory partition and type known only to one task.
Solution: Change SYSGEN parameter MEMEND1 from $20000 to $30000 resulting in partition zero from 0 to $30000. Change MEMEND2 from 0 to $30000 and MEMEND3 from 0 to $80000 resulting in partition one from $30000 to $80000. Add a new MTENTRY macro in the file INITDAT.AG as follows. Define in the <system>.MTYPE.SI file entries for the new type and partition. The example below shows this file with the necessary additions. Perform SYSGEN.

9

Example: Default VERSAdos for the MVME120 supports two memory partitions, 0 to $27F00 and $27F00 to $200000. Target system requires memory from 0 to $30000, $40000 to $80000, and $100000 to $140000. Memory gaps from $30000 to $40000 and from $80000 to $100000 present no problem.
Solution: Jumper memory boards, plug into VMEbus. No SYSGEN required.

9.2.1 Memory Type Addresses

The following listing shows the absolute address for the beginning and end of a new partition. This is shown for illustration only.

```
* &.MTYPE.SI - Memory type assignments for standard systems
*
*  System memory
MTYPE$SRO  EQU MTYP0 read only
MTYPE$SRW  EQU MTYP0 read/write
*
*  User memory
MTYPE$URO  EQU MTYP0 read only
MTYPE$URW  EQU MTYP0 read/write
*
*  Partition types
MTYPE$P0   EQU MTYP0 partition #0
MTYPE$P1   EQU MTYP0 partition #1
MTYPE$P7   EQU MTYP7 partition #7
*
MEMTABL    EQU    *

           MTENTRY RAM,$00000,\MEMEND1,MTYPE$P0,PART0,TOP
           IFNE   \MEMEND2
              MTENTRY RAM,\MEMEND2,\MEMEND3,MTYPE$P1,PART1,BOTTOM
           ENDC
********* new entry
*
           MTENTRY RAM,$80000,$C0000,MTYPE$P7,PART7,TOP
*
********* end new entry

           IFNE   \$ROPT
              MTENTRY ROM,\ROMSADDR,\ROMEADDR
           ENDC
           MTEND
           DS.L   10
ENDMEMT    EQU    *
           END
```

As stated earlier, this listing is shown for illustrative purposes only. The preferred method is to define two additional substitution parameters in the <system>.SYSTEM.CI file such as MEMEND4 and MEMEND5. The entry then would look as follows:

```
MTENTRY RAM,\MEMEND4,\MEMEND5,MTYPE$P7,PART7,TOP
```

9

9.2.2 Memory Configuration Files

Files that are of interest to the system programmer when changing memory include:

<system>.SYSTEM.CI where <system> is the catalog of the desired target system (i.e., VME/10, MVME110-1, MVME120 ... etc.). This file contains the MEMEND parameters for the existing default memory configurations.

<system>.MTYPE.SI contains the memory type definitions and assignments for the <system>. The user may add new type equates in this file.

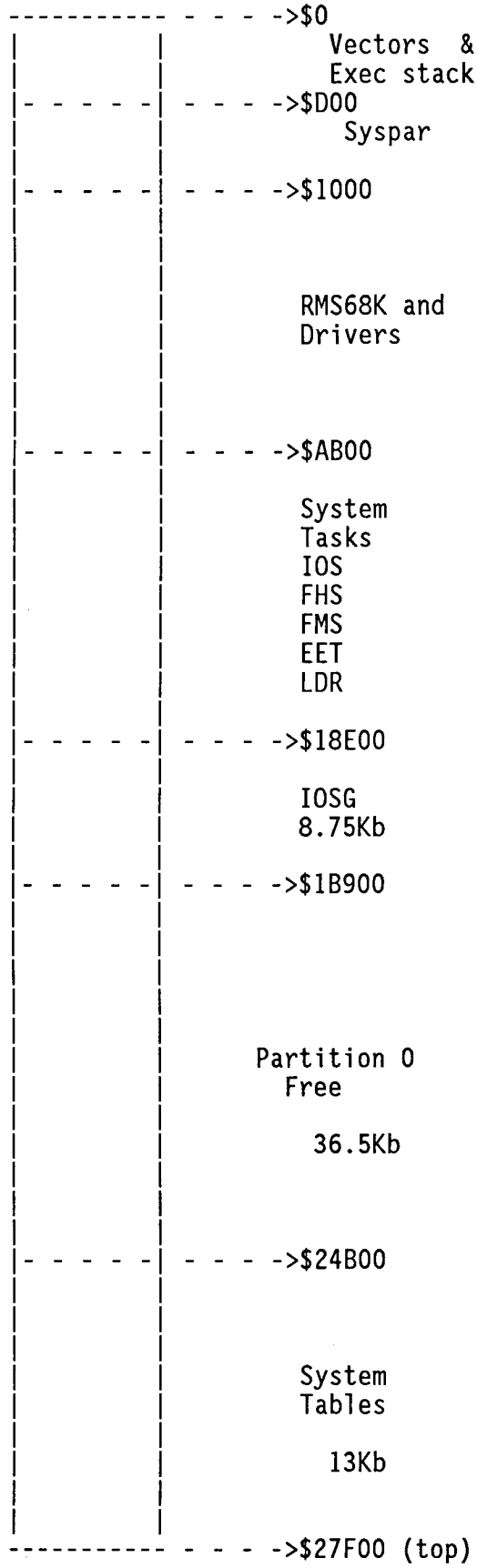
&.INITDAT.AG contains all the SYSPAR definitions and memory table macros. This file is assembled at SYSGEN time.

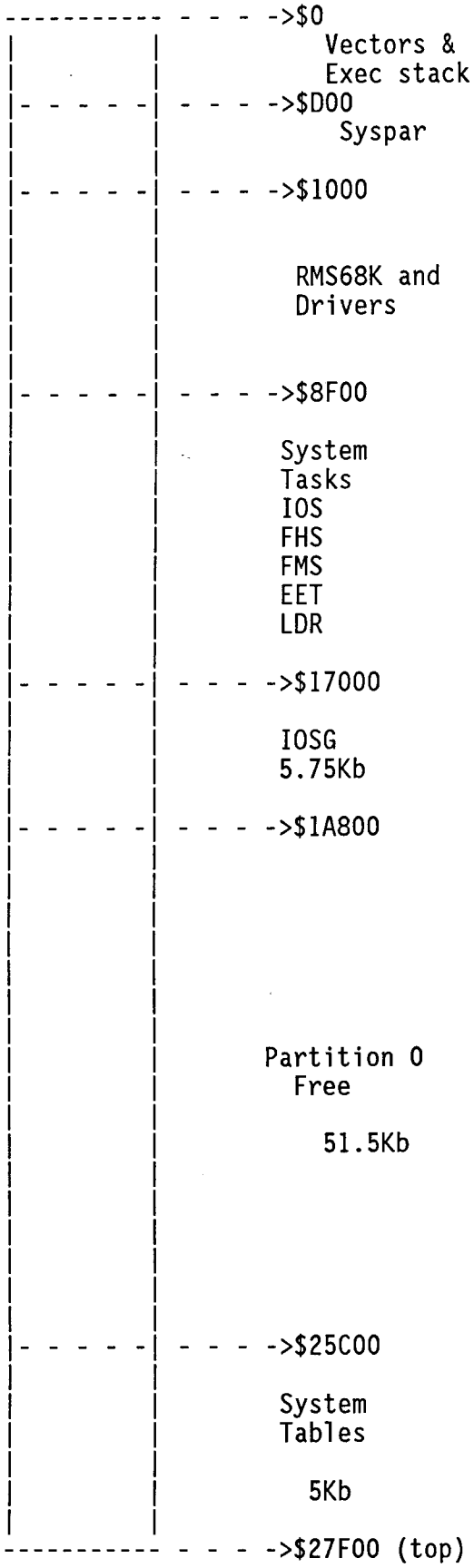
9.2.3 Altering Memory Resources

During the development and testing phase of a project it is necessary to support more features and devices than actually required in the final production version of the product. The following memory maps show the allocation of memory resources for systems used in different applications. Changes in the memory configuration are often required during the process of developing a target system.

The left-side map details the memory configuration of Partition 0 for a standard VERSAdos 4.4 system. The free area of this partition will be allocated for TCBs and ASQs only. Partition 1 from $27F00 to $200000 would be free for user tasks. All tasks will reside in Partition 1, although their TCB will come out of part 0. As tasks are loaded, the free area decreases by .5Kb. The only other significant piece of system memory is allocated in partition 1 for the FMS Data segment. This is currently 32Kb.

The right-side map shows the allocation of memory resources after a SYSGEN to configure a smaller system. By reducing the number of devices, the user saves 7Kb by eliminating drivers and 3k of IOSG space. The FMS data segment was reduced from 32Kb to 15Kb (not shown). The total saving realized from re-SYSGEN for this smaller system is approximately 20k. Due to space constrants, the relative size of the maps is distroted.


->$0
Vectors &
Exec stack
->$D00
Syspar
->$1000
RMS68K and
Drivers
->$AB00
System
Tasks
IOS
FHS
FMS
EET
LDR
->$18E00
IOSG
8.75Kb
->$1B900
Partition 0
Free
36.5Kb
->$24B00
System
Tables
13Kb
->$27F00 (top)


->$0
Vectors &
Exec stack
->$D00
Syspar
->$1000
RMS68K and
Drivers
->$8F00
System
Tasks
IOS
FHS
FMS
EET
LDR
->$17000
IOSG
5.75Kb
->$1A800
Partition 0
Free
51.5Kb
->$25C00
System
Tables
5Kb
->$27F00 (top)

9

9.3 ADDING SERIAL DEVICES

Most of the standard VERSAdos configurations support multiple terminals; however, this may not be sufficient for a specific target system. The following example illustrates the method used to alter serial devices in a VERSAdos configuration without changing default parameters.

Example: Standard VERSAdos does not support the MVME400 module on the I/O Channel. Target system requires three additional serial ports.
Solution: Edit the <system>.CNFGDRVR.CI file. Locate the following series of commands and change as indicated. Perform SYSGEN.

```
NVME400  = 2 (was 0)        Number of MVME400 dual 7201 serial port boards
IFGT  \NVME400
   NU400$1    = 2 (was 0)   Number of ports/users on MVME400 #1; max= 2/bd.
   NU400$2    = 1 (was 0)   Number of ports/users on MVME400 #2; max= 2/bd.
ENDC
```

9.4 CHANGING DEFAULT PARAMETERS

The following example provides a step-by-step procedure for altering standard default parameters of a particular system.

Example: The user starts with the standard configurations for the MVME110-1 system and supported modules. Assume a smaller system is required for a particular application.
Solution: The following is a step-by-step procedure for changing the standard configuration.

STEP 1

Review the default system configuration and the particular application requirements to determine what must be eliminated and/or added.

Default system supports:	Application requires:
MVME110-1	MVME110-1
RWIN1 1 HARD 2 FLOPPY'S	RWIN1 2 HARD 1 FLOPPY
VME420 W/SASI 5-1/4" 1 HARD 2 FLOPPY'S	DON'T NEED SASI
VME315 1 HARD 4 FLOPPY'S	DON'T NEED VME315
VME320 1 HARD 1 FLOPPY	DON'T NEED VME320
1 LOCAL TERMINAL	1 LOCAL TERMINAL
VME400 W/2 TERMINALS	DON'T NEED VME400
1 VME410 W/1 PRINTER	1 VME410 W/1 PRINTER
MEMORY FROM $40000 TO $200000	MEMORY $40000 TO $100000

Now that the major hardware features of the system have been defined, the necessary software requirements need to be determined. Features to be examined include the total numbers of tasks expected to execute at one time, the number of files to be opened at once, and the sizes of various system data tables. A new feature of the VERSAdos 4.4 release enables the SYSGEN process to produce a SYMBOLS listing containing all of the symbols defined during SYSGEN, their final value, and where they are defined and used. This is a useful tool in determining what parameters to set and how large or small to make them.

STEP 2

Select a user number on the development system to be used for SYSGEN. It is suggested that when performing multiple SYSGENs for various target configurations, a separate user account should be set aside for each configuration.

Example:

```
SYS:9100.VME101          Containing all files for MVME101 SYSGENs.
SYS:9101.VME110          Containing all files for MVME110-1 SYSGENs.
```

Assume the user number 9101 has been selected on the default volume of SYS. It may be desirable to delete all files from that account making it free for MVME110-1 work. Logon to that user number and set the catalog. Execute the COPYSGEN chain file to accumulate all files necessary for the SYSGEN.

```
=SYS:9998.VME110.COPYSGEN.CF SYS,SYS,9101
```

This procedure generally takes about 15 minutes to copy all files. The above arguments are defined in VME110.COPYSGEN.CF.

```
=/*
=/*  VME110.COPYSGEN.CF  -  Chain file to copy files for SYSGEN
=/*
=/*  --  FORCE CATALOG TO BE SET = VME110
=USE :.VME110
=/*
=/*      This  chain file copies those files required to perform a SYSGEN into
         a specified volume and user number.
=/*
=/*   Invoke as:    =VME110.COPYSGEN.CF  arg1,arg2,arg3,arg4,arg5
=/*     where:
=/*         Arg1= \1    Source volume name            (must specify)
=/*         Arg2= \2    Target volume name            (must specify)
=/*         Arg3= \3    Target user number            (must specify)
=/*         Arg4= \4    COPY option letter(s)         (optional)
=/*                       B = both copy & verify
=/*                       P = pack files
=/*                       Y = overwrite if file exists
=/*                       N = skip copy if file exists   (default)
=/*                      VT = verify & show only 1st difference
=/*         Arg5= \5    include PCDRV driver files    (optional)
=/*                       PCDRV = include them
=OPT N
```

STEP 3

Using the Editor, review the &.VERSADOS.CD file for any parameter changes appropriate for the system configuration. Most parameters in this file are automatically calculated for a "typical" target.

However, a target system that Motorola has defined as "typical" may not satisfy the requirements of the intended application. Review the descriptions of each parameter. Make any changes that appear appropriate at this time.

This step is especially important when standard parameters do not meet the needs of the target system. For instance, this specific application may have a large number of tasks. Some of these tasks have a large number of data files open with different logical unit numbers. The following changes may be appropriate for this situation.

PARAMETER	ORIGINALLY	CHANGED TO	COST/BENEFIT
UST	2	1	saves 256 bytes
Reason:	Only one of the tasks has need of a semaphore.		
NOTASKS	6+(3*\totterm)	20	
Reason:	There are more tasks than the formula would define with only one terminal.		
LUMAX bytes	\totdsk+3	30	cost 8 bytes #lu's
Reason:	Some of the tasks require more than the formula would define with only three disks.		

NOFILES	2*\notasks	3*\notasks	cost 176 bytes per file + 2Kb

Reason: Some tasks may require more than three files but the average for all tasks is greater than 2. Compromise is 3.

GST	(\nofiles/14)+1	4	cost 256 bytes/page

Reason: Some tasks have multiple shared segments which require an entry in GST. Change to 4 as a precaution.

STEP 4

Using the editor, review the VME110.CNFGDRVR.CI file to configure the required boards and drivers. Make the following changes for the example system described in STEP 1.

PARAMETER	WAS	CHANGE TO	COST/BENEFIT
NHRWIN$1	1	2	save approx. 2-1/4Kb (FMS buffers)

Reason: Need two hard disks.

NFRWIN$1	2	1	save approx. 2-1/4Kb (FMS buffers)

Reason: Need only one floppy disk.

NVME315	1	0	eliminate driver and system space 4Kb

Reason: System does not require MVME315 disk controller.

NVME320	1	0	eliminate driver and system space 4Kb

Reason: System does not require MVME320 disk controller.

NVME400	1	0	eliminate device-dependent driver and system space 500 bytes

Reason: Only need one terminal port.

NVME420	1	0	eliminate device-dependent driver and system space 4Kb

Reason: System does not require MVME420 disk controller

STEP 5

The system is now ready to execute SYSGEN. For convenience, use the supplied STD.SYSGEN.CF file which has pre-defined all of the arguments. The calling sequence is =STD.SYSGEN.CF.

STEP 6

With the SYSGEN completed, there is still a VME110.VERSADOS.SY file under user :9101. Copy this file to a bootable MVME110-1 5-1/4 inch diskette and boot up the target system. If the user has been developing and performing SYSGENs on a MVME110-1 system, testing can be done on this system but the user should boot directly from the 9101 account rather than copying to the default user 0 file. In case something has gone wrong in the SYSGEN process, do not overwrite the existing VERSADOS.SY file until the new one has been tested. The command from the MVME110-1 Debug Monitor is:

```
VMEbug 4.0 >BO 0,0,:9101.VME110.VERSADOS.SY

                THE SYSTEM IS UP AND RUNNING!
```

Review: The files necessary to change any of the standard Motorola-supported system configurations are:

&.VERSADOS.CD	contains parameters common to all system configurations
<system>.CNFGDRVR.CI	where <system> specifies the various catalog names for support systems (e.g., VME/10, MVME110-1, MVME120 ... etc.). This file contains parameters to define the various device drivers for the boards specified.

9.5 ADDING AN OPERATING SYSTEM TASK TO THE VERSAdos SYSTEM

The following steps describe the procedure required to add an operating system task to a standard VERSAdos 4.4 system.

STEP 1

Write the code necessary to handle the desired operating system functions. This could be a particular task which handles events from several user tasks and sends events to other user tasks based on the inputs received. This could be a Server for a TRAP #'x' where 'x' is a number other the those currently assigned to VERSAdos. TRAPS #0-4 are currently used by VERSAdos.

STEP 2

Debug the task. This could be accomplished prior to building the task into the operating system by using the firmware debugger for the system processor module.

STEP 3

Add the necessary commands to the file CNFGTASK.CI. The following listing shows the parameters that need to be confirmed.

```
*
*        &.CNFGTASK.CI
*
MSG
MSG              OPERATING SYSTEM TASK SWITCHES
MSG
*
FHS$IOS$  = 1            Set =0 for skip FHS/IOS module, not =0 to include it
FMS$      = 1            Set =0 for skip FMS module, not =0 to include it
EET$      = 1            Set =0 for skip EET module, not =0 to include it
LDR$      = 1            Set =0 for skip LDR module, not =0 to include it
*********************
NEWTSK$   = 1                EXAMPLE
*********************
```

The parameter NEWTSK$ will then be used in the file IFTASK.CI to determine if the task should be included in the system.

```
*
*        &.IFTASK.CI
*
MSG
MSG            PROCESS O/S TASKS AS REQUIRED
MSG
IFNE         \FHS$IOS$+\FMS$
   SUBS         GET.TASKID.AG
   ASM          GET.TASKID.AG,GET.TASKID.RO,\ASMLS;R
   IFEQ         \ASMLSW
      =COPY        \ASMLS,\WORKLS;A
   ENDC
ENDC
IFNE         \FHS$IOS$
   INCLUDE      FHSIOS.VERSADOS.CI
ENDC
IFNE         \FMS$
   INCLUDE      FMS.VERSADOS.CI
ENDC
IFNE         \EET$
   INCLUDE      EET.VERSADOS.CI
ENDC
***************
IFNE          \NEWTSK$
   INCLUDE     NEWTASK.VERSADOS.CI
ENDC
***************
IFNE         \LDR$
   IFNE         \MMU
      INCLUDE      MMULDR.VERSADOS.CI
   ENDC
```

```
   IFEQ        \MMU
      INCLUDE     NOMMULDR.VERSADOS.CI
   ENDC
ENDC
```

STEP 4

Build the necessary NEWTASK.VERSADOS.CI file containing the commands to build the new task into the system.

```
*
*        NEWTASK.VERSADOS.CI
*
MSG
MSG                    NEWTASK -- TRAP X SERVER
MSG
TASK         &.NEWTASK.LO,.NEW
STATE      = 'DORM'
ATTRIB     = 'CRIT'
ATTRIB     = 'RTIM'
ATTRIB     = 'SYST'
PRIORITY   = $D0
NEWSTR     = *             NEW load addr.
NEWASR     = *+2           NEW ASR entry point
SUBS         &.NEW.LG
LINK         &.NEW.LG
IFEQ         \LINKLSW
   =COPY        \LINKLS,\WORKLS;A
ENDC
END          NEW
* Build VERSAdos  <system>.OSLIST.AG
=COPY        OSLIST.NEW.SI,OSLIST.AG;A
* Build VERSAdos patch chain file <system>.VERSAPT.CF
=COPY        VERSAPT.NEW.CF,VERSAPT.CF;A
```

STEP 5

Perform SYSGEN and test. If the I/O Initializer (IOI) task is to perform such duties as allocating data segments and starting the target task, the user must add the appropriate command and data blocks to <system>.OSLIST.AG

```
*
*        OSLIST.BEGIN.AG
*
         XDEF      OSMTBL
         XDEF      OSMEND
*
*        OPERATING SYSTEM MODULE TASK NAMES
*
```

```
*   Included equate files:
*         INCLUDE   &.IOE.EQ
*         INCLUDE   &.EXE.EQ
*         INCLUDE   &.NIO.EQ
*         INCLUDE   &.FME.EQ
*         INCLUDE   &.FMI.EQ
          NOLIST
          INCLUDE   &.IOE.EQ
          INCLUDE   &.EXE.EQ
          INCLUDE   &.NIO.EQ
          INCLUDE   &.FME.EQ
          INCLUDE   &.FMI.EQ
          LIST
 SECTION 1

*
*         The following NOP is required if the user has elected to eliminate
*         all modules from OSLIST such that it becomes a null file. Since
*         LINK 1.6 has difficulty in linking a null module that has XREF and
*         XDEF references, the NOP is here to keep this module from being a
*         null module.
*
          NOP

OSMTBL:
*
*         FHS
*
          DC.B      $20                 STARTUP VALUE
*                                       This value will determine (low to high)
                                        the  order  a  task is to be started by
                                        IOI.

          DC.B      1                   BIT NUMBER
*                                       This value represents a bit number
*                                       that must be set by the task being
*                                       initiated IOI will attempt to start the
*                                       next task.  Along with STARTUP VALUE,
*                                       this allows the coordination of
*                                       bringing up system tasks at boot-time
*                                       in an orderly process. The offset
*                                       System Value Table System Task Config-
*                                       uration Flag (SVTSTCF) is where the bit
*                                       must be set.  If the system task does
*                                       not require boot-time coordination, the
*                                       bit number must be set to $FF.
*
```

9

```
         DC.L    FHSID           FHS TASK NAME
         DC.L    IOSESS          SESSION
*
         DC.B    0               NO ASQ WAS REQUESTED
         DC.B    0
         DC.L    0
         DC.L    0
         DC.W    0
*
         DC.L    0,0,0,0         NO DATA SEGMENT NEEDED
*
         XDEF    OESIZE
*
*        IOS
*
IOSXXX:
         DC.B    $30             STARTUP VALUE
*                                This value will determine (low to high)
                                 the  order  a  task is to be started by
                                 IOI

         DC.B    0               BIT NUMBER
*                                This value, if present, represents a
*                                bit number that must be set by IOS
*                                before IOI will attempt to start the
*                                next task.  Along with STARTUP VALUE,
*                                this value allows the coordination of
*                                bringing up system tasks at boot-time
*                                in a orderly  process. The offset
*                                SVTSTCF is where the bit must be set.
*                                If the system task does not require
*                                boot-time coordination, the bit number
*                                must be set to $FF.
*
         DC.L    IOSID           IOS TASK NAME
         DC.L    IOSESS          SESSION
*
         DC.B    0               NO ASQ WAS REQUESTED
         DC.B    0
         DC.L    0
         DC.L    0
         DC.W    0
*
         DC.L    0,0,0,0         NO DATA SEGMENT NEEDED
*
OESIZE   EQU     *-IOSXXX
         XDEF    FMSDLEN
*
*        FMS
*
```

9

```
          DC.B      $10                 STARTUP VALUE
*                                       This value will determine (low to high)
                                        the  order  a  task is to be started by
                                        IOI

          DC.B      2                   BIT NUMBER
*                                       This value, if present, represents a
*                                       bit number that must be set by FMS
*                                       before IOI will attempt to start the
*                                       next task.  This, along with STARTUP
*                                       VALUE, allows the coordination of
*                                       bringing up system tasks at boot-time
*                                       in an orderly process.  The offset
*                                       SVTSTCF is where the bit must be set.
*                                       If the system task does not require
*                                       boot-time coordination, the bit number
*                                       must be set to $FF.

          DC.L      FMSID               FMS TASK NAME
          DC.L      IOSESS              SESSION
*
          DC.B      0                   NO ASQ WAS REQUESTED
          DC.B      0
          DC.L      0
          DC.L      0
          DC.W      0
*
          DC.W      SGOPPA              OPTIONS-LOG. ADDR=PHYS. ADDR.
          DC.W      SGATRW+SGATSS       ATTRIBUTES
          DC.L      FMSD                SEG. NAME
          DC.L      0                   LOGICAL ADDRESS
FMSDLEN   DC.L      0                   SEGMENT LENGTH
*
*
*         OSLIST.END.AG
*
OSMEND    EQU       *                   END OF TABLE
          END       OSMTBL
```

9.6 ADDING USER-WRITTEN COMMANDS TO THE SESSION MANAGER (EET)

The following step-by-step procedure is a simplified approach to adding user-written commands to the Session Manager task called &EET.

STEP 1

Write the code necessary to implement the new command. This code should be tested and debugged as a task using SYMbug.

STEP 2

Add the acronym for the new command to the file CMDLIST.AG. This file will be assembled at SYSGEN time. The user should XDEF the entry point to the command handling module here and XREF the entry point at the beginning of the module containing the code.

```
*
*        &.CMDLIST.AG
*
CMDLIST   IDNT   3,27 SESSION CONTROL COMMAND LIST
          SECTION 8
          XDEF   CMDACCTT
          XDEF   CMDPWVAL
          XDEF   COMTSKNO
          XDEF   DEFAULT
          XDEF   EET_GFLG
          XDEF   EETSBAT
          XDEF   INDFVOLN
          XDEF   INITPAR
          XDEF   INSWORD
          XDEF   LODRETRY
          XDEF   LOGM
          XDEF   LOGNTERM
          XDEF   LOGTRYNO
          XDEF   PATCH
          XDEF   SYSCMD
          XDEF   SYSCMDS
          XDEF   SYSOPT
          XDEF   SYSTBEND
          XDEF   TIMECNT
          XDEF   UCBMAXLU
          XDEF   UCBPRIBT
          XDEF   UCBPRICH
          XDEF   UCBPRILI
*
*        ** EQUATES
*
```

```
SPCCMD    EQU    \SPCCMD
CHAINBAT  EQU    \CHAINBAT
SECURITY  EQU    \SECURITY
TOTTERM   EQU    \TOTTERM
CONBATCH  EQU    \CONBATCH
BATCHPGE  EQU    \BATCHPGE
NOLOGON   EQU    \NOLOGON
MAXLU     EQU    \MAXLU
TERMOCNT  EQU    \TERMOCNT
BATDLY    EQU    \BATDLY
NOLOGONS  EQU    \NOLOGONS
*
EETPNAM   EQU    '&EET'
DCLSHR    EQU    7                 DECLARE SHAREABLE
GTASQ     EQU    31                GET ASQ
SGATRO    EQU    $4000             READ ONLY
SGATGS    EQU    $1000             GLOBALLY SHAREABLE
AQSTQE    EQU    1
SOSYSMSG  EQU    1                 BIT POSITION FOR INITIALIZING CHAINFILE
IOUCBK    EQU    $8001             CLAIM UNCLAIMED BREAKS
ISECLVL   EQU    0                 INITIAL SECURITY LEVEL (NONE)
ISWORD1   EQU    0                 INITIAL SECURITY WORD PART 1
ISWORD2   EQU    0                 INITIAL SECURITY WORD PART 2
MAXVNTLN  EQU    26                NO. BYTES SCT EVENTS (MAX)
MAXTSKNO  EQU    8  MAX. NO. OF TASKS COMMUNICATING WITH SCT (TERM,ABORT,ATTN)
IPRIUTSK  EQU    66                INITIAL    "   USER TASK
LPRIUTSK  EQU    127               LIMIT      "    "   "
IPRICTSK  EQU    66                INITIAL    "   CHAIN MODE USER TASK
LPRICTSK  EQU    127               LIMIT      "   CHAIN MODE USER TASK
IPRIBTSK  EQU    65                INITIAL    "   BATCH MODE USER TASK
LPRIBTSK  EQU    65                LIMIT      "   BATCH MODE USER TASK
*
*        ** OPTION BITS FOR SYSOPT (1=INCLUDE, 0=EXCLUDE)
*
INITFILE   EQU    1 ACTIVATE INITIALIZING CHAINFILE (PRIV.UPSYSTEM.NW) SESSION
1
*

*        ** IDENTIFICATION MESSAGE
LOGM      DC.B   $D8,'\LOGMSG1\REVNUMBR          \$DATE\$TIME'
          DS.W      0
          DS.B   LOGM+80-*
*
*        ** INITIALIZATION INFORMATION
*
INITPAR   DC.B   TOTTERM            NO. TERMINALS
          DC.B   CONBATCH           NO. CONCURRENT BATCH JOBS EXECUTING
          DC.B   BATCHPGE           NO. PAGES BATCH QUEUING (32 JOBS PER PAGE)
          DC.B   ISECLVL            INITIAL SECURITY LEVEL
INSWORD   DC.L   ISWORD1            INITIAL SYSTEM SECURITY WORD
          DC.L   ISWORD2
INDFVOLN  DC.L   '    '             SYSTEM DEFAULT VOLUME NAME (DYNAMIC)
          DC.L   0                  RESERVED
          DC.L   0                  RESERVED
```

9

```
COMTSKNO  DC.B   MAXTSKNO           MAXIMUM NO. TASKS COMMUNICATING WITH SCT
LOGTRYNO  DC.B   NOLOGON            NO. LOGON TRIES BEFORE REJECT LOGON
UCBPRILI  DC.B   IPRIUTSK,LPRIUTSK  PRIORITIES FOR ON LINE MODE USER TASKS
UCBPRICH  DC.B   IPRICTSK,LPRICTSK  PRIORITIES FOR CHAIN MODE USER TASKS
UCBPRIBT  DC.B   IPRIBTSK,LPRIBTSK  PRIORITIES FOR BATCH MODE USER TASKS
UCBMAXLU  DC.B   MAXLU              NUMBER MAX LUNS PER TASK
TIMECNT   DC.B   TERMOCNT           LOG OFF AFTER SPECIFIED NO. TIME OUTS
SYSOPT    DC.B   INITFILE<<SOSYSMSG SYSTEM OPTION BITS
          DC.B   0                  RESERVED
LODRETRY  DC.L   BATDLY             WAIT TO RETRY LOAD IF NO MEMORY AVAILABLE
          DC.L   0,0                RESERVED
PATCH     DS.B   $80                PATCH AREA
DEFAULT   DC.B   '\DEFAULT'         DEFAULT VOLUME:USN.CAT
          DC.B   $0D,$0A            CR,LF
          DS     0                  FORCE EVEN BYTE BOUNDARY
          PAGE
*
*        GENERAL USAGE FLAG FOR 'YES' OR 'NO' TYPE CONSIDERATIONS
*
EET_GFLG:
         DC.W     \AUTOLOGN
*                           BIT       MEANING
*                             0       AUTOLOGN
*                                     0 --> Auto break denied
*                                     1 --> Auto break requested
*                             1
*                                     0 --> Auto vol:usn.cat denied
*                                     1 --> Auto vol:usn.cat requested
*
*                           2-15      Available for use
*
LOGNTERM:
         DC.B     \AUTOTERM
*                           TERMINAL  ID  AUTO  LOGON  IS TO OCCUR ON SESSION
                            CONTROL COMMANDS
*
SYSCMDS   EQU    *
*
*        ** BASIC CONTROL COMMANDS
*
          XREF   LOGOFF,LOGBYE,LOAD,USTRT,STOPTASK,CONTINU
          XREF   TRMINATE,PROC260,LOGDATE,LOGTIME
          XREF   USE,DEFAULTS,ARG,NOARG
          XREF   BREAKS,BREAKT,OPTCMD,MENUCMD
          XREF   EETSXT,EETSBATE
SYSCMD    DC.B   'LOG OF',$E6,0     F
          DC.L   LOGOFF
          DC.B   'LOGOF',$E6,0      F
          DC.L   LOGOFF
          DC.B   'OFF',0
          DC.L   LOGOFF
          DC.B   'BYE',0
          DC.L   LOGBYE
```

```
            **************
            DC.B    'MENU', 0               EXAMPLE OF
            DC.L    MENUCMD                 NEW COMMANDS
            **************
            DC.B    'LOAD',0
            DC.L    LOAD
            DC.B    'STAR',$F4,0        T
            DC.L    USTRT
            DC.B    'STOP',0
            DC.L    STOPTASK
            DC.B    'CONT',$E9,$EE,$F5,$E5,0 INUE
            DC.L    CONTINU
            DC.B    'TERM',$E9,$EE,$E1,$F4,$E5,0 INATE
            DC.L    TRMINATE
            DC.B    '^',0               HALT TERMINAL INPUT
            DC.L    PROC260             GO WAIT FOR EVENT
            DC.B    'DATE',0
            DC.L    LOGDATE
            DC.B    'TIME',0
            DC.L    LOGTIME
            DC.B    'USE',0
            DC.L    USE
            DC.B    'DEF',$E1,$F5,$EC,$F4,$F3,0 AULTS
            DC.L    DEFAULTS
            DC.B    'ARG',$F5,$ED,$E5,$EE,$F4,$F3,0 UMENTS
            DC.L    ARG
            DC.B    'NOARG',$F5,$ED,$E5,$EE,$F4,$F3,0 UMENTS
            DC.L    NOARG
            DC.B    'BSTO',$F0,0        P
            DC.L    BREAKS
            DC.B    'BTER',$ED,0        M
            DC.L    BREAKT
            DC.B    'OPT',$E9,$EF,$EE,$F3,0 IONS
            DC.L    OPTCMD
*           ** OPTIONAL COMMANDS
            IFNE    SPCCMD
            XREF    NEWS,HELP,ASSIGN,CLOSE
            DC.B    'NEWS',0
            DC.L    NEWS
            DC.B    'HELP',0
            DC.L    HELP
            DC.B    'ASSI',$E7,$EE,0    GN
            DC.L    ASSIGN
            DC.B    'CLOS',$E5,0        E
            DC.L    CLOSE
            ENDC
*           ** CHAIN/BATCH OPTIONAL COMMANDS
            IFNE    CHAINBAT
            XREF    CHAIN,CMDEND,RETRY,PROCEED,REGCHAIN
            DC.B    'CHAI',$EE,0        N
            DC.L    CHAIN
            DC.B    'END',0
            DC.L    CMDEND
            DC.B    'RETR',$F9,0        Y
```

```
            DC.L    RETRY
            DC.B    'PROC',$E5,$E5,$E4,0 EED
            DC.L    PROCEED
            DC.B    'R?',0
            DC.L    REGCHAIN
            XREF    BATCH,CANCEL,ELIM,QUERY
            DC.B    'BATC',$E8,0        H
            DC.L    BATCH
            DC.B    'CANC',$E5,$EC,0    EL
            DC.L    CANCEL
            DC.B    'QUER',$F9,0        Y
            DC.L    QUERY
            DC.B    'ELIM',$E9,$EE,$E1,$F4,$E5,0 INATE
            DC.L    ELIM
            ENDC
*           ** SECURITY OPTIONAL COMMANDS
            IFNE    SECURITY
            XREF    PASS,SWORD,SECURE,PWVAL,ACCTINN
            DC.B    'PASS',$F7,$EF,$F2,$E4,0
            DC.L    PASS
            DC.B    'SWORD',0
            DC.L    SWORD
            DC.B    'SECURE',0
            DC.L    SECURE
            ENDC
SYSTBEND    DC.W    0,0,0
*
*           ** TRANSFER FOR BATCH TERMINATION **
*
            IFEQ    CHAINBAT
EETSBAT     BRA     EETSXT              IGNORE
            ENDC
            IFNE    CHAINBAT
EETSBAT     BRA     EETSBATE
            ENDC
*
*           ** TRANSFER FOR PASSWORD VALIDATION **
*
            IFEQ    SECURITY
CMDPWVAL    RTS
*
*
*           ENTRY POINT CMDACCTT MUST HAVE A6=SCT DATA SEGMENT ADDRESS
*
*           EXIT FROM CMDACCTT WILL HAVE A3=SCT DATA SEGMENT ADDRRESS
*
*
CMDACCTT:
            MOVE.L    A6,A3
*                               A3=SCT DATA SEGMENT ADDRRESS
            RTS
            ENDC
            IFNE    SECURITY
```

```
CMDPWVAL  BRA     PWVAL
*
*
*         ENTRY POINT CMDACCTT MUST HAVE A6=SCT DATA SEGMENT ADDRESS
*
*         EXIT FROM CMDACCTT WILL HAVE A3=SCT DATA SEGMENT ADDRRESS
*
*
CMDACCTT  BRA     ACCTINN
          ENDC
*
          DS.B    LOGM+$300-
*         SECTION 1
          XDEF    BEGQUEUE,ENDQUEUE,STTABL,STTABLND,JOBQUEUE,JOBQUEND
          XDEF    TESEGPGM,TESEGSTT,TASQPAR,BRKPAR
          XREF    EETASR
BEGQUEUE  DC.L    JOBQUEUE
ENDQUEUE  DC.L    JOBQUEND
STTABL    EQU
STTABLND  EQU     STTABL+(8*NOLOGONS)
JOBQUEUE  EQU     STTABLND
*
*         ** FOLLOWING EXEC & FHS CALLS EXECUTED AT INITIALIZATION
*         ** THEN CLEARED TO BECOME STT TABLE AND BATCH JOB QUEUE
*
TESEGPGM  DC.L    DCLSHR,0,0,SGATRO+SGATGS,EETPNAM DECLARE SHAREABLE
TESEGSTT  DC.L    DCLSHR,0,0,SGATRO+SGATGS,'.STT' DECLARE SHAREABLE
TASQPAR   DC.L    GTASQ,EETPNAM,0     GET ASQ
          DC.B    AQSTQE,MAXVNTLN
          DC.L    0                   QUEUE LENGTH INITIALIZED AT START UP
          DC.L    EETASR,0
BRKPAR    DC.W    IOUCBK,0
          DC.L    0
          DS.B    BEGQUEUE+$100
          DS.B    BATCHPGE-$100
JOBQUEND  EQU
*         END
```

STEP 3

Add to the Link file EET.LG the INPUT statement in order to include the code module.

```
=/*
=/*       &.EET.LG
=/*
=/* Link chain file run at SYSGEN time to link EET
=/*
=/* SYSGEN parameter LINKLS = \LINKLS = file/device to which to send
=/* the linker listing
=/*
```

9

```
=/* SYSGEN parameter EETSTR = \EETSTR = address at which to link driver
=/*
=LINK ,&.EET.LO,\LINKLS;HAMIXS
SEGMENT .EET:0 \EETSTR
SEGMENT .STT(G):1
SEGMENT &EET(RG):8,14
INPUT &.DAT.RO
INPUT &.CMDLIST.RO
INPUT &.EET.RO
INPUT &.SCTSASR.RO
INPUT &.USM.RO
INPUT &.SYSCNTRL.RO
INPUT &.MENU.CMD
LIB &.USMOPT.RO
LIB &.UTILIB.RO
END
=/*
=END
```

STEP 4

Execute SYSGEN. The user should then have the command installed in the Session Manager.

The search path for commands entered at the prompt is:

<cmdtable> -- <user default> -- <system default>

The Session Manager locates a command entered at the prompt (=) by searching the command table first for a match on the four characters entered. If a match is found, the corresponding code is executed for the desired command. If no match is found, the Session Manager assumes the command is a Load Module on the User Default Volume. The Session Manager attempts to load the desired module from disk. If an error 'file not found' is returned, the Session Manager will make one more attempt to load from the System Default Volume.

9

9.7 ADDING A USER-WRITTEN DEVICE DRIVER

The following general steps should be performed to integrate a user-written device driver for a custom-designed board into VERSAdos-based system. This section makes no assumptions about the type of driver being written. For more information, refer to the Guide to Writing Device Drivers for VERSAdos (M68KDRVGD).

STEP 1

Become familiar with the hardware characteristics of the device for which the driver is being written. This should include Control and Status register layout and command/status packet format (if applicable).

STEP 2

Write the device-specific routines needed to "talk" to the device. Build a <DRVNAME>.AF file for use in the assembly.

STEP 3

Become familiar with VERSAdos data structure for the Device Control Block (DCB), the Channel Data Block (CDB), and the macros used to generate the data structures. There will be one DCB for each device supported by this driver. There will be one CDB for each channel. A channel is defined as a single contiguous segment of memory mapped I/O space. The CDB will be used to build a data structure called the Channel Control Block (CCB).

STEP 4

Use an existing VERSAdos device driver as a guide, preferably one that comes close in functionality to the device currently being integrated. All driver source is included with VERSAdos object release.

a. Define the driver front end, address offset to interrupt, and command and initialization routines. Determine the number of pages needed for CCB allocation as well as revision and version information.

EXAMPLE:

SERVICE VECTOR TABLE, REVISION INFORMATION

This table is examined by Channel Management Request (CMR) when the channel is allocated to determine where to go for interrupts, commands, and initialization. It must always be at the beginning of the driver code. Also included is a parameter telling the number of extra pages of memory to include in the CCB, and information describing the revision of the driver.

```
 SECTION  0
 OPT    BRS

MFPDRV:

*
*  Service vector table.  Notice that these values are relative to the
*  beginning of the driver so the mechanism is position-independent!
*
          DC.L    INTERRUPT-MFPDRV    Address of interrupt service routine.
          DC.L    COMMAND-MFPDRV      Address of command service routine.
          DC.L    INIT-MFPDRV         Address of initialization routine.
          DC.L    0                   Address of SYSFAIL routine.
                                      0 = not used.
```

```
*  Other CMR parameters.
*
        DC.B    EXTCCBSZ        Number of extra pages in the CCB.  The CCB
                                may be any size from 1 to 256 pages (a
                                page is 256 bytes):  if this field is 0,
                                it will be 1 page, etc.
        DC.B    0,0,0           Reserved.
        DC.L    0,0,0           Reserved.

*
*  Revision info:
*
        DC.B    '102384'        Date of last assembly as MMDDYY.
        DC.B    ' '             Space indicates no patches to .LO file.
        DC.B    '4'             Major VERSAdos revision number.
*
*  Jump table for use by the generic TERMLIB.
*
        BRA.L   PUT_CHAR
        BRA.L   CK_TBE
        BRA.L   DDP_RESET
        BRA.L   SETUP
        BRA.L   CLOCK_RESET
        BRA.L   GET_STAT
        BRA.L   DDP_STOP
        BRA.L   DDP_UNSTOP
        BRA.L   DDP_BEG_BREAK
        BRA.L   DDP_END_BREAK
        PAGE
```

b. Define the Initialization Handling code. In some cases, no specific initialization may be required as a simple RTS would be sufficient at the INIT entry point.

INITIALIZATION ROUTINE

This routine is called by CMR when the channel is allocated. Its purpose is to set up things for command service and interrupt service. The major functions performed are:

1. Takes over additional vectors required by this driver.
2. Device-independent initialization in the call to TERM_INIT.
3. Determine if this is the driver for the MVME120.
4. Device-dependent initialization in a call guarded to DO_INIT to initialize the device.

```
*
*       This point was reached through a JSR from CMR, and will exit with an
        RTS.  All registers have been saved by CMR and may be used freely.
*       Entry is at interrupt level 0.  These functions are completed in
        supervisor mode.
*
```

```
*         Entry:  A5 = address of CCB.
*
*         Calls:  TERM_INIT in the generic terminal driver TERMLIB.
*
*         Exit:   CMR (via RTS) if all is OK.
*                 KILLER if exec won't allocate memory,
*                  or can't get background routine entry point.
*
          DS     0

*
* The MC68901 Multifunction Peripheral chip has 4 vectors that
* support the USART part of the chip.  Currently, there is no
* way that CMR can process 4 vectors for one channel.  This
* issue is handled here by reading the address stored away in
* the vector given CMR via the channel data block, and storing
* that address in the other 3 vectors that CMR has not taken over
* Actually, the driver only uses 3 of the four vectors. The vector
* number $6C is the one stored in the CCB.  The address from vector
* number $6C must be put into vector numbers $6B and $6A.  The vector
* number $69 is used to report transmit errors and is not supported by
* this driver.
*

INIT:
*
*  Take over the additional vectors.
*
          MOVEQ.L #0,D0              Clear working register.
          MOVE.B CCBVECT(A5),D0      Get the vector number from our CCB.
          MULU   #4,D0               Multiply by 4 to get the address.
          MOVE.L D0,A0               Copy to address register.
          MOVE.L (A0),-(A0)          Copy contents to address - 4.
          MOVE.L (A0),-(A0)          Copy contents to address - 4.
*
*  Perform the device-independent initialization by calling the
*  generic routine TERM_INIT in TERMLIB.
*
         JSR       TERM_INIT  Complete device-independent changes.
         IF   <NE>   THEN     Error occurred with TERM_INIT.
         IF   <PL>   THEN     The exec call failed -- no access to
*                             the scheduler's entry point.
              TR0$.KILLER
         ELSE
             RTS              The channel is down.
         ENDI
         ENDI
*
*  Read the high order byte of the Module Control Register to determine
*  if the driver is on the MVME120.  The high order byte of the
*  MCR is guaranteed to be $FF on the MVME120 board. Set the
*  proper driver code in our CCB.
*
```

```
        MOVE.W MCR,D0                   Read the MCR.
        CMP.B  #$FF,D0                  High order byte guaranteed $FF.
        MOVE.B #120,DRV_FLAG(A5)        Set drive flag for MVME120.
        BRA.S  INIT200                  Done for the 120.
*
*  Perform the minimum device-dependent initialization using
*  call guarded to the EXEC for a routine resident in background.
*
INIT200 LEA       D0_INIT(PC),A0        Device-dependent routine address.
        MOVE.L    #TOGUARD,D0           Call guarded to the EXEC.
        TRAP      #0
        RTS
```

c. Define the Command Handling jump table. In most cases it should only require slight modification to this table. If a special command handling routine is necessary, simply define the code and enter the offset in the command table.

```
*         COMMAND TABLE & JUMP TABLE
*
*         These commands are used by the command service routine to jump to
*         the appropriate routine to handle a command when not in
*         transparent mode.
*
*         Note that IOHALT (halt I/O) is not in the table -- it is checked
*         for separately.
*
          DS     0

CMD_TBL:  DC.W   IOWRIT           WRITE command.
          DC.W   IOOWIN           OUTPUT W/ INPUT command.
          DC.W   IOREAD           READ command.
          DC.W   IOTBRK           TRANSMIT BREAK command.
          DC.W   IOSTAT           REQUEST STATUS command.
          DC.W   IOCNFG           CONFIGURE command.
          DC.W   IOCHDC           CHANGE DEFAULT CONFIGURATION command.

NUM_CMDS  EQU    (*-CMD_TBL)/2    The number of commands in the table.

JMP_TBL:  DC.W   WRITE-*          Biases to the corresponding routines.
          DC.W   OUTWINP-*
          DC.W   READ-*
          DC.W   XMIT_BRK-*
          DC.W   REQ_STAT-*
          DC.W   CONFIGUR-*
          DC.W   CHNG_DEF-*

JMP_OFF   EQU    JMP_TBL-CMD_TBL  Offset between the two tables.
*                                 Having found the command in the first
*                                 table, this offset is used to get to
*                                 the corresponding value in the second.
```

d. Define the interrupt handling routine. This routine contains most of the device dependent code and will require device specific modifications or a totally different method than used in the example. Be sure to adhere to the entry and exit conventions.

```
PC=0000A59E  SR=2600=.S6.....  USP=0002E9E6  SSP=00000CA8  VBR=00000000  SFC=0
DFC=0
D0-7 0000FF00 FFFFFF00 0000328A 00000000 0001DCD2 0001DD0E 000054E4 00000040
A0-7 0002666C 0000A59E 00006420 0001DC88 0000A500 00026500 0000F300 00000CA8
 PC=00A59E    226D0022              MOVE.L  34(A5),A1
```

Upon entry at the interrupt handling routine the registers above reflect the state of the system. The register of interest to the device driver is A5. This register points to the CCB of the first device on the interrupt polling chain for the vector.

```
MVME120 1.1>  MD (A5) 20
026500    21 43 43 42 00 02 62 00  00 00 00 00 00 00 00 00  !CCB..b.........
026510    00 00 00 00 43 4D 46 50  6C FE 00 00 A5 00 00 00  ....CMFP!~..%...
```

STEP 5

Define the necessary substitution parameters which will be used with the DCB macro. It may be necessary to define a new macro for the type of device to be supported. The only requirement is to keep the device-independent portion of the macro intact. The specific device may then be defined as device-dependent. Use the files MACRO.DCB.SI and MACRO.DCBDISK.SI as examples.

```
*
*  MACRO.DCB.SI  -- DEFINES BASIC DCB AND CDB MACROS
*      See MACRO.DCB*.SI for specific DCB macros, i.e., DISK,TERM,GPIB, etc.
*
*  SECTION 0   DCB = Device Control Block:  contains info about device,
*                    including default configuration; used heavily
*                    by FMS, FHS, IOS.
*  SECTION 1   CDB = Channel Data Block:  used by IOI to allocate channels.
*


*  DIPDCB macro defines device-independent portion of a DCB.  Used by the
*  other device-dependent macros DSKDCB, CRTDCB, PRTDCB.
*
```

9

```
DIPDCB MACRO
 DC.L    *+\1                Address of next DCB in linked list.
 DC.L    \2                  ASCII identification for this DCB.
 DC.L    0                   Address of DCQ entry.
 DC.L    \3                  Name of task making the request.
 DC.L    \4                  Session of task making the request.
 DC.L    0                   Address of LUT.
 DC.L    \5                  Device attributes associated with DCB.
 DC.W    0                   Write/Read protect codes.
 DC.W    0                   'Device in use' flag.
 DC.L    0                   Write/Read counts.
 DC.B    \6                  Device flag (device code).
 DC.B    \7                  Device flag (device status).
 DC.L    \8                  Channel identification.
 DC.B    \9                  Device number associated with the channel.
 DC.B    0                   Task priority.
 DC.L    0                   Current record #.
 DS.B    IOSBLN              Storage area for the IOCB being processed.
 DC.L    0                   Logical address of IOCB in user's address space.
 DC.B    0                   Config. coordination flag (0 --> at defaults).
 DC.B    0                   Break count.
 DC.L    0                   Address of break service LUT.
 DC.L    0                   Break service address.   ----------------
 DC.L    0                   Event claimer -- task name
 DC.L    0                   Event claimer -- session number
 DC.L    \A                  Address of supervisor DCB or Session
*                            number if this is a supervisor DCB
 DC.L    0                   Supervisor/subordinate DCB open count
 DC.L    0,0,0,0             Device-independent/dependent buffer zone

 ENDM

*
*  The following listing shows the macro used to define a Channel Data Block
*  (CDB).  This is used by IOI to allocate channels; equates are in LV5.eq.
*
CDBLN EQU $2E    Length of the CDB data structure.

CDB MACRO
 SECTION 1          === CDB SECTION ===
 DC.L *+CDBLN       Pointer to next CDB in list.
 DC.W \1            Options for the ALLOCATE command.
 DC.L \2            Channel mnemonic.
 DC.B \3            Channel type.
 DC.B \4            Masked interrupt maximum instruction count.
 DC.L \5            Physical address of driver.
 DC.L \6            Supervisor channel's mnemonic (only if bit 3 of options set).
 DC.L \7            Physical address of device in memory mapped I/O space.
 DC.W \8            Number of bytes device occupied in memory mapped I/O space.
 DC.B \9            Vector number.
 DC.B \A            Polling priority.
 DC.B \B            Software priority.
 DC.B \C            Segment count.  (Number of polling entries)
```

```
DC.W \D         Polling byte offset. -- [#1] --
DC.B \E         Polling mask.
DC.B \F         Polling test value.
DC.W \G         Offset from physical device address for reset.
DC.B \H         Value for reset.
DC.B 0          Reserved.
DC.W \I         Polling byte offset. -- [#2] --
DC.B \J         Polling mask.
DC.B \K         Polling test value.
DC.W \L         Offset from physical device address for reset.
DC.B \M         Value for reset.
DC.B 0          Reserved.

ENDM
```

STEP 6

Define the CDB Macro. All channel definitions for devices are identical. Therefore, only enter the specific parameters such as channel name, I/O address, and vector number.

```
        IOC.MFPDRV.AG
*
*  Included equate files:
*      &.IOE.EQ
*      &.NIO.EQ
*
         NOLIST
         INCLUDE   &.IOE.EQ
         INCLUDE   &.NIO.EQ
         LIST

*  Included device specific macros:
*      MACRO.DCB.SI
*      MACRO.DCBTERM.SI
*
         NOLIST
         INCLUDE   MACRO.DCB.SI
         INCLUDE   MACRO.DCBTERM.SI
         LIST
```

```
*  Assign values from SYSGEN parameters
*
DVCODE   SET \&CRTDV     Define starting device code
LTDA$01  EQU \LTDA$01    Define MFP #1 address
MFP      EQU \MFPDRV     Define the physical address of the driver

CNAME    SET     'CMFP' channel name mnemonic


   SET UP DCBs & CDBs FOR LOCAL TERMINALS  -  MFPs


* Define TCP$ATW once for all terminals
TCP$ATW  SET  \TCP$HCPY+\TCP$XCTL<<1+\TCP$BITS<<2+\TCP$STPB<<3+\TCP$USEP<<4
TCP$ATW  SET  TCP$ATW+\TCP$PRTY<<5+\TCP$ECHO<<6+\TCP$TAHD<<7+\TCP$TFUL<<8
TCP$ATW  SET   TCP$ATW+\TCP$PNUL<<9+\TCP$MODM<<10+\TCP$OFFH<<11

*****  MFP port #1  *****
*
*   MFP port #1
*
*--  Data Control Block  --*


   SET UP DCBs FOR LOCAL TERMINALS

 CRTDCB DVCODE,IOSID,IOSESS,$133,35,1,CNAME,0,$0FFF,$7FF3,TCP$ATW,\TCP$REC,
& \TCP$RSZ,\TCP$WTO,\TCP$RTO,\TCP$XOF,\TCP$XON,\TCP$BRC,\TCP$DOP,\TCP$RLN,
& \TCP$CLC,\TCP$RTV,\TCP$EOL,\TCP$BRT,\TCP$NLS,\TCP$TRC,\TCP$TTP

     Channel Data Block
 CDB $0000,CNAME,XTMFP0,254,MFP,0,LTDA$01,1,$6C,6,$30,
&    0,0,0,0,0,0,0,0,0,0,0

DVCODE SET DVCODE+1

 XYZDCB DVCODE,IOSID,IOSESS,$FFF,55,1,CNAME,0,$0FFF,$7FF3,XYZ$ATW,\XYZ$REC,
& \XYZ$RSZ,\XYZ$WTO,\XYZ$RTO,\XYZ$XOF,\XYZ$XON,\XYZ$BRC,\XYZ$DOP,\XYZ$RLN,
```

9

Example:

The above example of a DCB macro call is for a newly invented device called the "XYZ". Using the CRTDCB macro, a new macro structure called XYZDCB has been constructed. This macro should then go into MACRO.DCBXYZ.SI to be included when IOC.XYZDRV.AG is assembled. Most cases where a need for a new file or data structure is called for, existing files may be used as samples.

```
Example:
*
*  Macro used to define DCB for device XYZ:
*
XYZDCB MACRO
 SECTION 0        === DCB SECTION ===
 DIPDCB CDCBLN,\1,\2,\3,\4,\5,\6,\7,\8,0
 DC.B    0,0,0,0   Space for status fields.
 DC.W    \9        Attributes mask.       $0FFF
 DC.W    \A        Parameters mask.       $7FF3
 DC.W    \B        Attributes word.       XYZ$ATW
 DC.W    \C        # of characters/line.  XYZ$REC
 DC.L    \D        # of lines/page.       XYZ$RSZ
 DC.L    \E        Write time-out.        XYZ$WTO
 DC.L    \F        Read time-out.         XYZ$RTO
 DC.B    \G        XOFF char.             XYZ$XOF
 DC.B    \H        XON char.              XYZ$XON
 DC.B    \I        BREAK EQUIVALENT char.XYZ$BRC
 DC.B    \J        DISCARD OUTPUT char.   XYZ$DOP
 DC.B    \K        REPRINT LINE char.     XYZ$RLN
 DC.B    0,0,0,0,0,0,0,0,0,0  Internal use only -- do not use.
 DC.B    0,0,0,0,0,0 Reserved.

 ENDM
```

STEP 7

Build SYSGEN files to include the driver into the system during the SYSGEN procedure. Step 8 may be added to the <system>.CNFGDRVR.CI file. Essentially, these commands set parameters which will be used in the file <system>.IFDRVR.CI to include the specific commands necessary to link in the required driver.

STEP 8

The files that need to be built are described here.

New files:	Modify files:
XXXXDRV.AG	<system>.CNFGDRVR.CI
IOC.XXXXDRV.AG	<system>.IFDRVR.CI

Example:

```
*
*       VME120.CNFGDRVR.CI
*
* Configuration file for device drivers
```

9

```
------------------------------------------------------------------------
* This file sets up the flags used by the "VME120.IFDRVR.CI" file to
* conditionally include device drivers.
*
* The user should only have to modify this file to include/exclude drivers.
* To add more boards/devices, the user may have to increase the SYSGEN
* command T option for more symbols in the "&.SYSGEN.CF" file.
*
* To modify specific items of a driver, edit the corresponding driver file,
* "&.xxxxDRV.CI" except as noted where one driver handles multiple boards.
* Board/system dependencies are included from the &.VERSADOS.CD file.
* VME120.SYSTEM.CI   Add processor board/system dependencies,
*                    including local terminals and printers.
* &.CNFGTASK.CI      Add O/S task configuration for ROM/RAM.

------------------------------------------------------------------------
NVME      = 1       Number of terminals on VME120 serial ports (MFP); max= 1

------------------------------------------------------------------------
NRWIN     = 0       Number of RWIN  Winchester controller boards

IFGT        \NORWIN
   CONTWIN1  = "2"          1st RWIN1 is controller 2
   NHRWIN$1  = 1            # of hard   disk drives on 1st M68RWIN1; max= 2
   NFRWIN$1  = 2            # of floppy disk drives on 1st M68RWIN1; max= 2
   RWIN0$1   = "'H5WIN15'"  Type of hard disk on 1st M68RWIN1, drive 0
   RWIN1$1   = "'H5WIN15'"  Type of hard disk on 1st M68RWIN1, drive 1
   RWIN2$1   = "'F5DDDSI'"  Type of floppy disk on 1st M68RWIN1, drive 2
   RWIN3$1   = "'F5DDDSI'"  Type of floppy disk on 1st M68RWIN1, drive 3

   CONTWIN2  = "5"          2nd RWIN1 is controller 5
   NHRWIN$2  = 0            # of hard disk drives on 2nd M68RWIN1; max= 2
   NFRWIN$2  = 0            # of floppy disk drives on 2nd M68RWIN1; max= 2
   RWIN0$2   = "'H5WIN15'"  Type of 1st hard disk on 2nd M68RWIN1, drive 0
   RWIN1$2   = "'H5WIN15'"  Type of 2nd hard disk on 2nd M68RWIN1, drive 1
   RWIN2$2   = "'F5DDDSI'"  Type of 1st floppy disk on 2nd M68RWIN1, drive 2
   RWIN3$2   = "'F5DDDSI'"  Type of 2nd floppy disk on 2nd M68RWIN1, drive 3
   *
   *  NOTE:  5-1/4" and 8" floppies cannot be mixed.  Pick one or the other.
   *
ENDC

------------------------------------------------------------------------
NVME050   = 1       Number of VME050 System Controllers
IFGT        \NVME050
   NT050$1   = 2    Number of terminals on the MVME050 board; max= 2
   NP050$1   = 1    Number of printers  on the MVME050 board; max= 1
ENDC

------------------------------------------------------------------------
NVME300   = 0       Number of MVME300  IEEE 488 GPIB controller boards
```

9

```
---------------------------------------------------------------------------
NVME315   = 1       Number of MVME315  Winchester/floppy disk controller
                    boards
IFGT        \NVME315
   CONT315   = "1"          MVME315 is controller 1
   NH315$1   = 1            Number of hard disk drives on 1st MVME315; max=2
   NF3158$1  = 2            Number  of  8"  floppy disk drives on 1st MVME315;
                            max=4
   NF3155$1  = 2            Number   of  5-1/4"  floppy  disk  drives  on  1st
                            MVME315; max=4

   M3150$1   = "'H5WIN15'"  Type of 1st hard disk on 1st MVME315 board
   M3151$1   = "'H5WIN15'"  Type of 2nd hard disk on 1st MVME315 board
   M31584$1  = "'F8SDDSM'"  Type of 1st floppy disk on 1st MVME315 board
   M31585$1  = "'F8SDDSM'"  Type of 2nd floppy disk on 1st MVME315 board
   M31556$1  = "'F5DDDSI'"  Type of 3rd floppy disk on 1st MVME315 board
   M31557$1  = "'F5DDDSI'"  Type of 4th floppy disk on 1st MVME315 board
ENDC

---------------------------------------------------------------------------
NVME316   = 1       Number of MVME316  VMEbus to I/O Channel interface boards;
*                   max= 1

---------------------------------------------------------------------------
NVME320   = 1       Number of MVME320  Winchester/floppy controller boards
IFGT        \NVME320
   CONT320   = "0"          MVME320 is controller 0
   NH320$1   = 1    Number of hard disk drives on controller; max= 2
   NF3205$1  = 1    Number of 5-1/4" floppy disk drives on MVME320; max= 1
   NF3208$1  = 0    Number of of 8" floppy disk drives on MVME320; max= 1

   M3200$1   = "'H5WIN15'"  Type of 1st hard disk on 1st MVME320 board
   M3201$1   = "'H5WIN15'"  Type of 2nd hard disk on 1st MVME320 board
   M32052$1  = "'F5DDDSI'"  Type of 1st floppy disk on 1st MVME320 board
   M32083$1  = "'F8SDSSI'"  Type of 2nd floppy disk on 1st MVME320 board
ENDC

---------------------------------------------------------------------------
NVME400  = 0          Number of MVME400  dual 7201 serial port boards
IFGT  \NVME400
   NU400$1   = 0      Number of ports/users on MVME400 bd. #1; max= 2/bd.
   NU400$2   = 0      Number of ports/users on MVME400 bd. #2; max= 2/bd.
ENDC
```

9

```
------------------------------------------------------------------------
NVME410  = 0          Number of MVME410  dual 16-bit parallel port boards
IFGT  \NVME410
   NU410$1    = 0     Number of printers in use on MVME410 board #1:  max= 2
   NU410$2    = 0     Number of printers in use on MVME410 board #2:  max= 2
ENDC

------------------------------------------------------------------------
NVME4208  = 0         Number of MVME420  SASI 8" interface controller boards
*
IFGT         \NVME4208
   CONT4208   = "4"          MVME420 (SASI 8) is controller 4
   NH4208$1   = 0     Number of hard disk drives on MVME420 (SASI); max= 2
   NF4208$1   = 0     Number of floppy disk drives on MVME420 (SASI); max= 4
   M42080$1   = "'H8WIN10'"  Type of hard disk on 1st MVME420, drive 0
   M42081$1   = "'H8WIN10'"  Type of hard  disk on 1st MVME420, drive 1
   M42082$1   = "'F8DDDSI'"  Type of floppy disk on 1st MVME420, drive 2
   M42083$1   = "'F8DDDSI'"  Type of floppy disk on 1st MVME420, drive 3
   *
   *     NOTE:  Total disks for SASI= 4, i.e. NH4208$1+NF4208$1 <= 4.
   *
ENDC

------------------------------------------------------------------------
NVME435    = 0       Number of MVME435 mag tape controller boards
*                          (max= 2)
IFGT         \NVME435
   N435$1     = 0    Number of tape drives on first  MVME435  board
   N435$2     = 0    Number of tape drives on second MVME435  board
ENDC
------------------------------------------------------------------------
NVME600    = 0       Number of MVME600  analog input  controller boards

------------------------------------------------------------------------
NVME605    = 0       Number of MVME605  analog output controller boards
IFGT         \NVME605
   NU605      = 0            Number of users (total) for the MVME605 boards
ENDC

------------------------------------------------------------------------
NVME610    = 0       Number of MVME610 AC input  controller boards

------------------------------------------------------------------------
NVME615    = 0       Number of MVME615/616 AC  output controller boards
IFGT         \NVME615
   NU615      = 0            Number of users (total) for the MVME615 boards
ENDC
```

```
-------------------------------------------------------------------------------
NVME620   = 0      Number of MVME620  DC input controller boards

-------------------------------------------------------------------------------
NVME625   = 0      Number of MVME625 DC output controller boards
IFGT        \NVME625
   NU625     = 0            Number of users (total) for the MVME625 boards
ENDC

-------------------------------------------------------------------------------
NRAD      = 0      Number of RAD Remote A/D boards
IFGT        \NRAD
   NURAD     = 0      Number of RAD users
ENDC

-------------------------------------------------------------------------------
NRIO      = 0         Number of RIO Remote I/O boards
IFGT        \NRIO
   NRIOINT   = 0      Number of interrupt levels per I/O module
ENDC

-------------------------------------------------------------------------------

        'XXXX'  is  the  acronym  of the device for which the driver is being
        written. VERSAdos conventions have been loosely defined as follows:

        Use the chip name or number if driver is for a specific chip (such as
        MFP).   Use  the name of the board if the drive is for a specific VME
        board such as M400DRV for the driver support of the MVME400 board.
-------------------------------------------------------------------------------
        Example:
-------------------------------------------------------------------------------

NVMEXYZ   = 1         Number of MVMEXYZ  dual 9999 serial port boards
IFGT  \NVMEXYZ
   NUXYZ$1   = 2      Number of ports/users on VMEXYZ bd. #1; max= 2/bd.
   NUXYZ$2   = 0      Number of ports/users on VMEXYZ bd. #2; max= 2/bd.
ENDC
```

9

STEP 9

Add the following commands to the file <system>.IFDRVR.CI.

```
IFNE \NVMEXYZ
    INCLUDE &.XYZDRV.CI
ENDZ
```

These commands allow SYSGEN to process the INCLUDE file XYZDRV.CI only if the appropriate switch \NVMEXYZ is set.

STEP 10

Build the XYZDRV.CI file.

XYZDRV=*	Defines the position within the BOOT file where this file resides.
SUBS &.MPSCDRV.LG	Allows any parameter substitution.
LINK &.MPSCDRV.LG	Links the driver.
PROCESS &.MPSCDRV.LO	Directs SYSGEN to place the driver into the BOOT file.
END &.MPSCDRV.LO	Indicates the end of processing for this module.

STEP 11

Build the XYZDRV.LG file. This file is used to link the driver .RO information and .LO file for inclusion into the target BOOT file. Use any existing <DRVNAME>.LG file as an example.

STEP 12

EQUATE file modifications. An EQUATE file of interest to device driver writers and which may require additions (never deletions) are located in system account :9995. The following list of EQUATEs should be examined to determine which files could be used by the new driver. Some devices may require the construction of a new equate files to contain in one place all the EQUATE definitions necessary for that board or chip. Note the following EQUATE for MK68901.EQ

```
* Include these EQUATE and macro files
*
*         STR.EQ
*         TCB.EQ
*         CCB.EQ
*         IOE.EQ            Parameter block offsets, channel type, error codes
*         NIO.EQ            Event msg, DCB,LUT,DCQ equates
*         LV5.EQ
*         BAB.EQ            Background Activation Block description.
*         TERMCCB.EQ        Offsets and EQUATEs for CCBDDP for terminals.
*         TERMINAL.EQ       Equates for terminals.
*         MK68901.EQ        Device description.
*         UTILITY.MC        Macros.
*
          NOLIST
          INCLUDE 9995.&.STR.EQ
          NOLIST
          INCLUDE 9995.&.TCB.EQ
          NOLIST
          INCLUDE 9995.&.CCB.EQ
          NOLIST
          INCLUDE 9995.&.IOE.EQ
          NOLIST
          INCLUDE 9995.&.NIO.EQ
          NOLIST
          INCLUDE 9995.&.LV5.EQ
          NOLIST
          INCLUDE 9995.&.BAB.EQ
          NOLIST
          INCLUDE 9995.&.UTILITY.MC
          LIST
          INCLUDE 9995.&.TERMCCB.EQ
          INCLUDE 9995.&.TERMINAL.EQ
          INCLUDE 9999.MK68901.MK68901.EQ
          TTL       MK68901 DEVICE DESCRIPTION
          LIST
```

9

STEP 13

Once all the files have been defined and the code written, perform the SYSGEN. For convenience a file called STD.SYSGEN.CF has been included in the release and should have been copied to the user account containing all the SYSGEN files. Log-on the SYSGEN account specifying the proper catalog.

Example:

```
=USE SYS:9100.VME120<cr>
=STD.SYSGEN.CF<cr>
```

The SYSGEN will take approximately 35 minutes on a single-user system.

When the SYSGEN is complete and there are no errors, copy the <system>.VERSADOS.SY to an bootable floppy disk for testing on the target configuration. Or, if the user is already running on the target, boot directly from the user number used to complete the SYSGEN.

9.8 ADDING AN UNSUPPORTED DISK DRIVE

The following procedure describes the method used to add an unsupported disk drive to a system running VERSAdos. Standard VERSAdos configurations support Winchester disk drives of 15Mb and 40Mb in size. The following example illustrates the steps required to alter SYSGEN in order to add a 20Mb Winchester disk drive to MVME120-based system equipped with an MVME315 disk controller.

STEP 1

Create a new file named &.H5WIN20.SI as follows:

a. Copy the existing file &.H5WIN15.SI to &.H5WIN20.

b. Edit &.H5WIN20 and change all driver parameters according to the manufacturer's specifications for the 20Mb Winchester disk drive to be used.

9

STEP 2

Edit the file IOC.M315DRV.AG.

a. Add a conditional assembly section for the inclusion of the &.H5WIN20.SI file immediately following the corresponding section for the 15Mb Winchester disk drive, as follows:

```
IFC\M3150$1,'H5WIN20'
     INCLUDE &.H5WIN20.SI
SIZESET SET 1
ENDC
```

A simple method to complete this step is to duplicate corresponding sections for a 15Mb drive and change appropriate statements.

b. Repeat STEP 2a. for the second hard disk on the first MVME315 disk controller.

c. Repeat STEPS 2a. and 2b. for the second MVME315 disk controller.

STEP 3

Edit the file VME120.CNFGDRVE.CI. Change appropriate drive definitions to "H5WIN20", for example:

"M3150$1 = H5WIN15" is changed to "M3150$1 = H5WIN20"

If the 20Mb Winchester disk drive is to be used by all disk controllers supported by SYSGEN for the MVME120, similar steps must be taken for the MVME320 and M68RWIN1 disk controllers.

STEP 1 does not have to be repeated. The new file &.HWIN20.SI applies to all disk controllers that use the same disk drive (or same drive characteristics). If multiple drives are to be used that have the same nominal size but have different characteristics (e.g. number of cylinders and/or heads), then STEP 1 must be repeated in order to create a new "&.DRIVETYPE.SI" file for each type of disk drive.

STEPS 2 and 3 must be repeated for each disk controller. However, in STEP 2 edit:

IOC.M320DRV.AG for the MVME320 disk controller and,
IOC.RWINDRV.AG for the M68RWIN1 disk controller.

9.9 NOTES ON MVME120 AND MVME121 MEMORY ARCHITECTURE

9.9.1 Overview

The MVME120 and MVME121 microprocessor modules are designed for high-performance within a multiple processor architecture. In order to meet the performance requirements, these boards include an onboard cache, a large onboard memory array, and a new memory array/MMU architecture to enable the MMU translation cycle to execute concurrently with the decoding of the low order address bits. This section describes the impact of the last feature on the system software.

9.9.2 Theory of Operation

The onboard memory architecture of the MVME120 and MVME121 processor boards is shown in Figure 9-1. Note that the cache is on the logical address bus. If the instruction accessed by a program space read cycle is found within the cache, the cache controller places the instruction on the data bus (D15-D0) and the cycle is terminated. Otherwise, the MC68451 MMU translates the high order address bits (A23-A10) at the same time that the memory array decodes the low order bits (A9-A1). These low order bits are used as the row address in the two dimensional row address/column address memory array.

If the MMU cannot translate the high order address bits, it signals a bus error to the processor, and the cycle is terminated. Otherwise, the MMU presents the translated high order address bits to the memory array which uses them as the column address. If these high order bits of the physical address point within the onboard memory array, the memory controller places the data on bits D15-D0 of the data bus. If the memory cycle was a program space read, the cache controller updates one cell in the cache to reflect the current address/data combination.

9.9.3 Memory Addressing Problem

A potential problem arises within VERSAdos because the MMU bypasses the low 10 bits of the effective address. (All previous Motorola processor boards bypassed only the bottom 8 bits, and translated address bits A23-A8.) If either bit A9 or A8 of the logical address is not equal to the equivalent bit of the corresponding physical address, the memory array will use the logical address bits instead of the physical address bits, and access the wrong cell in the memory array.

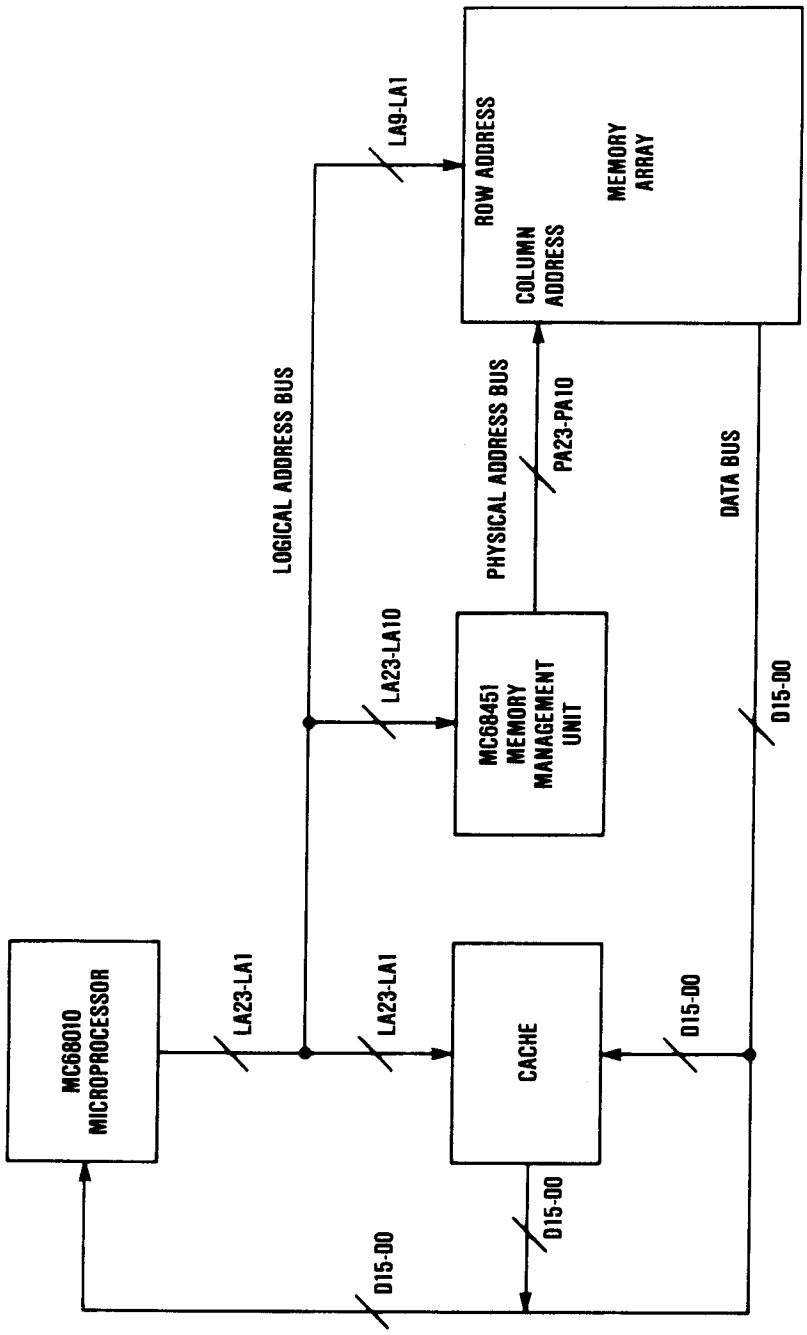


FIGURE 9-1. MVME120/MVME121 Memory Architecture

9

9.9.4 The VERSAdos Solution

The solution to this problem is to guarantee that bits A9 and A8 of the logical address are always equal to bits A9 and A8 of the physical address. VERSAdos 4.4 has modified the LINKER, LOADER, and SYSGEN command files to implement this solution. A "pagesize" option was added to the LINKER to describe the granularity of the memory array. For MVME120 and MVME121 systems, this variable should be set to 1024 to reflect the 1Kb pagesize of this family. In order to support the loading of object programs which were not linked with this pagesize option, the LOADER relocates all position-independent tasks so that the beginning logical address of all segments starts on 64Kb boundaries. (64Kb was chosen instead of 1Kb to support future extensions of this basic architecture.)

These two fixes guarantee that bits A9 and A8 of the beginning logical address of a segment are always equal to zero. Therefore, we need a corresponding fix to RMS68K's memory manager to ensure that bits A9 and A8 of the beginning physical address are also equal to zero. This fix was implemented in the following three stages:

a. Allocate operating system memory (unmapped by MMU) from a different partition than task memory (mapped by MMU).

b. Ensure that task memory is always allocated in pagesize chunks.

c. Ensure that the partition(s) reserved for task memory start on a pagesize boundary. (Bits A9 and A8 of the beginning physical address are equal to zero.)

These individual pieces of the fix are described in the following paragraphs.

9.9.4.1 Operating System vs. Task Memory. RMS68K kernel has access to all of memory (unprotected) and acknowledges all memory mapped with logical addresses equal to their corresponding physical addresses (unmapped). Therefore, the problem involving the MMU bypassing bits that should be translated does not affect RMS68K. A simple solution would have been to force RMS68K to abide by the same constraints as tasks; set memory on pagesize boundaries and in pagesize chunks. This would have forced all system data structures such as TCBs, ASQs, and other internal tables to consume at least 1Kb of memory each instead of the usual 512 or 256 bytes.

Instead of wasting so much memory, partition zero is reserved for system memory and partition one is reserved for task memory. Thus RMS68K can allocate its tables from partition zero in 256 byte chunks, but tasks must get memory in 1Kb chunks from partition one. This is implemented by setting the SYSGEN parameters for the default memory types for TCBs and ASQs (MEMTYPT and MEMTYPA) to zero, and the default memory type for user task and system task memory (MEMTYPU and MEMTYPS) to one. (The default type for system tables is always zero.)

Thus, a task that allows the GTSEG request for memory (get a segment of memory) to default to the partition type defined at SYSGEN time for task memory will experience little difficulty. However, a task which explicitly requests memory from the partition reserved for system memory, either by directly encoding the partition number or partition type in the GTSEG directive, must request the memory from a specific physical address, and ensure that the physical address is on a pagesize boundary.

9.9.4.2 Pagesize Chunks for Task Memory. RMS68K's memory manager guarantees that all requests for task memory are allocated in pagesize chunks by rounding the length of the segments requested via the GTSEG directive up to the next pagesize boundary. Pagesize is a SYSGEN parameter which is set to 1K for the MVME120 and MVME121.

9.9.4.3 Partition One Starting on Pagesize Boundary. In order to guarantee that allocatable memory within partition one (reserved for task memory) starts on a pagesize boundary, the partition must start on a pagesize boundary minus 256 bytes. The 256 bytes is used for the free memory list header node (FML header) describing the partition. Thus the SYSGEN parameter MEMEND2 (start of partition one) is set to $28000 - $100, and the MTENTRY macro describing partition one is written so that the FML header is placed in the BOTTOM of partition one (low memory). Thus, after the FML header is allocated at locations $27F00 - $27FFF, the "real" memory available for task allocation will start at location $28000. (Location $28000 was chosen as a good value for Motorola's default configurations. The user can change this value as described in the next section.)

9.9.5 Changing the Default Configurations

There are two ways that the user may change the default memory configurations for the MVME120 or MVME121 modules:

a. Modify the partition zero/partition one boundary.

b. Add additional memory partitions.

Partition one is configured to include addresses $27F00 to $400000. Therefore, to add more task memory, simply jumper the memory board to fit within this address range and plug it in. No SYSGEN is required.

However, if the boundary between partition zero and partition one must be changed, make sure that the boundary is on some pagesize boundary minus $100 bytes for the FML header. (One easy way to do this is to use a boundary of $XXX000 - $100.) To reconfigure the system, set the SYSGEN parameters MEMEND1 (end of partition zero), and MEMEND2 (start of partition one) to this value and run the SYSGEN utility.

To add memory partitions, use the MTENTRY macro in the INITDAT.AG module, and run SYSGEN. Describe all partitions reserved for system memory (TCBs, ASQs, and operating system tables) as type 0, and all memory for user or system task memory as type 1. (If more system memory is required, it would be easier to enlarge partition zero, rather than to add another partition of type 0.)

Describe all partitions of task memory in such a way as to reserve one 256 byte chunk for the FML header, leaving the rest of the memory as some integral number of pagesize (1K) chunks. The FML header can be placed at either the BOTTOM (low memory) or the TOP (high memory) of the partition as shown in the following examples:

a. FML header at BOTTOM of partition;

```
MTENTRY   RAM,$XXX000-$100,$YYY000,MTYP1,PART3,BOTTOM
```

b. FML header at TOP of partition;

```
MTENTRY   RAM,$XXX000,$YYY000+$100,MTYP1,PART4,TOP
```

The fields of the macro are as follows:

- MTENTRY - name of macro
- RAM or ROM - basic type af memory
- Starting address of partition

  (Note that in the BOTTOM example $100 bytes are reserved for the FML header at the beginning of the partition.)

- Ending address of partition

  (Note that in the TOP example above, $100 bytes are reserved for the FML header at the end of the partition.)

- Type of memory ($0 through $7)
- Partition number ($0 through $F)
- Placement of FML header (TOP or BOTTOM)

9

CHAPTER 10

BOOTING A SYSTEM

10.1 INTRODUCTION

The System Integrator or System Programmer concerned with altering the Kernel, operating tasks, or the number of drivers is occasionally confronted with a newly SYSGENed system that will not boot. This chapter describes the sequence of events that occur during the boot process.

This information may also be valuable for certain applications which might require the modification of this boot sequence. For example, during the RMS68K initialization, the user may require a new data structure table for a specific application with a pointer in SYSPAR. Knowing the sequence of the operations that the SYSINIT routine performs would be helpful when adding the code necessary to allocate a new data table.

When adding new user-written drivers, it is helpful to know exactly when a new driver is to be entered for the first time. This is explained in the VERSAdos Initialization paragraph 10.4.

The following defines the boot capabilities of various Motorola Debug Monitors. The capabilities are shown by indicating the controller number that corresponds to each module. The lack of a controller number indicates that a Debug Monitor does not provide boot support for a specific module.

VME CONTROLLERS

DEBUG MONITORS	MVME315	MVME320	MVME420 5-1/4 IN.	MVME420 8 IN.	M68RWIN1
MVME101 1.1	0				
MVME110 4.0	3	4	1	2	0
MVME115 1.1	1	0	3	4	2
MVME120 1.1	1	0	3 (NOTE)	4 (NOTE)	2 (NOTE)
TENBUG 2.1					0

NOTE: Current version of MVME120 Debug Monitor does not support boot capabilities from these controllers.

The following examples show a sampling of boot commands used by the Debug Monitors.

MVME110 4.0 > BO 0, 4	Boot from first hard disk on MVME320.
MVME120 1.1 > BO 6, 1	Boot from floppy disk on MVME315.
MVME120 1.1 > BO	Boot from default first hard disk on controller 0 (MVME320).

10.2 FIRMWARE BOOT SEQUENCE

This paragraph describes the sequence of events which the operating system executes from the time the BO command is entered until the Initial Program Loader (IPL) begins to execute.

MVME120 1.1> BO

This command initiates the boot sequence.

a. The MVME120 Debug Monitor resets controller 0 and drive 0. The HE command specifies which board is controller 0. In this example, the MVME320 is designated as controller 0.

b. The MVME120 Debug Monitor reads sector 0. The following character strings must be at offset $F8:

"EXORMACS" or "MOTOROLA"

If either string is not present an error will occur.

The data retrieved from sector 0 by the Debug Monitor in order of byte offset value is:

Get the volume ID from offset $0:	4 bytes
Starting sector to read at offset $14:	2 bytes
Number of blocks to read at offset $18:	2 bytes
Memory address to start load at offset $1E:	2 bytes
Sector address of the media configuration parameters at offset $90:	4 bytes
Length of configuration area offset $94:	1 byte

10

The program load module is defined by the data at locations $14 and $18 of Sector 0. The program is read and transferred to its memory destination. The processor status register is set to supervisor mode and interrupt level 7. The Stack Pointer (SP) is loaded from locations $0-3 relative to the beginning of the destination memory . The Program Counter (PC) is loaded from location $4-7. The registers are set-up as defined below, and the program just loaded by the BO command now has control.

```
D0...DRIVE NUMBER
D1...IPC NUMBER
D2...DISK CONFIGURATION CODE
D3...FLAG FOR IPL; 'ME4U' = 'USE BUGS' DISK READ ROUTINE
D4...IPLX; WHERE X ARE BITS INDICATION TRAP 15 AND DISK SUPPORT
     (NOT DURING AUTOBOOT)
D5...
D6...
D7...
A0...ADDRESS OF DISK CONTROLLER BOARD
A1...ADDRESS OF PROGRAM JUST LOADED
A2...ADDRESS OF DISK CONFIGURATION DATA
A3...ADDRESS OF BUGS' DISK READ ROUTINE
A4...ADDRESS OF THE DEBUGGER ENTRY POINT
A5...START OF TEXT
A6...END OF TEXT+1 (WHERE THE NEXT CHARACTER WOULD GO)
A7...STACK OF PROGRAM JUST LOADED
SR...SUPERVISOR MODE AND LEVEL SEVEN
```

The example below shows the registers as they would appear after a BOOT and HALT.

```
MVME120 1.1 > BO .,;H

Booting from: SYS
 Into RAM at: $010000
Boot in progress ...
Boot complete

PC=0001EA8A SR=2704.S7..Z.. USP=FFFFFFFF SSP=0001F510 VBR=00000000 SFC=0 DFC=0
D0-7 00000060 00F070C6 0001EA00 00000655 00000004 00000000 00000000 000000BF
A0-7 00001000 0001DF00 000007B8 0001087E 00FFB000 0001DE0E 00F0056A 0001F510
 PC=01EA8A    4ED1                JMP     (A1)

MVME120 1.1 >
```

The dump of Sector 0 (following) shows the starting sector to read ($14) as $2D (note this is a longword (4 bytes)). The number of blocks to read ($18) is $23 (this is a word field (2 bytes)). The memory address to read the boot file into is $10000 (this is also a longword (4 bytes)).

Normally, the program load module defined by the data at offsets $14 and $18 is the IPL module. This module is booted first and will locate and boot the desired opeating system task or diagnostic.

Sometimes it may be useful to point Sector 0 directly to the task that is supposed to execute independently of the operating system (i.e., a stand-alone task or a diagnostic). This can be done by changing fields $14, $18, and $1E to the appropriate starting sector, length, and memory address, respectively.

(1) Use the DIR ;A command to determine where the target code resides on the disk.

(2) Add 1 to the starting sector in order to bypass the Loader Information Block (LIB).

(3) Subtract 1 from the length to compensate for the previous add.

(4) Determine the location to load and execute.

(5) Using the DUMP utility in interactive mode, read in sector 0. Modify the appropriate offsets with the values determined in steps 1 through 4 above. Write out the altered sector 0.

```
   SN=$0          0
00   53 59 53 20 00 00 00 00  00 02 00 1D 00 00 00 27   SYS ...........'
10   00 00 00 00 00 00 00 2D  00 23 00 00 00 00 00 01   .......-.#......
20   00 00 00 00 29 91 56 45  52 53 41 44 4F 53 20 34   ....).VERSADOS 4
30   2E 34 20 50 41 53 53 32  2B 20 30 34 32 30 82 B1   .4 PASS2+ 0420..
40   0F 1E 2D 3C 4B 5A 69 78  87 96 A5 B4 C3 D2 E1 F0   ..-<KZix........
50   0F 1E 2D 3C 4B 5A 69 78  87 96 A5 B4 C3 D2 E1 F0   ..-<KZix........
60   F1 F2 F4 F8 F9 FA FC FE  FF 7F BF DF EF 6F AF CF   .............o..
70   4F 8F 0F 07 0B 0D 0E 06  0A 0C 04 08 04 02 01 00   O...............
80   00 00 00 00 00 00 00 00  00 00 00 00 00 1F 00 08   ................
90   00 00 00 01 01 00 00 00  00 00 00 00 00 00 00 00   ................
A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
B0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
C0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
D0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
E0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
F0   00 00 00 00 00 00 00 00  45 58 4F 52 4D 41 43 53   ........EXORMACS
```

10

c. The Debug Monitor will read using the previous parameters obtained from sector 0 for the starting sector, length, and starting memory address.

```
MVME120 1.1> B0

Booting from: SYS
 Into RAM at: $10000
Boot in progress ...
Boot complete
```

d. In the previous example, the Debug Monitor issues the messages:

```
Booting from: SYS
 Into RAM at: $10000
```

At this point, the load of the target code is complete. The task loaded at this time is IPL.SY. Control is transferred to the target code by the Debug Monitor which loads the stack from offset 0 and PC from offset $4.

e. When the Initial Program Loader (IPL) obtains control, it issues the messages:

```
Boot in progress
Boot complete
```

"Boot in progress" appears when the loading of VERSADOS.SY begins. "Boot complete" appears when transfer of control to the operating system is about to take place.

Examples of other Boot commands are:

VME120 1.1> BO 0,1	Boots VERSADOS.SY from controller 1, drive 0.
VME120 1.1> BO ,,TEST.SY	Boots TEST.SY from controller 0, drive 0.
VME120 1.1> BO 2,4,VME120.VERSADOS.SY	Boots from controller 4, drive 2; the file from :0.VME120.VERSADOS.SY.
VME120 1.1> BO 0,0,VERSADOS.SY;H	Boots from controller 0, drive 0; the file VERSADOS.SY then halts.

The sequence of execution for the IPL is:

a. Save registers by parsing command line received from the Debug Monitor save options and/or filename. Register A5 will contain a address of the string after the controller and drive number.

b. Read sector 0 using address of the Debug Monitor disk read routine received in register A3.

c. Get pointer to Secondary Directory Block (SDB) from sector 0 offset $C. By referring to the dump of sector 0, notice that the starting sector for the SDB is $27 (4-byte field).

d. Read SDB, one sector in length. Search this SDB for a match on the user number and catalog entered on the boot command line (or default user 0 and null catalog if none entered). If user:catalog entered does not exist, the message "file not found" is displayed. The starting sector of the Primary Directory Block (PDB) is obtained for the next step in the search for the target boot file.

In the example dump of an SDB, note the entry at offset $10. The first 2 bytes are the user number (0 in this case). The next 8 bytes identify the catalog (null or spaces in this case). The next 4 bytes are the starting sector of the first PDB. The final 2 bytes are reserved.

```
   SN=$27         39
00   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
10   00 00 20 20 20 20 20 20  20 20 00 00 00 28 00 00    ..          ...(..
20   00 00 50 52 49 56 20 20  20 20 00 00 06 59 00 00    ..PRIV      ...Y..
30   00 00 56 4D 45 31 32 30  20 20 00 00 06 74 00 00    ..VME120    ...t..
40   00 00 50 41 53 53 31 20  20 20 00 00 08 29 00 00    ..PASS1     ...)..
50   00 01 20 20 20 20 20 20  20 20 00 00 0A 09 00 00    ..          ......
60   00 01 50 52 49 56 20 20  20 20 00 00 10 43 00 00    ..PRIV      ...C..
70   00 01 56 4D 45 31 32 30  20 20 00 00 10 5E 00 00    ..VME120    ...^..
80   00 01 50 41 53 53 31 20  20 20 00 00 12 13 00 00    ..PASS1     ......
90   00 28 43 4F 4E 46 49 47  20 20 00 00 0E B5 00 00    .(CONFIG    ......
A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
B0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
C0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
D0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
E0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
F0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
```

e. Read PDB, four sectors in length. Locate a match on the filename entered on the boot command line or in VERSADOS.SY. If no file exists by the appropriate name the message "file not found" displays. A check is made to ensure the file found is a contiguous file.

The first 4 bytes of the PDB contain the pointer to the sector of the next PDB. The header portion (16 bytes) of the PDB is made up of 2 bytes for the user number, 8 bytes for CATALOG, and 2 bytes in reserve. Each entry is 50 bytes ($32). For a complete description of each field, refer to the EQUATE file :9995.&.FME.EQ. Each PDB is four sectors in length and accommodates 20 entries.

```
    SN=$28          40
00    00 00 06 55 00 00 20 20   20 20 20 20 20 20 00 00    ...U..          ..
10    49 50 4C 20 20 20 20 20   53 59 00 00 00 00 00 2C    IPL     SY.....,
20    00 00 00 23 00 00 00 00   00 00 00 00 00 00 00 00    ...#............
30    00 00 00 00 00 00 29 91   29 91 00 00 00 00 00 00    ......).).......
40    00 00 42 41 43 4B 55 50   20 20 4C 4F 00 00 00 00    ..BACKUP  LO....
50    00 50 00 00 00 4D 00 00   00 00 00 00 00 00 00 00    .P...M..........
60    00 00 00 00 00 00 00 00   29 91 29 91 00 00 00 00    ........).).....
70    00 00 00 00 43 4F 50 59   20 20 20 20 4C 4F 00 00    ....COPY    LO..
80    00 00 00 9E 00 00 00 27   00 00 00 00 00 00 00 00    .......'........
90    00 00 00 00 00 00 00 00   00 00 29 91 29 91 00 00    ..........).)...
A0    00 00 00 00 00 00 44 45   4C 20 20 20 20 20 4C 4F    ......DEL     LO
B0    00 00 00 00 00 C6 00 00   00 11 00 00 00 00 00 00    ................
C0    00 00 00 00 00 00 00 00   00 00 00 00 29 91 29 91    ............).).
D0    00 00 00 00 00 00 00 00   44 49 52 20 20 20 20 20    ........DIR
E0    4C 4F 00 00 00 00 00 D8   00 00 00 27 00 00 00 00    LO.........'....
F0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 29 91    ..............).

    SN=$29          41
00    29 91 00 00 00 00 00 00   00 00 44 4D 54 20 20 20    ).........DMT
10    20 20 4C 4F 00 00 00 00   01 00 00 00 00 0C 00 00      LO............
20    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
30    29 91 29 91 00 00 00 00   00 00 00 00 44 55 4D 50    ).).........DUMP
40    20 20 20 20 4C 4F 00 00   00 00 01 0D 00 00 00 28        LO.........(
50    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
60    00 00 29 91 29 91 00 00   00 00 00 00 00 00 45 20    ..).).........E
70    20 20 20 20 20 20 4C 4F   00 00 00 00 01 36 00 00          LO.....6..
80    00 6B 00 00 00 00 00 00   00 00 00 00 00 00 00 00    .k..............
90    00 00 00 00 29 91 29 91   00 00 00 00 00 00 00 00    ....).).........
A0    45 52 52 4F 52 4D 53 47   53 59 00 00 00 00 01 A6    ERRORMSGSY......
B0    00 00 02 48 00 00 00 AE   00 00 03 B8 00 00 02 04    ...H............
C0    00 00 00 04 01 04 29 91   29 91 00 00 00 00 00 00    ......).).......
D0    00 00 46 52 45 45 20 20   20 20 4C 4F 00 00 00 00    ..FREE    LO....
E0    02 55 00 00 00 0F 00 00   00 00 00 00 00 00 00 00    .U..............
F0    00 00 00 00 00 00 00 00   29 91 29 91 00 00 00 00    ........).).....

    SN=$2A          42
00    00 00 00 00 49 4E 49 54   20 20 20 20 4C 4F 00 00    ....INIT    LO..
10    00 00 02 65 00 00 00 4C   00 00 00 00 00 00 00 00    ...e...L........
20    00 00 00 00 00 00 00 00   00 00 29 91 29 91 00 00    ..........).)...
30    00 00 00 00 00 00 4C 49   53 54 20 20 20 20 4C 4F    ......LIST    LO
40    00 00 00 00 02 B2 00 00   00 1E 00 00 00 00 00 00    ................
50    00 00 00 00 00 00 00 00   00 00 00 00 29 91 29 91    ............).).
60    00 00 00 00 00 00 00 00   4D 54 20 20 20 20 20 20    ........MT
70    4C 4F 00 00 00 00 02 D1   00 00 00 1D 00 00 00 00    LO..............
80    00 00 00 00 00 00 00 00   00 00 00 00 00 00 29 91    ..............).
90    29 91 00 00 00 00 00 00   00 00 50 41 54 43 48 20    ).........PATCH
A0    20 20 4C 4F 00 00 00 00   02 EF 00 00 00 3D 00 00      LO.........=..
B0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
C0    29 91 29 91 00 00 00 00   00 00 00 00 52 45 50 41    ).).........REPA
D0    49 52 20 20 4C 4F 00 00   00 00 03 2D 00 00 00 73    IR  LO.....-...s
E0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
F0    00 00 29 91 29 91 00 00   00 00 00 00 00 00 53 50    ..).).........SP
```

```
    SN=$2B          43
00    4C 20 20 20 20 20 4C 4F   00 00 00 00 03 A1 00 00    L     LO........
10    00 2A 00 00 00 00 00 00   00 00 00 00 00 00 00 00    .*..............
20    00 00 00 00 29 91 29 91   00 00 00 00 00 00 00 00    ....).).........
30    53 50 4F 4F 4C 20 20 20   4C 4F 00 00 00 00 03 CC    SPOOL   LO......
40    00 00 00 12 00 00 00 00   00 00 00 00 00 00 00 00    ................
50    00 00 00 00 00 00 29 91   29 91 00 00 00 00 00 00    ......).).......
60    00 00 53 59 53 41 4E 41   4C 20 4C 4F 00 00 00 00    ..SYSANAL LO....
70    03 DF 00 00 00 70 00 00   00 00 00 00 00 00 00 00    .....p..........
80    00 00 00 00 00 00 00 00   29 91 29 91 00 00 00 00    ........).).....
90    00 00 00 00 56 41 4C 49   44 20 20 20 4C 4F 00 00    ....VALID   LO..
A0    00 00 04 50 00 00 00 04   00 00 00 00 00 00 00 00    ...P............
B0    00 00 00 00 00 00 00 00   00 00 29 91 29 91 00 00    ..........).)...
C0    00 00 00 00 00 00 56 45   52 53 41 44 4F 53 53 59    ......VERSADOSSY
D0    00 00 00 00 04 55 00 00   01 FF 00 00 00 00 00 00    .....U..........
E0    00 00 00 00 00 00 00 00   00 00 00 00 29 91 29 91    ............).).
F0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
```

Next, the IPL reads the Loader Information Block (LIB) of the target file (that is, the first sector of the file). In the example VERSADOS.SY, which is the last directory entry in the PDB, notice the starting sector is $455 and the length is $1FF sectors.

f. Read the LIB, first sector of the boot file only and determine the load address and length. Then determine if IPL must be relocated. IPL will have to be relocated if the target code must reside in the same place as IPL is currently loaded. IPL will then locate suitable memory outside of the boundary of the target boot file, and relocate itself.

```
    SN=$0           0
00    56 45 52 53 00 00 00 00   00 00 00 00 00 00 00 00    VERS............
10    00 00 00 00 00 00 00 01   B5 00 00 00 00 00 00 00    ................
20    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
30    00 00 00 00 56 45 52 30   00 00 10 00 00 01 B0 00    ....VER0........
40    00 00 00 00 00 00 00 00   00 00 00 00 FF FF FF FF    ................
50    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
60    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
70    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
80    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
90    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
A0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
B0    00 10 01 BF 00 00 FF FF   00 00 00 00 00 00 00 00    ................
C0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
D0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
E0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
F0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
```

g. IPL now outputs the message "Boot in progress" and reads the target file from the disk and starting sector +1 for length -1. When the load is complete, IPL outputs the message "Boot complete".

h. IPL transfers control to the target code by setting the stack (A7) to the contents of offset 0 of the recently loaded file, and by setting the PC to the contents of offset $4.

```
    SN=$1          1
00    00 00 00 00 00 01 B5 00   00 00 00 00 00 00 00 00    ................
10    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
20    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
30    00 00 00 00 56 45 52 30   00 00 10 00 00 01 B0 00    ....VER0........
40    00 00 00 00 00 00 00 00   00 00 00 00 FF FF FF FF    ................
50    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
60    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
70    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
80    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
90    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
A0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
B0    00 10 01 BF 00 00 FF FF   00 00 00 00 00 00 00 00    ................
C0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
D0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
E0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
F0    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
```

If the halt option was requested at boot time (;H), control will return to the MVME120 Debug Monitor.

The boot sequence continues with transfer of control to the operating system initializer "INIT.LO". The following steps describe the INIT sequence. This code executes at interrupt level 7.

a. Test for front panel. If one exists, display "BF". EXORmacs is the only system with a front panel displays. Determine if relocation of the operating system needs to take place. The parameter WHERLOAD (if not zero) contains the address to relocate the operating system. This only occurs with M68KVM01 systems with only 32Kb of RAM onboard.

   The MVME101 also has a status LED but at this point would not be changed. When VERSAdos is booted properly, a "d" will be displayed when the initializer is executed.

b. Set-up the vectors ($0-$400) using the VECTTBL.AB file assembled as RMSGEN time.

c. Build the Free Memory List describing the various memory partitions defined in INITDAT.AG

d. Set-up TCBHD and READYHD pointer in SYSPAR offsets $10 and $14.

e. Initialize the Memory Management Unit (MMU) (if one exists). JSR to routine in INITIO.AG, which was assembled at SYSGEN time.

f. Set up the following system tables from parameters in INITDAT.AG.

!ASN	Address Segment Number
!GST	Global Segment Table
!UST	User Semaphore Table
!IOV	I/O Vector Map
!PAT	Periodic Activation Table
!UDR	User deDined Directive
	Vector Use Table
	Trace Table
	Trap Assignment Table

g. Initialize any I/O device that may require initialization before dropping the interrupt mask. JSR to routine in INITIO.AG

h. Initialize the timer. JSR to routine in INITIO

i. Put "CO" in front panel (if one exists).

j. Jump to RMS68K dispatcher. Address of STARTRMS in INITDAT.

RMS68K execution sequence continues next. All further execution is at interrupt level 0, timer running.

The highest priority task on the ready list is placed into execution. This is normally IOI.LO -- the I/O Initializer task for the initialization of operating system tasks and device drivers.

10

10.3 IOI INITIALIZATION SEQUENCE

Figure 10-1 shows the structure of the IOI initialization sequence. The following text describes the major tasks that are performed by IOI.

a. IOI gets a data segment called DIOI.

b. IOI then follows the CDB linked list allocating channels defined by each CDB macro. If the channel is present, the appropriate device driver is entered at the INIT entry point with a register pointing to the newly created CCB. The allocate call is a TRAP #1 with D0=60, that is handled by the RMS68K Channel Management Routine (CMR). If the channel is not present as indicated and with CMR receiving a Bus Error when trying to access the memory address specified in the CDB macro, an error is returned and IOI continues to the next CDB.

c. IOI declares the IOSG as globally shareable, then grants shared access of this segment to FMS.

d. IOI then calculates the size of an FMSD segment needed by FMS, based on various SYSGEN parameters. IOI makes a GTSEG call for this segment, declares the segment shareable, grants shared access of FMSD to IOS, and transfers the segment to FMS.

e. IOI then starts the File Management System task FMS.

   At this point IOI relinquishes execution and allows FMS to run. FMS completes its initialization sequence and sets a bit in the System Value Table called the System Value Table System Task Configuration Flag (SVTSTCF). IOI is periodically given execution time while FMS is performing initialization, but IOI does not continue from the relinquish loop until FMS has set the required bit. (Refer to paragraph 10.4 for a description of the FMS initialization sequence.)

f. Once FMS has completed initialization, IOI continues to execute.

g. IOI grants shared access of the IOSG to FHS and starts FHS. IOI then relinquishes execution until FHS has completed and set its assigned bit in the SVTSTCF byte.

h. Once FHS has completed initialization, IOI continues execute.

i. IOI grants shared access of the IOSG to IOS and starts IOS. IOI then relinquishes execution until IOS has completed and set its assigned bit in the SVTSTCF byte.

j. IOS completes initialization and IOI continues to execute. At this point, all operating system tasks for which IOI is responsible have completed execution.

k. IOI now starts the Session Manager Task &EET and then terminates itself. This frees the memory segment containing the IOI code but leaves the IOSG segment which has been made globally shareable.

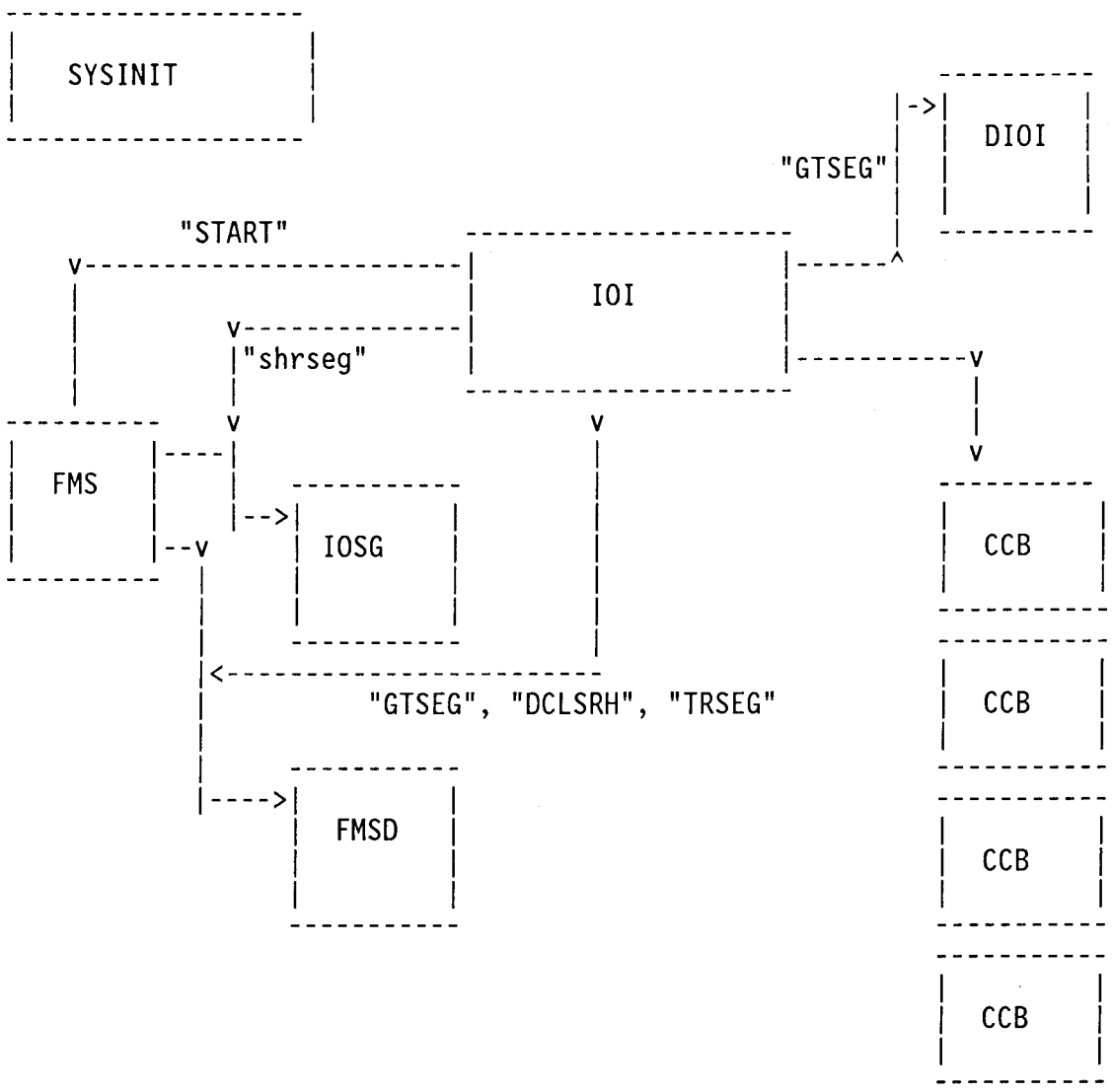


FIGURE 10-1. IOI Initialization Sequence Structure

10

10.4 FMS BOOT SEQUENCE FUNCTIONS

Figure 10-2 represents the "housekeeping" duties performed by FMS during the Boot sequence. The numbers in parentheses are the order of execution.

(1) IOI has issued a START for task FMS.

(2) FMS receives control with a register pointing to the FMSD segment that IOI has obtained and attached to FMS. FMS then divides this data segment into various functions areas, such as Volume Descriptor Table (VDT) space, File Control Block (FCB) space, File Assignment Table (FAT) space, Default Volume (DV) space, buffer space for File Access Blocks (FAB), and buffer space for Data Blocks (DB).

(3) FMS issues a GTASQ call to obtain an ASQ for message processing.

(4) FMS makes the GTTASKID calls to find out the ID of FHS. This ID will be used in all Queue Events.

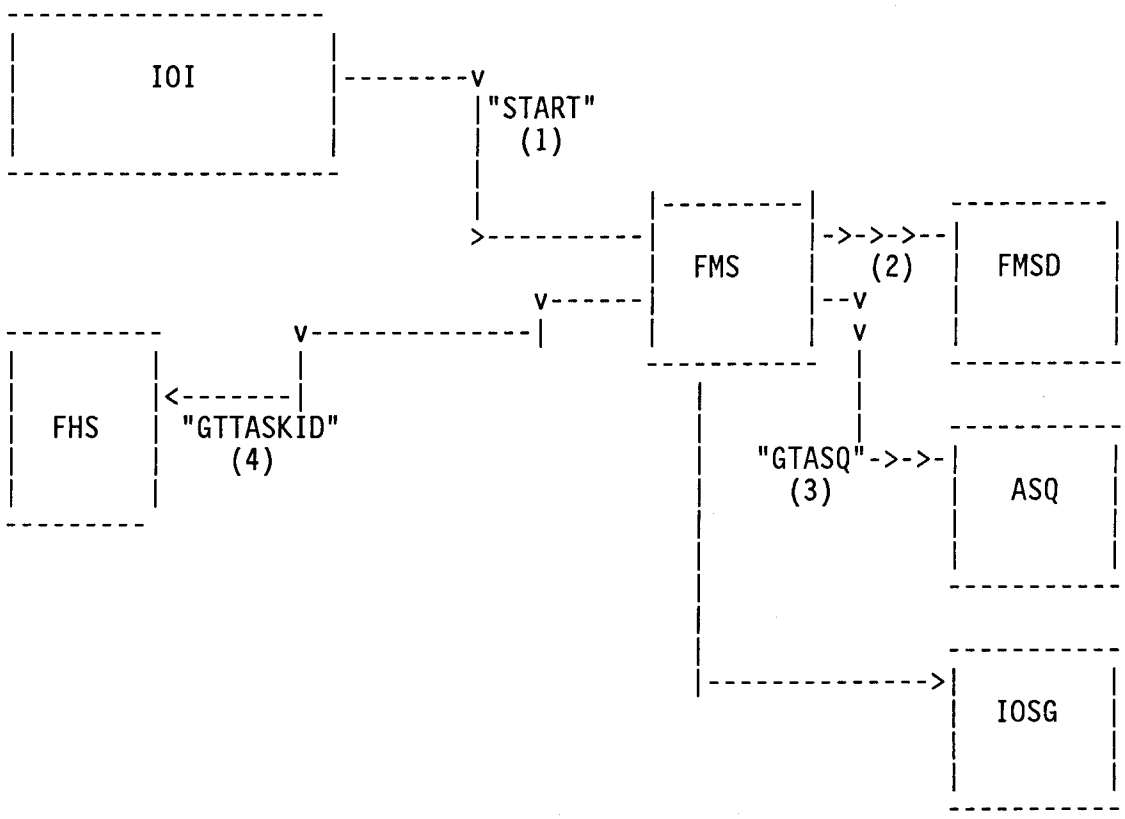


FIGURE 10-2. FMS Boot Sequence Functions

Note in Figure 10-2 that FMSD segment has previously been allocated by IOI and attached to FMS.

Once the housekeeping duties are complete, FMS will issue an GTEVNT call which is the idle state for FMS waiting for events to process.

10.4.1 Size of FMS Main Data Segment

FMS has a data segment that is allocated at initialization time. The segment size is governed by four SYSGEN parameters:

\NODIFFIL	Maximum number of different files that can be opened at a single time.
\TOTDSK	Number of disk devices on the system.
\NOFILES	Maximum number of files that can be opened at a single time (not necessarily different).
\NODEFVOL	Maximum number of default volumes allowed at a single time.

The data segment is partitioned into six functional areas:

Stack
VDT Storage
FCB Storage
FAT Storage
DV Storage
Buffer Storage

The size of each partition is computed as:

Stack	&12*(\NODIFFIL+1)+&100
VDT Storage	\TOTDSK*&374
FCB Storage	\NODIFFIL*&86
FAT Storage	\NOFILES*&144
DV Storage	\NODEFVOL11.6*&8
Buffer Storage	\TOTDSK*&1536

10

10.4.2 FMS Dynamic Segments

In addition to the data segment, FMS allocates dynamic segments for processing noncontiguous files. For each separate assignment to a noncontiguous file, FMS allocates a separate dynamic segment. The size of the segment will be equal to the size of a data block (1K by default, 2K if created by the Editor), plus the size of a FAB (0.25K default) plus 0.25K scratch pad area.

If all defaults are used, a 1.5K dynamic segment will be allocated. The segment will be made globally shareable, thus an entry in the GST will also be required. Note that it would be possible to have several such segments in memory simultaneously for the same file if there were several assignments open to that file at the same time.

10.4.3 I/O Common Data Segment

A data segment containing the Device Connection Queue (DCQ), Logical Unit Table (LUT), and Device Control Blocks (DCBs) is created at SYSGEN time. The size of each portion of the segment is computed as:

DESCRIPTION	SIZE (BYTES)
Header	$48 (&72)
DCQ Space	$200 (&512)
LUT Space	\NOTASKS*($10+$8*(\MACLU+1))
DCB Space	Sum of Individual DCBs (Described below) ($B8 (&184) per device for the worst case)

NOTE: \NOTASKS is a SYSGEN parameter specifying the maximum number of tasks in the system at a single time.

A DCB exists for each separate device on the system. The five types of devices allowed and the different sizes of each are:

TYPE	SIZE (BYTES)
Terminals	$B4 (&180)
Printers	$A4 (&164)
Disk	$A8 (&168)
Magnetic Tape	$B6 (&183)
GPIB	$B8 (&184)

An initialization data segment consisting of Channel Data Blocks (CDBs) is required to set up the I/O common data segment. The segment size depends on the number of channels that are in the system. It is common for one channel to control more than one device. The size of the CDB is:

CDB (All)	$2E (&46) per channel

10.4.4 Operating System Task Sizes

FMS	Code Segment	20.50Kb
	Data Segment	Variable (Refer to paragraph 10.3.4)
	ASQ	(\NOFILES+2)*$5E rounded to $100 bytes boundary.
	TCB	0.50Kb RAM, Not ROMable.
IOS (ROMable) (TRAP #2 Server)	Code Segment	6.25Kb
	Data Segment	0.50Kb
	ASQ	0.25Kb
	TCB	0.50Kb RAM $60 &96 bytes ROM
FHS (ROMable) (TRAP #3 Server)	Code Segment	5.00Kb
	Data Segment	0.50Kb
	ASQ	0.25Kb
	TCB	0.50Kb RAM $60 &96 bytes ROM
EET (Root task for session control)	Code Segment	15.25Kb
	Data Segment	0.75Kb
	2nd Data Segment	0.75Kb
	ASQ	\TOTTERM*2**$1A
	TCB	0.50Kb RAM, Not ROMable
SCT (Session Control Task)	Code Segment	Shared with EET
	Data Segment	0.75Kb
	2nd Data Segment	0.25Kb
	ASQ	0.25Kb
	TCB	0.50Kb RAM, Not ROMable
LDR (Task Loader)	Code Segment	5.25Kb
	Data Segment	0.75Kb
	ASQ	0.25Kb
	TCB	0.50Kb RAM, Not ROMable
IOI (ROMable) (I/O Initialization Task)	Code Segment	1.50Kb
	Data Segment	0.50Kb
	TCB	0.50Kb RAM $60 &96 bytes ROM
	CDB	Variable
	I/O Common	Variable

IOI only runs during initialization time. After completion, the code segments and data segments are given back to the system. A separate SCT task exists for each session that is created on the system. They all share the same code segment with EET.

10.4.5 RAM/ROM Sizes for Operating System Tasks

DESCRIPTION	SIZE	ROM	RAM	COMMENTS
FMS Code	20.50Kb	N	Y	
IOS Code	6.25Kb	Y	N	
FHS Code	5.00Kb	Y	N	
EET Code	15.25Kb	N	Y	
SCT Code	Shared	N	Y	
LDR Code	5.25Kb	N	Y	
IOI Code	1.50Kb	Y	N	ROMable
FMS Data	Variable	N	Y	Obtained at SYSGEN time
IOS Data	0.50Kb	N	Y	Dynamically obtained
FHS Data	0.50Kb	N	Y	Dynamically obtained
EET Data 1	0.75Kb	N	Y	Obtained at SYSGEN time
EET Data 2	0.75Kb	N	Y	Obtained at SYSGEN time
SCT Data 1	0.25Kb	N	Y	Dynamically obtained
SCT Data 2	0.25Kb	N	Y	Dynamically obtained
LDR Data	0.25Kb	N	Y	Dynamically obtained
IOI Data	0.50Kb	N	Y	Dynamically obtained
FMS ASQ	Variable	N	Y	Dynamically obtained
IOS ASQ	0.25Kb	N	Y	Dynamically obtained
FHS ASQ	0.25Kb	N	Y	Dynamically obtained
EET ASQ	Variable	N	Y	Dynamically obtained
SCT ASQ	0.25Kb	N	Y	Dynamically obtained
LDR ASQ	0.25Kb	N	Y	Dynamically obtained
I/O Common	Variable	Y	Y	Moved from ROM to RAM
CDB	Variable	Y	N	
FMS TCB	0.50Kb	N	Y	
IOS TCB	0.50Kb	Y	Y	&96 bytes in ROM
FHS TCB	0.50Kb	Y	Y	&96 bytes in ROM
EET TCB	0.50Kb	N	Y	
SCT TCB	0.50Kb	N	Y	Dynamically obtained
LDR TCB	0.50Kb	N	Y	

10.5 FHS BOOT SEQUENCE FUNCTIONS

Figure 10-3 represents the "housekeeping" duties performed by FHS during the Boot sequence. The numbers in parentheses are the order of execution.

(1) IOI has issued a "START" for task FHS.

(2) FHS gets control and issues a GTSEG call to obtain its data space, segment FHSD.

(3) FHS issues a GTASQ call to obtain an ASQ for message processing.

(4) FHS makes the GTTASKID call to find out the ID of IOS. This ID will be used in all Queue Events.

(5) FHS then issues the directive to establish itself as the TRAP #3 server.

Once these housekeeping duties are complete, FHS issues a GTEVNT directive which effectively is an idle state for FHS waiting for an event.

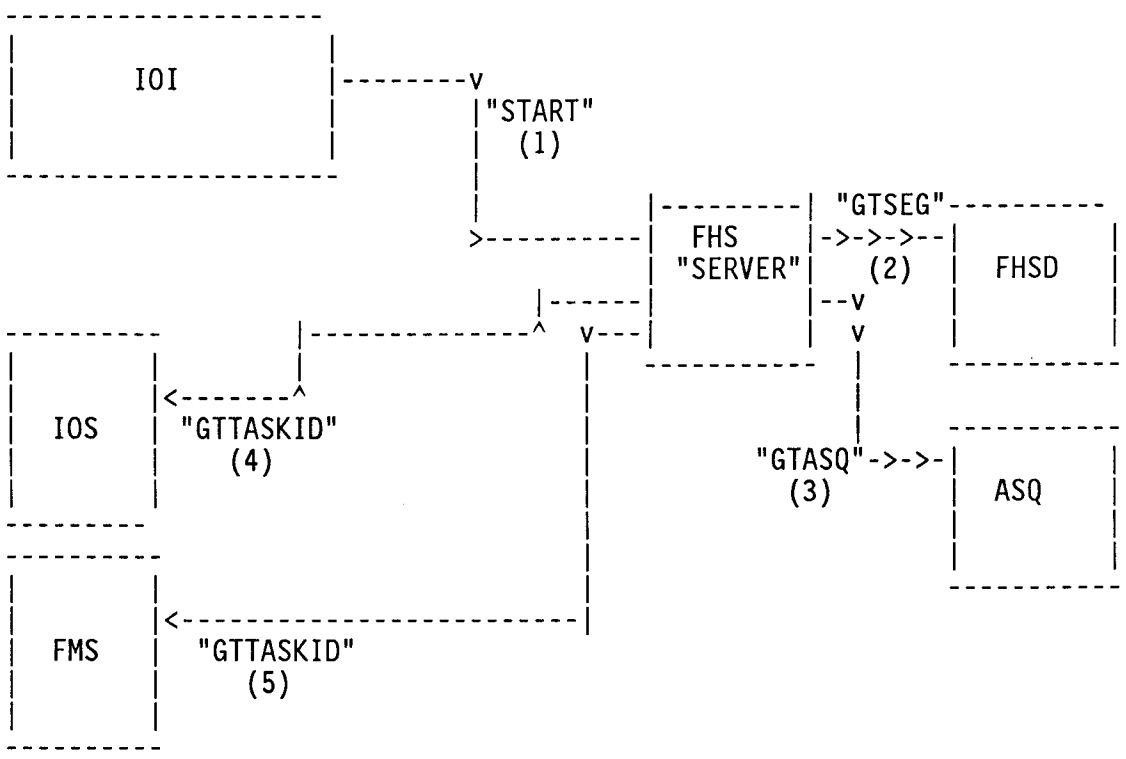


FIGURE 10-3. FHS Boot Sequence Functions

10.6 IOS BOOT SEQUENCE FUNCTIONS

The following is a list of the "housekeeping" duties performed by IOS (in order of execution) during the boot sequence.

```
IOI TRAP #1,D0=13       Start task IOS.
IOS TRAP #1,D0=1        Get a data segment called IOSD
IOS TRAP #1,D0=31       Get an ASQ segment
IOS TRAP #1,D0=10       Get task_id of task FHS.
IOS TRAP #1,D0=10       Get task_id of task FMS.
IOS TRAP #1,D0=51       Establish self as TRAP #2 server.
```

IOS walks the DCB list doing the following:

- Issue an attach channel (TRAP #1,D0=60) which causes CMR to make the appropriate CCB with the task and session number of IOS.

- TRAP #1,D0=60 to Initiate I/O to configure each device on this channel. IOS then waits for the completion event from the device driver and, if the device is a random access device (disk), IOS queues an event to FMS. IOS continues in this loop until the NULL device is reached.

During this initialization, loop IOS occasionally issues the GTEVNT directive resulting in no immediate action. This enables other real-time tasks on the ready list to execute. FMS, as a result of receiving events from IOS for each of the disk devices, assigns the device via a TRAP #3 to a Logical Unit (LU). The LU sequence follows the pattern of the first device being LU 1, next LU 2 ... etc. until all disk devices have been assigned.

During the initialization sequence performed by IOS, all three O/S real-time tasks will be executing. IOS, FHS, and FMS each will receive processor time as one or the other enters their GETEVNT state. This continues as all the devices in the DCB list are configured for default configuration, and the disk devices identified by FMS and Sector 0 are read and written back. The WRITE of Sector 0 occurs to determine if a device is write-protected. FMS configures each disk device with the MOUNT option bit to make all online, ready devices available for use. When this process is completed, IOS signals to IOI via a bit in the IOSG that all initialization has been completed. Each of the three real-time tasks sets a different bit such that the value is 7 when all initialization is complete. IOS is the last task to set its initialization-complete bit. IOI then breaks out of its RELINQUISH loop and starts the Session Manager Task &EET. Figure 10-4 is a diagram of the IOS Boot Sequence. Figure 10-5 depicts the Initialization Sequence Performed by IOS.

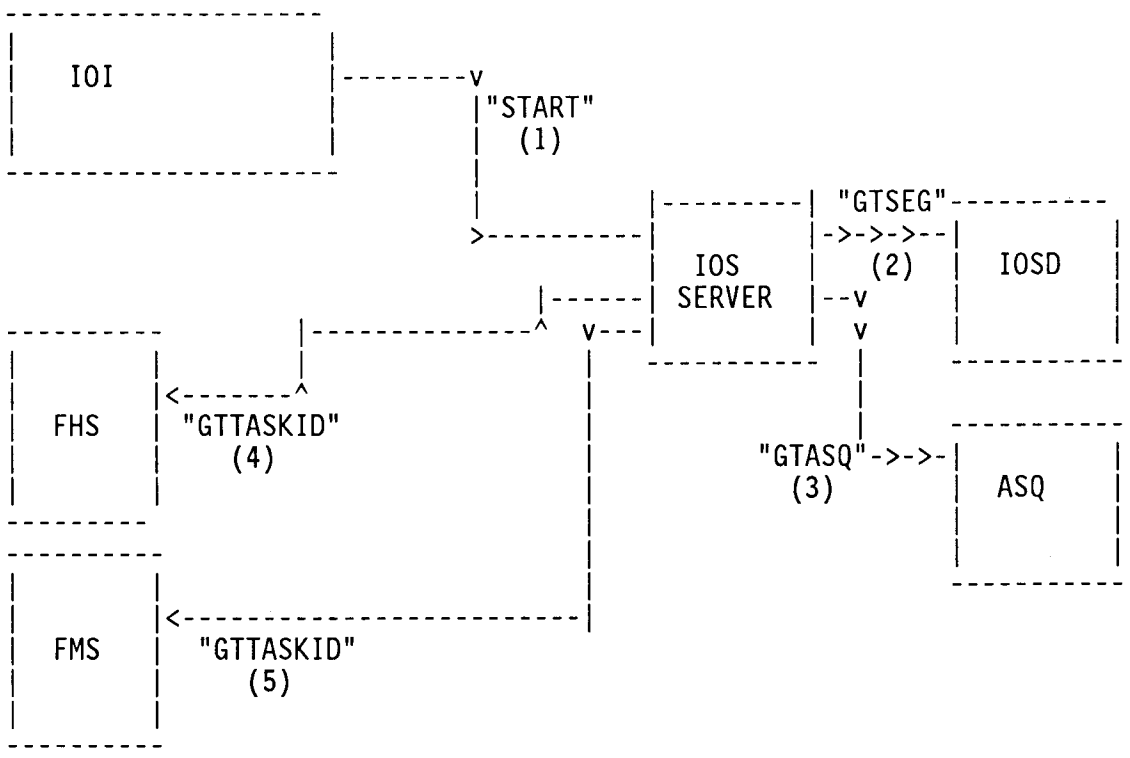


FIGURE 10-4. IOS Boot Sequence

10

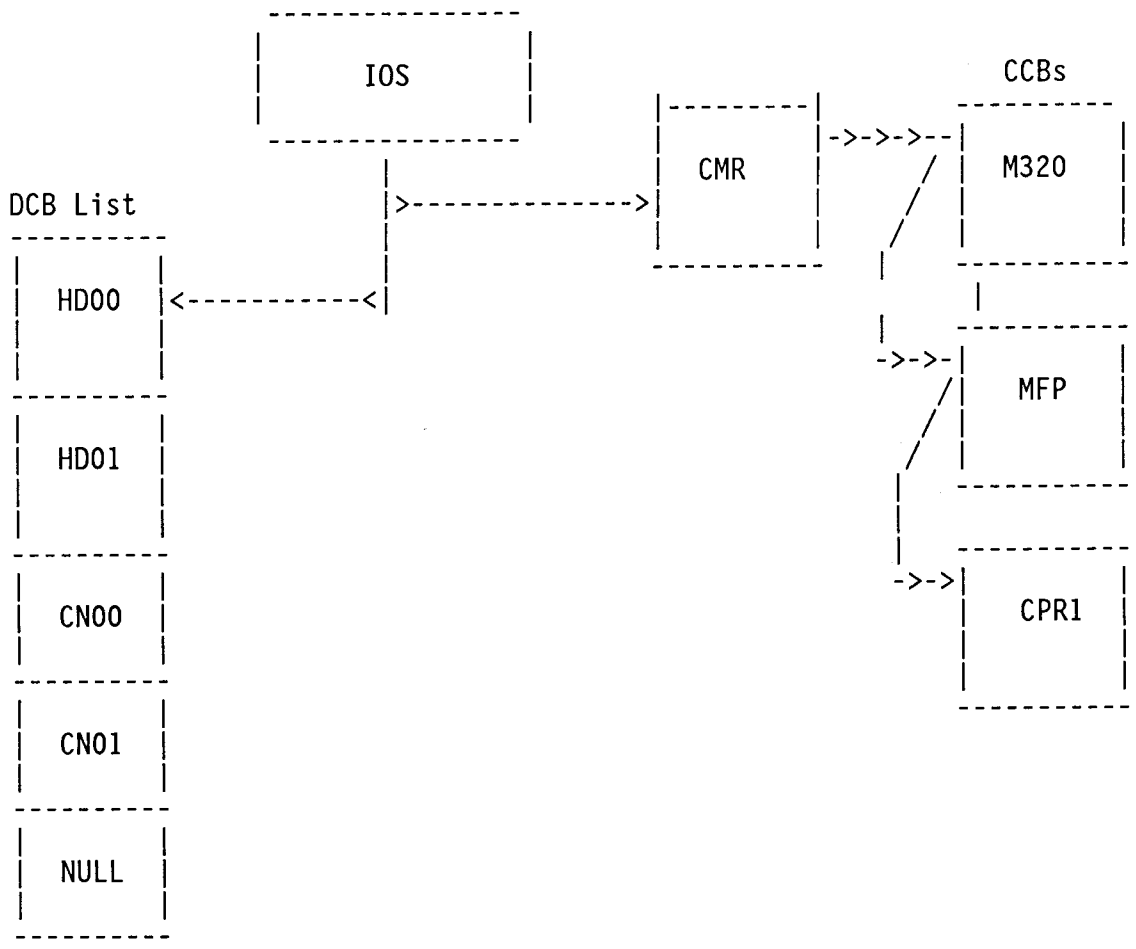


FIGURE 10-5. Initialization Sequence Performed by IOS

10.7 SESSION MANAGER EXECUTION SEQUENCE

The execution sequence performed by the Session Manager Task &EET is shown in Figure 10-6.

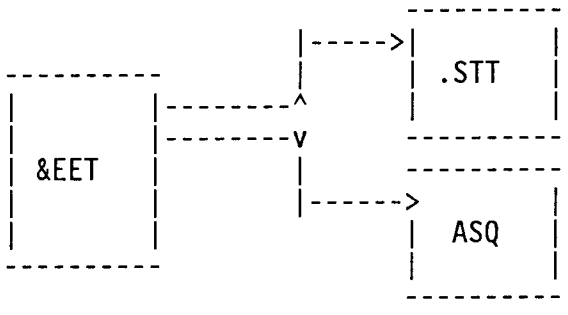


FIGURE 10-6. Session Manager Execution Sequence

The Entry/Exit Task (EET) is started by the I/O Initializer (IOI) after all other system tasks have executed their initialization sequences. The following description is for the major operating system functions performed by &EET (TRAPS #1, #2, #3, #4).

&EET TRAP #1,DO=7 Declare its code segment as shareable. This code segment will be used by all &SCT task as sessions are created.

TRAP #1,DO=7	Declare .STT segment as shareable.
TRAP #1,DO=31	Get an ASQ. Used to "break" events and notify when an &SCT terminates (session logoff).
TRAP #2 command $8001	Establish self as system break claimer. This tells IOS where to send break events if not claimed by task assigned to the terminal.
TRAP #1 DO=33	Set ASQ status to enable ASQ and enable ASR.
TRAP #1 DO=21	Delay for 1000 ms.
TRAP #1 DO=35	Queue an event to self. This is to simulate the break on a terminal used, if SYSGENed for autobreak.
TRAP #1 DO=34	Read the event.
TRAP #1 DO=11	Create a TCB for &SCT0001. Session Control Task (SCT) for session 0001.
TRAP #1 DO=5	Grant shared access to &SCT0001 of &EET code segment. All SCTs use the same code but different data segments.
TRAP #1 DO=1	Get a segment called .SCT.
TRAP #1 DO=1	Get a segment called .ARG.
TRAP #1 DO=6	MOVELL, move data from EET data segment to SCT data segment. This has the effect of priming SCT with certain useful data.
TRAP #1 DO=31	Get an ASQ for &SCT0001.
TRAP #1 DO=13	Start task &SCT.
TRAP #1 DO=37	Return from event.
TRAP #1 DO=33	Set ASQ to ASQ enabled
TRAP #1	DO=36 Wait for event

&SCT starts execution at this point. Figure 10-7 illustrates the &SCT Execution Sequence. From now on SCT controls all activity associated with this session. &EET is the monitor.

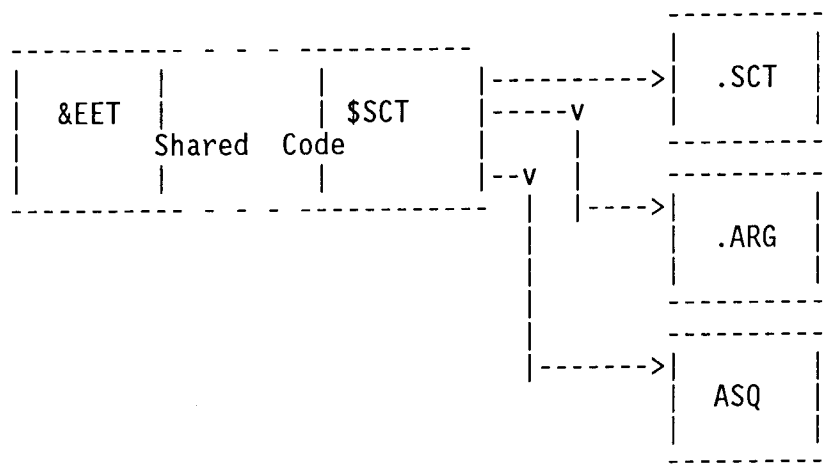


FIGURE 10-7. &SCT Execution Sequence

TRAP #1 DO=33 — Set ASQ to ASQ enabled and ASR enabled.

TRAP #3 — Assign #CN00 to LU 0. This will be the LU that SCT uses for all I/O to the terminal. As a result of the TRAP #3, the server FHS executes to handle the building of the LU entry.

FHS TRAP #1 DO=6 (MOVELL)	Move status to user parameter block.
FHS TRAP #1 DO=54 (AKRQST)	Acknowledge server request -- this allows &SCT to continue execution.
FHS TRAP #1 DO=38 (GTEVNT)	FHS idle state.

&SCT TRAP #3 — Assign #CN00 to LU 5 for read access. This becomes the standard input device and is passed to each task started under control of the SCT.

Same FHS sequence is executed as above. (MOVELL>AKRQST>GTEVNT)

&SCT TRAP #3 — Assign #CN00 to LU 6 for write access. This becomes the standard output device and is passed to each task started under control of the SCT.

Same FHS sequence is executed as previously described.

10

&SCT TRAP #2

Write command to CN00 to display the Logon message. As a result of the TRAP #2, the execution sequence that occurs is:

IOS TRAP #1 D0=29 (RQSTPA)	Start timer for I/O time-out.
IOS TRAP #1 D0=60 (CMR)	I/O command to device driver.
IOS TRAP #1 D0=53 (DERQST)	Enable ASQ for more server events.
IOS TRAP #1 D0=38 (GTEVNT)	IOS idle state.
IOS TRAP #1 D0=29 (RQSTPA)	Cancel timer event.
IOS TRAP #1 D0=54 (AKRQST)	Acknowledge server request. This allows user to continue.
IOS TRAP #1 D0=38 (GTEVNT)	IOS idle state.

&SCT TRAP #3

Assign the file ERRORMSG.SY. This assignment is an indication to the SCT whether the default volume is ready for use. The following execution sequence occurs as a result of this assignment:

FHS TRAP #1 D0=6 (MOVELL)	Moves status to user parameter block.
FHS TRAP #1 D0=35 (QEVNT)	Queue an event to FMS.
FHS TRAP #1 D0=53 (DERQST)	Allows more server events.
FHS TRAP #1 D0=38 (GTEVNT)	FHS idle state.
FMS TRAP #2 (READ)	Read of sector $27 (SDB).
IOS TRAP #1 D0=60 (CMR)	I/O command to device driver.
IOS TRAP #1 D0=53 (DERQST)	Allow server events.
IOS TRAP #1 D0=38 (GTEVNT)	IOS idle state.
IOS TRAP #1 D0=54 (AKRQST)	Acknowledge user task (FMS).
IOS TRAP #1 D0=38 (GTEVNT)	IOS idle state.
FMS TRAP #2 (READ)	Read of sector $28 (PDB)

	The same IOS sequence takes place as for the previous read.	
	FMS TRAP #1 DO=74 (GTDTIM)	Retrieve systems date and time.
	FMS TRAP #1 DO=6 (MOVELL)	Moves VOL:USER.CAT. FILE.EX to requestor's parameter block SYS:0.&. ERRORMSG.SY.
	FMS TRAP #1 DO=54 (AKRQST)	Acknowledge user.
	FMS TRAP #1 DO=38 (GTEVNT)	FMS idle state.
&SCT TRAP #3 (CLOSE)	Close assignment of ERRORMSG.SY.	
	FHS TRAP #1 DO=6 (MOVELL)	Move status to user parameter block.
	FHS TRAP #1 DO=35 (QEVNT)	Queue event to FMS.
	FHS TRAP #1 DO=53 (DERQST)	Allow move server events.
	FHS TRAP #1 DO=38 (GTEVNT)	FHS idle state.
	FMS TRAP #2 (WRITE)	Write sector $28 (PDB).
	IOS TRAP #1 DO=60 (CMR)	I/O command to device driver.
	IOS TRAP #1 DO=53 (DERQST)	Allow server events.
	IOS TRAP #1 DO=38 (GTEVNT)	IOS idle state.
	IOS TRAP #1 DO=54 (AKRQST)	I/O has completed, acknowledge user.
	IOS TRAP #1 DO=38 (GTEVNT)	IOS idle state.
	FMS TRAP #1 DO=54 (AKRQST)	FMS acknowledges user.
	FMS TRAP #1 DO=38 (GTEVNT)	FMS idle state
&SCT TRAP #3	Set default volume.	
	FHS TRAP #1 DO=6 (MOVELL)	Updates user parameter block status.
	FHS TRAP #1 DO=35 (QEVNT)	Queues event to FMS.

	FHS TRAP #1 DO=53 (DERQST)	Allow more server events.
	FHS TRAP #1 DO=38 (GTEVNT)	FHS idle state.
	FMS TRAP #1 DO=54 (AKRQST)	Acknowledge user.
	FMS TRAP #1 DO=38 (GTEVNT)	FMS idle state.
&SCT TRAP #3	Set default volume.	

&SCT TRAP #1 DO=6 (MOVELL)
Moves default volume from SCT data segment to EET data segment.

&SCT TRAP #2 (WRITE)	One character write to clear screen.	
	IOS TRAP #1 DO=29 (RQSTPA)	Start timer for I/O time-out.
	IOS TRAP #1 DO=60 (CMR)	I/O command to driver.
	IOS TRAP #1 DO=53 (DERQST)	Allows server events.
	IOS TRAP #1 DO=38 (GTEVNT)	IOS idle state.
	IOS TRAP #1 DO=29 (RQSTPA)	Cancel time event.
	IOS TRAP #1 DO=54 (AKRQST)	Acknowledge user.
	IOS TRAP #1 DO=38 (GTEVNT)	IOS idle state.

A

APPENDIX A

VERSAdos CONFIGURATION VALUES

A.1 DEVICE DRIVER CONFIGURATION TABLE

Appendix A.1 contains a complete list of all device driver information relevant to the Channel Data Block. The information provided under each column heading is explained as follows:

DRIVER — Lists the name assigned to each device driver. Chip-type drivers are named by the popular name of the chip and drivers for VMEmodules are defined by the number of the module preceded by an "M".

SYSTEM/BOARD — Probable locations of chip-type drivers.

CHANNEL NAME — ASCII ID of the channel located in the Channel Control Block.

CHANNEL TYPE — Unique value used by drivers to verify channel identity.

ADDRESS — Defines the MMIO address where a device resides or, in some cases, the label of the parameter defined in SIO.ADDRESS.CI.

VECTOR NUMBER/LEVEL — Defaults defined for the drivers.

SOFT PRI — Relative position of the driver on the CCB list for multiple channels on the save vector. Higher values correspond to closer proximity to the head of the list.

SIZE — Size of driver (in bytes).

CHIP NUMBER — Number of the chip for a chip-type driver. VMEmodules are not listed in this column since module names are already listed in the DRIVER column.

Device Driver Configuration Table

DRIVER	SYSTEM/BOARD	CHANNEL NAME	CHANNEL TYPE	ADDRESS	VECTOR NUMBER	VECTOR LEVEL	SOFT PRI	SIZE	CHIP NUMBER
ACIADRV	EXORmacs #1	'CACx'	XTACD1 ($66)	$FEE011	$8B	5	$30	$300	M6850
	EXORmacs #2	'CACx'	XTACD2 ($67)	$FEE015	$8B	5	$30		M6850
	MVME110-1	'CACx'	XTX110 ($68)	$FE8001	$1D	5	$30		M6850
DARTDRV	MVME115M #1	'CDAx'	XTDRTA ($6A)	$F82001	$C2	5	$30		SCN268
	MVME115M #2	'CDAx'	XTDRTB ($6B)	$F82011	$C2	5	$30		SCN268
DRVLIB	ALL								
EPCIDRV	M68KVM01 #1	'CEPx'	XTEPCI ($6E)	$F70011	$1D	5	$30		SCN268
	M68KVM01 #2	'CEPx'	XTEPCI ($6E)	$F70011	$1D	5	$30		SCN268
	MVME101	'CEPx'	XTEPCI ($6E)	$FE00B1	$1C	4	$30		SCN268
IPCDRB	M68KVM20 #1	'CFD1'	XTDIPC ($10)	DEVADD	VECTNO	4	$10	$1500	
	M68KVM20 #2	'CFD2'	XTDIPC ($10)	DEVADD	VECTNO	4	$10		
	M68KVM21 #1	'CUD1'	CTDIPC ($10)	DEVADD	VECTNO	3	$10		
	M68KVM21 #2	'CUD2'	CTDIPC ($10)	DEVADD	VECTNO	3	$10		
	M68KVM30 #1	'COM1'	XTMIPC ($11)	LV30$07	$F7	4	$10		
	M68KVM30 #2	'COM2'	XTMIPC ($11)	LV30$06	$F6	4	$10		
	M68KVM30 #3	'COM3'	XTMIPC ($11)	LV30$05	$F5	4	$10		
	M68KVM30 #4	'COM4'	XTMIPC ($11)	LV30$04	$F4	4	$10		
M300DRV	#1	'BUS0'	XTGBUS ($4A)	L300$01	$B0	4	$40	$1900	
	#2	'BUS1'	XTGBUS ($4A)	L300$02	$B0	4	$40		
M315DRV		'M315'	XTD315 ($23)	L315$01	$FF	3	$10	$D00	
M320DRV		'M320'	XTD320 ($25)	L320$01	$FD	3	$10	$D00	
M420DRV 5		'SAS5'	XTDSAS ($20)	L420$02	IOCVEC3	IOCLVL3	$10	$900	
M420DRV 8		'SAS8'	XTDSAS ($20)	L420$01	IOCVEC3	IOCLVL3	$10	$900	
M435DRV	#1	'MTA0'	XTM435 ($24)	L435$01	IOCVEC1	IOCLVL1	$10	$F00	
		'MTA0'	XTM435 ($24)	L435$01	IOCVEC2	IOCLVL2	$10		
		'MTA0'	XTM435 ($24)	L435$01	IOCVEC3	IOCLVL3	$10		
		'MTA0'	XTM435 ($24)	L435$01	IOCVEC4	IOCLVL4	$10		
	#2	'MTA1'	XTM435 ($24)	L435$02	IOCVEC1	IOCLVL1	$10		
		'MTA1'	XTM435 ($24)	L435$02	IOCVEC2	IOCLVL2	$10		
		'MTA1'	XTM435 ($24)	L435$02	IOCVEC3	IOCLVL3	$10		
		'MTA1'	XTM435 ($24)	L435$02	IOCVEC4	IOCLVL4	$10		

Device Driver Configuration Table (cont'd)

DRIVER	SYSTEM/BOARD	CHANNEL NAME	CHANNEL TYPE	ADDRESS	VECTOR NUMBER	VECTOR LEVEL	SOFT PRI	SIZE	CHIP NUMBER
M600DRV	#1	'AD01'	$80	L600$01	IOCVEC1	IOCLVL1	$FF	$500	
	#2	'AD01'	$80	L600$02	IOCVEC1	IOCLVL1	$FF		
M605DRV		'DAC1'	$80	L605$01	IOCVEC1	IOCLVL1	$50	$400	
M610DRV	#1	'IN01'	$80	L610$01	IOCVEC1	IOCLVL1	$FF	$700	
	#2	'IN02'	$80	L610$02	IOCVEC2	IOCLVL2	$FF	$700	
M615DRV		'DC01'	$80	L615$01	IOCVEC1	IOCLVL1	$50	$600	
M625DRV		'DC01'	$80	L615$01	IOCVEC1	IOCLVL1	$50	$500	
MFPDRV	MVME120	'CMFP'	XTMFPO ($6C)	$F20001	$6C	6	$30		MC68901
MPCCDRV	MVME050 #1	'CMCx'	XTMPC1 ($72)	$FF1000	$80	3	$30		R68560
	MVME050 #2	'CMCx'	XTMPC2 ($73)	$FF1041	$81	3	$30		R68560
MPSCDRV	M68KVM02 A	'CMPx'	XTS7A2 ($60)	$F70015	$8C	6	$30	$700	NEC7201
	M68KVM02 B	'CMPx'	XTS7B2 ($61)	$F70017	$8C	6	$30		NEC7201
	M68KVM03 A	'CMPx'	XTS7A3 ($74)	$F80065	$43	4	$30		NEC7201
	M68KVM03 B	'CMPx'	XTS7B3 ($75)	$F80067	$43	4	$30		NEC7201
	MVME400 #1 A	'CMPx'	XTS7AR ($64)	L400$01	IOCVEC4	IOCLVL4	$30		NEC7201
	MVME400 #1 B	'CMPx'	XTS7BR ($65)	L400$01+2	IOCVEC4	IOCLVL4	$30		NEC7201
	MVME400 #2 A	'CMPx'	XTS7AR ($64)	L400$02	IOCVEC4	IOCLVL4	$30		NEC7201
	MVME400 #2 B	'CMPx'	XTS7BR ($64)	L400$02+2	IOCVEC4	IOCLVL4	$30		NEC7201
P050DRV	MVME050	'CPRx'	XTPV050 ($53)	$FF1081	$8B	5	$10	$700	discretes
P115DRV	MVME115M	'CPRx'	XTPV115 ($54)	$F81081	$C1	5	$10		M68230
P117DRV	MVME117	'CPRx'	XTPV117 ($)	$F54001			$10		discretes
	MVME117 #2	'CZIx'	XTZI01 ($79)	$F48005			$30		Z8530
PIADRV	EXORmacs	'CPRx'	XTPEXM ($50)	$FEE009	$8B	5	$10	$600	M6821
	MVME101	'CPRx'	XTPRTL ($52)	$FE00C1	$1B	3	$10		M6821
	MVME410 #1 A	'CPRx'	XTPRIL ($52)	$L410$01	IOCVEC1	IOCLVL1	$10		M6821
	MVME410 #1 B	'CPRx'	XTPRIL ($52)	$L410$01+8	IOCVEC1	IOCLVL1	$10		M6821
	MVME410 #2 A	'CPRx'	XTPRIL ($52)	$L410$01	IOCVEC1	IOCLVL1	$10		M6821
	MVME410 #2 B	'CPRx'	XTPRIL ($52)	$L410$01+8	IOCVEC1	IOCLVL1	$10		M6821

Device Driver Configuration Table (cont'd)

DRIVER	SYSTEM/BOARD	CHANNEL NAME	CHANNEL TYPE	ADDRESS	VECTOR NUMBER	VECTOR LEVEL	SOFT PRI	SIZE	CHIP NUMBER
PV01DRV	M68KVM01 #1	'CPRx'	XTPVM1 (L$51)	$F70020	$1E	6	$10		discretes
	M68KVM01 #2	'CPRx'	XTPVM1 (L$51)	$F70022	$1E	6	$10		discretes
RADDRV		'RA01'	$80	LRAD$01	IOCVEC3	IOCLVL3	$50	$E00	
RIODRV		'RI01'	$80	LRIO$01	IOCVEC3	IOCLVL3	$50	$C00	
RWINDRV	#1	'WIN1'	XTDWIN ($21)	LWIN$01	IOCVEC3	IOCLVL3	$10	$800	
	#2	'WIN2'	XTDWIN ($21)	LWIN$02	IOCVEC3	IOCLVL3	$10		
SIODRV	M68KVM04 #1	'CSIx'	XTSI00 ($76)	$FFB0040	$44	5	$30		MK68564
	M68KVM04 #2	'CSIx'	XTSI00 ($77)	$FFB0050	$44	5	$30		MK68564
TERMDRV	VME/10	'CN00'	XTSEXS ($6F)	$8200	$42	3	$30		
TERMLIB	ALL								
VM22DRV		'VM22'	XTVM22 ($12)	DEVM22	VECTNO	3	$10		
ZIODRV	MVME117 #1	'CZIx'	XTZI00 ($78)	SF48001			$30		Z8530

A.2 INTERRUPT/VECTOR LEVELS

The following defines the relationship between Interrupt Levels and Vector values for the four interrupt capabilities on the I/O Channel. Note that IOCLVL1 corresponds to IOCVEC1, etc.

Interrupt/Vector Levels

VECTOR LEVEL	EXOR macs	VM01	VM02	VM03	VM04	VME101	VME110-1	VME115M	VME122	VME/10
IOCLVL1	N/A	N/A	2	3	N/A	3	1	2	2	2
IOCLVL2	N/A	N/A	3	4	N/A	4	2	3	3	4
IOCLVL3	N/A	N/A	4	5	N/A	5	3	4	4	5
IOCLVL4	N/A	N/A	5	6	N/A	2	4	5	5	6
IOCVEC1	N/A	N/A	$71	$49	N/A	$1B	$19	$50	$50	$41
IOCVEC2	N/A	N/A	$72	$4A	N/A	$1C	$1A	$51	$51	$43
IOCVEC3	N/A	N/A	$73	$4B	N/A	$1D	$1B	$52	$52	$44
IOCVEC4	N/A	N/A	$74	$4C	N/A	$1A	$1C	$53	$53	$45

A.3 SYSGEN PARAMETER MATRIX

The following matrix defines the values for all SYSGEN parameters for several VMEsystem configurations. The CSYSTEMS.SYMBOLS.LS listings provide convenient references to determine the location of each parameter.

Note that the parameters for the MVME120 also apply to the MVME121, and the parameters for the MVME122 also apply to the MVME123.

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
&ADDRESS	-	$FE60E1	-	-	-	-
&CRTDV	$434E3032	$434E3033	$434E3032	$434E3033	$434E3033	$434E3033
&DEFVOL	$4	$6	$4	$6	$6	$6
&DEVADD	-	-	-	-	-	-
&DSKDV	$2F00	$2F00	$2F00	$2F00	$2F00	$2F00
&FILENAM	&.XPIA101.SI	&.XPIA410.SI	-	-	-	&.XPIA410.SI
&IOCBASE	$0	$FE6000	$FFE000	$FFE000	$FFE000	$F1C000
&M4200	-	'HSWIN15'	-	-	-	-
&M4201	-	'HSWIN15'	-	-	-	-
&M4202	-	'F5DDDSI'	-	-	-	-
&M4203	-	'F5DDDSI'	-	-	-	-
&M420FLG	$0	$1	$0	$0	$0	$0
&MPSCFLG	$0	$1	$0	$0	$0	$1
&NF420	-	$2	-	-	-	-
&NH420	-	$1	-	-	-	-
&PCDRV	$0	$0	$0	$0	$0	$0
&PIAFLAG	$1	$1	$0	$0	$0	$1
&PRTDV	$50523120	$50523120	$50523120	$50523120	$50523120	$50523120
&SDRVADD	-	$49400	-	-	-	$BB00
&SDRVR	-	MSPR	-	-	-	MSPR
&SERFLAG	$1	$2	$1	$2	$2	$1
&SPRFLAG	$0	$1	$0	$0	$0	$1
&SUPFLAG	$0	$0	$0	$0	$0	$0
&T	-	-	-	$F80006	$F80006	-
&TOTDSK	$6	$D	$5	$A	$A	$2
&TOTPRT	$1	$1	$1	$1	$1	$1
&TOTTERM	$2	$3	$2	$3	$3	$3
&VECTNO	-	-	-	-	-	-
ACIADRV	-	$49100	-	-	-	-
ASMLS	SYSGEN.TF	SYSGEN.TF	SYSGEN.TF	SYSGEN.TF	SYSGEN.TF	SYSGEN.TF
ASMLSW	$0	$0	$0	$0	$0	$0
ASN	$0	$0	$7F	$7F	$7F	$7F
AUTOLOGN	$1	$1	$1	$1	$1	$1
AUTOTERM	'CN00'	'CN00'	'CN00'	'CN00'	'CN00'	'CN00'
BATCHPGE	$2	$2	$2	$2	$2	$2
BATDLY	$3E80	$3E80	$3E80	$3E80	$3E80	$3E80
BCLRV	-	-	-	-	-	-
CACHE020	-	-	-	-	-	-

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
CACHEF	$0	$0	$0	$F80006	$F80006	$0
CACHESD	-	-	-	-	-	-
CACHESI	-	-	-	-	-	-
CACHEUD	-	-	-	-	-	-
CACHEUI	-	-	-	-	-	-
CHAINBAT	$1	$1	$1	$1	$1	$1
CLOCKFRQ	$320	$320	$7A1200	$0	$0	$0
CMULT	-	$2	$2	$2	$2	$2
CONBATCH	$1	$1	$1	$1	$1	$1
CONT3151	0	3	-	1	1	-
CONT3152	1	3	-	6	6	-
CONT320	-	4	0	0	0	-
CONT4205	-	1	-	-	-	-
CONTWIN1	-	0	2	2	2	0
CONTWIN2	-	5	5	5	5	1
DARTDRV	-	-	$18100	-	-	-
DARTSPR	-	-	$17F00	-	-	-
DCP$RTO	$0	$0	$0	$0	$0	$0
DCP$VSS	$100	$100	$100	$100	$100	$100
DCP$WTO	$0	$0	$0	$0	$0	$0
DCQPGE	$2	$2	$2	$2	$2	$2
DEFAULT	SYS: 0. &	SYS: 0. &	SYS: 0. &	SYS: 0. &	SYS: 0. &	SYS: 0. &
DEFDAT	$4	$4	$4	$4	$4	$4
DEFFAB	$1	$1	$1	$1	$1	$1
DPRAM	-	-	-	-	-	-
DRVL	$14E00	$44E00	$15200	$6200	$6000	$6300
EETS$	$1	$1	$1	$1	$1	$1
EETSTR	$20000	$52900	$21300	$13100	$12F00	$15000
EPCIDRV	$17000	-	-	-	-	-
FAIL	-	-	-	-	-	-
FHSIOS	$1	$1	$1	$1	$1	$1
FHSASR	$17A02	$4A302	$18D02	$AB02	$A902	$CA02
FHSSTR	$17A00	$4A300	$18D00	$AB00	$A900	$CA00
FMS$	$1	$1	$1	$1	$1	$1
FMSASR	$1AC02	$4D502	$1BF02	$D002	$D802	$FC02
FMSSTR	$1AC00	$4D500	$1BF00	$D000	$D800	$FC00
FOUR	4	4	4	4	4	4
GST	$2	$3	$2	$3	$3	$3

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
HOGMODE	-	-	-	-	-	-
INTSTR	$[illegible]7200	$5A600	$28400	$1B800	$1B600	$1C100
IOCBASE	$[illegible]	$FE6000	$FFE000	$FFE000	$FFE000	$F1C000
IOCLVL1	$[illegible]	$1	$2	$2	$2	$2
IOCLVL2	$[illegible]	$2	$3	$3	$3	$4
IOCLVL3	$[illegible]	$3	$4	$4	$4	$5
IOCLVL4	$[illegible]	$4	$5	$5	$5	$6
IOCM	[illegible]ESG	IOSG	IOSG	IOSG	IOSG	IOSG
IOCSTR	$[illegible]5D00	$58600	$27000	$18E00	$18C00	$1AD00
IOCVEC1	$[illegible]18	$19	$50	$50	$50	$41
IOCVEC2	$[illegible]1C	$1A	$51	$51	$51	$43
IOCVEC3	$[illegible]1D	$1B	$52	$52	$52	$44
IOCVEC4	$[illegible]1A	$1C	$53	$53	$53	$45
IOSASR	$[illegible]9002	$4B902	$1A302	$C102	$BF02	$E002
IOSSTR	$[illegible]9000	$4B900	$1A300	$C100	$BF00	$E000
IOV	$[illegible]1	$1	$1	$1	$1	$1
IPCDRV	-	-	-	-	-	-
KBDOVRD	-	-	-	-	-	$0
KILVECT	-	-	-	-	-	-
L050$01	$FF1000	$FF1000	$FF1000	$FF1000	$FF1000	$FF1000
L300$01	$FF0400	$FF0400	$FF0400	$FF0400	$FF0400	$FF0400
L300$02	$FF0440	$FF0440	$FF0440	$FF0440	$FF0440	$FF0440
L315$01	$FF0000	$FF0000	$FF0000	$FF0000	$FF0000	$FF0000
L315$02	$FF0200	$FF0200	$FF0200	$FF0200	$FF0200	$FF0200
L320$01	$FFB00D	$FFB000	$FFB00D	$FFB000	$FFB000	$FFB00D
L320$02	$FFAC0D	$FFAC00	$FFAC0D	$FFAC00	$FFAC00	$FFAC0D
L331$01	$FF3000	$FF3000	$FF3000	$FF3000	$FF3000	$FF3000
L331$02	$FF3100	$FF3100	$FF3100	$FF3100	$FF3100	$FF3100
L331$03	$FF3200	$FF3200	$FF3200	$FF3200	$FF3200	$FF3200
L331$04	$FF3300	$FF3300	$FF3300	$FF3300	$FF3300	$FF3300
L331$05	$FF3400	$FF3400	$FF3400	$FF3400	$FF3400	$FF3400
L331$06	$FF3500	$FF3500	$FF3500	$FF3500	$FF3500	$FF3500
L333$01	$FF3800	$FF3800	$FF3800	$FF3800	$FF3800	$FF3800
L333$02	$FF3900	$FF3900	$FF3900	$FF3900	$FF3900	$FF3900
L333$03	$FF3A00	$FF3A00	$FF3A00	$FF3A00	$FF3A00	$FF3A00
L333$04	$FF3B00	$FF3B00	$FF3B00	$FF3B00	$FF3B00	$FF3B00
L333$05	$FF3C00	$FF3C00	$FF3C00	$FF3C00	$FF3C00	$FF3C00
L333$06	$FF3D00	$FF3D00	$FF3D00	$FF3D00	$FF3D00	$FF3D00

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
L400$01	-	$FE61CD	$FFE1CD	$FFE1CD	$FFE1CD	$F1C1CD
L400$02	-	$FE61AD	$FFE1AD	$FFE1AD	$FFE1AD	$F1C1AD
L410$01	-	$FE61E1	$FFE1E1	$FFE1E1	$FFE1E1	$F1C1E1
L410$02	-	$FE6121	$FFE121	$FFE121	$FFE121	$F1C121
L420$01	-	$FE60F1	$FFE0F1	$FFE0F1	$FFE0F1	$F1C0F1
L420$02	-	$FE60E1	$FFE0E1	$FFE0E1	$FFE0E1	$F1C0E1
L435$01	-	$FE63E9	$FFE3E9	$FFE3E9	$FFE3E9	$F1C3E9
L435$02	-	$FE63A9	$FFE3A9	$FFE3A9	$FFE3A9	$F1C3A9
L600$01	-	$FE7E03	$FFFE03	$FFFE03	$FFFE03	$F1DE03
L600$02	-	$FE7C03	$FFFC03	$FFFC03	$FFFC03	$F1DC03
L605$01	-	$FE1601	$FFF601	$FFF601	$FFF601	$F1D601
L610$01	-	$FE6005	$FFE005	$FFE005	$FFE005	$F1C005
L610$02	-	$FE6009	$FFE009	$FFE009	$FFE009	$F1C009
L615$01	-	$FE6003	$FFE003	$FFE003	$FFE003	$F1C003
L625$01	-	$FE6001	$FFE001	$FFE001	$FFE001	$F1C001
LDR$	$1	$1	$1	$1	$1	$1
LDRSTR	$24600	$56F00	$25900	$17700	$17500	$19600
LINKLS	SYSGEN. TF	SYSGEN. TF	SYSGEN. TF	SYSGEN. TF	SYSGEN. TF	SYSGEN. TF
LINKLSW	$0	$0	$0	$0	$0	$0
LOGMSG1	VERSAdos	VERSAdos	VERSAdos	VERSAdos	VERSAdos	VERSAdos
LPDA$01	$FE00C1	-	$F81001	-	-	-
LPDA$02	-	-	-	-	-	-
LRAD$01	-	$FE6E01	$FFEE01	$FFEE01	$FFEE01	$F1CE01
LRIO$01	-	$FE6021	$FFE021	$FFE021	$FFE021	$F1C021
LTDA$01	$FE00A1	$FE8001	$F82001	$F20001	$F20001	-
LTDA$02	$FE00B1	$0	$F82011	-	-	-
LUMAX	$9	$10	$8	$D	$D	$5
LV30$01	$FF1000	$FF1000	$FF1000	$FF1000	$FF1000	$FF1000
LV30$02	$FF1200	$FF1200	$FF1200	$FF1200	$FF1200	$FF1200
LV30$03	$FF1400	$FF1400	$FF1400	$FF1400	$FF1400	$FF1400
LV30$04	$FF1600	$FF1600	$FF1600	$FF1600	$FF1600	$FF1600
LWIN$01	-	$FE60D3	$FFE0D3	$FFE0D3	$FFE0D3	$F1C0D3
LWIN$02	-	$FE60E3	$FFE0E3	$FFE0E3	$FFE0E3	$F1C0E3
M3150$1	'H5WIN10'	'H5WIN15'	-	'H5WIN15'	'H5WIN15'	-
M3150$2	'H5WIN10'	'H5WIN15'	-	'H5WIN15'	'H5WIN15'	-
M3151$1	'H5WIN10'	'H5WIN15	-	'H5WIN15'	'H5WIN15'	-
M3151$2	'H5WIN10'	'H5WIN15'	-	'H5WIN15'	'H5WIN15'	-
M3154$1	'F8SDDSM'	'F8SDDSM'	-	'F8SDDSM'	'F8SDDSM'	-

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
M3154$2	'F8SDDSM'	'F8SDDSM'	-	"F8SDDSM'	'F8SDDSM'	-
M3155$1	'F8SDDSM'	'F8SDDSM'	-	"F8SDDSM'	'F8SDDSM'	-
M3155$2	'F8SDDSM'	'F8SDDSM'	-	"F8SDDSM'	'F8SDDSM'	-
M3156$1	'F5DDDSI'	'F5DDDSI'	-	"F5DDDSI'	'F5DDDSI'	-
M3156$2	'F5DDDSI'	'F5DDDSI'	-	"F5DDDSI'	'F5DDDSI'	-
M3157$1	'F5DDDSI'	'F5DDDSI'	-	"F5DDDSI'	'F5DDDSI'	-
M3157$2	'F5DDDSI'	'F5DDDSI'	-	"F5DDDSI'	'F5DDDSI'	-
M315DRV	$16300	$47700	-	$8400	$8200	-
M320$DD	-	$36	$36	$36	$36	-
M320$HD	-	$F	$F	$F	$F	-
M320$LT5	-	$23	$23	$23	$23	-
M320$LT8	-	$23	$23	$23	$23	-
M320$LTH	-	$0	$0	$0	$0	-
M320$SD	-	$1B	$1B	$1B	$1B	-
M320$ST5	-	$F	$F	$F	$F	-
M320$ST8	-	$F	$F	$F	$F	-
M320$STH	-	$0	$0	$0	$0	-
M3200$1	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	-
M3201$1	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	-
M3202$1	-	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	-
M3203$1	-	'F8SDSSI'	'F8SDSSI'	'F8SDSSI'	'F8SDSSI'	-
M320DRV	-	$48400	$16700	$7700	$7500	-
M42050$1	-	'H5WIN15'	-	-	-	-
M42051$1	-	'H5WIN15'	-	-	-	-
M42052$1	-	'F5DDDSI'	-	-	-	-
M42053$1	-	'F5DDDSI'	-	-	-	-
M420DRV	-	$46E00	-	-	-	-
MAXLU	$9	$10	$8	$0	$0	$8
MEMBEG	$17A00	$4A300	$18D00	$AB00	$A900	$CA00
MEMEND1	$200000	$200000	$200000	$27F00	$27F00	$2FF00
MEMEND2	$0	$0	$0	$27F00	$27F00	$2FF00
MEMEND3	$0	$0	$0	$200000	$200000	$280000
MFPDRV	-	-	-	$9C00	$9A00	-
MMU	$0	$0	$1	$1	$0	$1
MPCCDRV	-	-	-	$A000	$9E00	-
MPSCDRV	-	$49600	-	-	-	$BD00
MPSCSPR	-	$49400	-	-	-	$BB00
MPSCSUP	-	-	-	-	-	-

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
NF315$1	$4	$4	-	$4	$4	-
NF315$2	$0	$0	-	$0	$0	-
NF320$1	-	$1	$1	$1	$1	-
NF4205$1	-	$2	-	-	-	-
NF4208$1	-	-	-	-	-	-
NFRWIN$1	-	$2	$2	$2	$2	$1
NFRWIN$2	-	$0	$0	$0	$0	$0
NFV20$1	-	-	-	-	-	-
NFV20$2	-	-	-	-	-	-
NFV21$1	-	-	-	-	-	-
NFV21$2	-	-	-	-	-	-
NFV22$1	-	-	-	-	-	-
NH315$1	$2	$1	-	$1	$1	-
NH315$2	-	$0	-	$0	$0	-
NH320$1	-	$1	$1	$1	$1	-
NH4205$1	-	$1	-	-	-	-
NH4208$1	-	-	-	-	-	-
NHRWIN$1	-	$1	$1	$1	$1	$1
NHRWIN$2	-	$0	$0	$0	$0	$0
NHV21$1	-	-	-	-	-	-
NHV21$2	-	-	-	-	-	-
NHV22$1	-	-	-	-	-	-
NODEFVOL	$4	$6	$4	$6	$6	$6
NODIFFIL	$18	$1E	$18	$1E	$1E	$1E
NOFILES	$18	$1E	$18	$1E	$1E	$1E
NOLOGON	$3	$3	$3	$3	$3	$3
NOLOGONS	$2	$3	$2	$3	$3	$3
NOLPRT	$1	-	$1	-	-	-
NOLTERM	$2	$1	$2	$1	$1	$1
NORWIN	$0	$1	$1	$1	$1	$1
NOTASKS	$C	$F	$C	$F	$F	$F
NOTNT	$0	$0	$0	$0	$0	$0
NOVM20	-	-	-	-	-	-
NOVM21	-	-	-	-	-	-
NOVM22	-	-	-	-	-	-
NOVM30	-	-	-		-	-
NP050$1	-	-	-	$1	$1	-
NPV30$1	-	-	-	-	-	-

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
NPV30$2	-	-	-	-	-	-
NPV30$3	-	-	-		-	-
NPV30$4	-	-	-	-	-	-
NRAD	$0	$0	$0	$0	$0	$0
NRIO	$0	$0	$0	$0	$0	$0
NT050$1	-	-	-	$2	$2	-
NTV30$1	-	-	-	-	-	-
NTV30$2	-	-	-	-	-	-
NTV30$3	-	-	-	-	-	-
NTV30$4	-	-	-	-	-	-
NU400$1	-	$2	-	-	-	$2
NU400$2	-	$0	-	-	-	-
NU410$1	-	$1	-	-	-	$1
NU410$2	-	$0	-	-	-	$0
NVME050	$0	$0	$0	$1	$1	$0
NVME300	$0	$0	$0	$0	$0	$0
NVME315	$1	$1	$0	$1	$1	$0
NVME316	$0	$0	$1	$1	$1	$0
NVME320	$0	$1	$1	$1	$1	$0
NVME400	$0	$1	$0	$0	$0	$1
NVME410	$0	$1	$0	$0	$0	$1
NVME4205	-	$1	-	-	-	-
NVME4208	-	$0	-	-	-	-
NVME435	$0	$0	$0	$0	$0	$0
NVME600	$0	$0	$0	$0	$0	$0
NVME605	$0	$0	$0	$0	$0	$0
NVME610	$0	$0	$0	$0	$0	$0
NVME615	$0	$0	$0	$0	$0	$0
NVME620	$0	$0	$0	$0	$0	$0
NVME625	$0	$0	$0	$0	$0	$0
OFFBD$HI	$FFF	$FFF	$FFF	$1FFFF	$1FFFF	$05FFFF
OFFBD$LO	$0	$0	$0	$0	$0	$000000
ONBD$HI	$FFF	$FFF	$FFF	$1FFFF	$1FFFF	$5FFFF
ONBD$LO	$0	$0	$0	$0	$0	$0
P050DRV	-	-	-	$A400	$A200	-
P115DRV	-	-	$18600	-	-	-
PAGESIZE	$100	$100	$100	$800	$800	$100
PANEL	$0	$0	$0	$0	$0	$0

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
PAT	$1	$1	$1	$1	$1	$1
PCDRV	$0	$0	$0	$0	$0	$0
PCP$AFF	$0	$0	$0	$0	$0	$0
PCP$ELC	$0	$0	$0	$0	$0	$0
PCP$LNFD	$0	$0	$0	$0	$0	$0
PCP$LRL	$84	$84	$84	$84	$84	$84
PCP$REC	$84	$84	$84	$84	$84	$84
PCP$RSZ	$42	$42	$42	$42	$42	$42
PCP$TLRL	$0	$0	$0	$0	$0	$0
PCP$WTO	$1D4C0	$1D4C0	$1D4C0	$1D4C0	$1D4C0	$1D4C0
PIADRV	$17400	$49D00	-	-	-	$C400
PTMVECT	-	-	-	-	-	-
PV01DRV	-	-	-	-	-	-
RAM$SQ	$0	$0	$0	$0	$0	$D00000
REVNUMBR	4.4	4.4	4.4	4.4	4.4	4.4
ROMEADDR	$0	$0	$0	$0	$0	$0
ROMSADDR	$10000	$40000	$10000	$1000	$1000	$1000
RWIN0$1	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'
RWIN0$2	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'
RWIN1$1	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'
RWIN1$2	-	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'	'H5WIN15'
RWIN2$1	-	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'
RWIN2$2	-	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'
RWIN3$1	-	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'
RWIN3$2	-	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'	'F5DDDSI'
RWINDRV	-	$46300	$17400	$9100	$8F00	$7800
SECURITY	$1	$1	$1	$1	$1	$1
SERFLAG	$1	$2	$1	$2	$2	$1
SERPRTS	-	-	-	-	-	-
SET1	-	-	-	DONT-CARE	DONT-CARE	-
SET2	-	-	-	-	-	-
SIOBASE	$FF0000	$FF0000	$FF0000	$FF0000	$FF0000	$FF0000
SIODRV	-	-	-	-	-	-
SIX	6	6	6	6	6	6
SPCCMD	$1	$1	$1	$1	$1	$1
STACK	$C00	$C00	$D00	$D00	$D00	$E00
STARTRMS	$10100	$40100	$10100	$1100	$1100	$1100
SWABRT	-	-	-	-	-	-

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
SYSFAIL	-	-	-	-	-	-
TCP$BITS	$0	$0	$0	$0	$0	$0
TCP$BRC	$3	$3	$3	$3	$3	$3
TCP$BRT	$E	$E	$E	$E	$E	$E
TCP$CLC	$18	$18	$18	$18	$18	$18
TCP$DOP	$F	$F	$F	$F	$F	$F
TCP$ECHO	$0	$0	$0	$0	$0	$0
TCP$EOL	$D0A0000	$D0A0000	$D0A0000	$D0A0000	$D0A0000	$D0A0000
TCP$HCPY	$0	$0	$0	$0	$0	$0
TCP$MODM	$0	$0	$0	$0	$0	$0
TCP$NLS	$0	$0	$0	$0	$0	$0
TCP$OFFH	$0	$0	$0	$0	$0	$0
TCP$PNUL	$0	$0	$0	$0	$0	$0
TCP$PRTY	$0	$0	$0	$0	$0	$0
TCP$REC	$50	$50	$50	$50	$50	$50
TCP$RLN	$1A	$1A	$1A	$1A	$1A	$1A
TCP$RSZ	$18	$18	$18	$18	$18	$18
TCP$RTO	$D8BA0	$D8BA0	$D8BA0	$D8BA0	$D8BA0	$D8BA0
TCP$RTV	$DDE0000	$DDE0000	$DDE0000	$DDE0000	$DDE0000	$DDE0000
TCP$STPB	$0	$0	$0	$0	$0	$0
TCP$TAHD	$0	$0	$0	$0	$0	$0
TCP$TFUL	$1	$1	$1	$1	$1	$1
TCP$TRC	$0	$0	$0	$0	$0	$0
TCP$TTP	$0	$0	$0	$0	$0	$0
TCP$USEP	$0	$0	$0	$0	$0	$0
TCP$WTO	$D8BA0	$D8BA0	$D8BA0	$D8BA0	$D8BA0	$D8BA0
TCP$XCTL	$0	$0	$0	$0	$0	$0
TCP$XOF	$13	$13	$13	$13	$13	$13
TCP$XON	$0	$0	$0	$0	$0	$0
TERMDRV	-	-	-	-	-	$8300
TERMLIB	$15000	$45000	$15400	$6400	$6200	$6500
TERMOCNT	$2	$2	$2	$2	$2	$2
TIMER	$FE0000	$FE8010	$F81001	$F20001	$F20001	$F1A081
TIMINTV	$A	$A	$A	$A	$A	$10
TIMSLIC	$2	$2	$2	$2	$2	$2
TOTDSK	$6	$0	$5	$A	$A	$2
TOTTERM	$2	$3	$2	$3	$3	$3
TRACE	$0	$0	$0	$0	$0	$0

Parameter	VME101	VME110	VME115	VME120	VME122	VMES10
TRCFLAG	$0	$0	$0	$0	$0	$0
TWO	2	2	2	2	2	2
UDR	$1	$1	$1	$1	$1	$1
UST	$2	$2	$2	$2	$2	$2
WHERLOAD	$0	$0	$0	$0	$0	$0
WORKLS	SYSASM.LS	SYSASM.LS	SYSASM.LS	SYSASM.LS	SYSASM.LS	SYSASM.LS
ZERO	0	0	0	0	0	0

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX B

MEDIA AND DISK CONTROLLER ATTRIBUTE TABLES

B.1 MEDIA ATTRIBUTE TABLE

The following table lists the media files (.SI) from User 9998 (null catalog), the media related attributes values, and a cross reference displaying which media files can be used with the currently supported disk controllers.

	ATT_WORD	SECT_PER_TRK	NUM_HEADS	CYL_DISK	SPIRAL_OFF	BYTES_PER_SECTOR	START_HEAD_NUM	CYL_DRIVE	PRE_COMP	SECT_DRIVE	STEP_RATE	R_W_PRECOMP	RWIN1	SASI5	SASI8	VM20 Floppy IPC	VM21 LDC	VM22	VME315	VME320
F8SDSSI floppy 8", IBM sngl dens, sngl sided	$0208	26	2	77	0	128	0	77	38	0	0	44	●		●			●	●	●
F8DDDSI floppy 8", IBM dbl dens, dbl sided	$0200	26	2	77	0	256	0	77	38	0	0	44	●		●			●	●	●
FXLRK25 fixed LARK 25mb	$0090	64	2	624	0	256	2	624	0	0	0	0					●	●		
RMLRK25 removable LARK 25mb	$0080	64	2	624	0	256	0	624	0	0	0	0					●	●		
FXLRK08 fixed LARK 8mb	$0090	64	2	206	0	256	2	206	0	0	0	0					●	●		
RMLRK08 removable LARK 8mb	$0080	64	2	206	0	256	0	206	0	0	0	0					●	●		
FXCMD80 fixed CMD 80mb	$0010	64	5	823	2	256	16	823	0	0	0	0					●	●		
FXCMD16 fixed CMD 16mb	$0010	64	1	823	0	256	16	823	0	0	0	0					●	●		
RMCMD16 removable CMD 16mb	$0010	64	1	823	0	256	0	823	0	0	0	0					●	●		
F5DDDSI floppy 5-1/4", IBM dbl dens, dbl sided	$004F	16	2	80	0	256	0	80	40	0	0	0	●	●				●	●	●
F8SDSSM floppy 8", Motorola sngl dens, sngl sided	$0200	26	2	77	0	128	0	77	38	0	0	44	●			●	●	●	●	
F8SDDSM floppy 8", Motorola sngl dens, dbl sided	$0204	26	2	77	0	128	0	77	38	0	0	44	●			●	●	●	●	
H5WIN40 Win 5-1/4", 40mb	$0010	32	6	830	0	256	0	830	400	0	0	831	●	●						●
H5WIN15 Win 5-1/4", 15mb	$0010	32	6	306	0	256	0	306	153	0	0	307	●	●					●	●
H5WIN12 Win 5-1/4", 12mb	$0010	32	4	360	0	256	0	360	128	0	0	361							●	
H5WIN10 Win 5-1/4", 10mb	$0010	32	4	306	0	256	0	306	153	0	0	307	●	●					●	●
H5WIN05 Win 5-1/4", 5mb	$0010	32	2	306	0	256	0	306	153	0	0	307	●	●					●	●
H8WIN10 Win 8", 10mb	$0210	32	4	256	0	256	0	256	128	0	0	257	●		●				●	●

MEDIA ATTRIBUTE TABLE

B.2 DISK CONTROLLER ATTRIBUTE TABLE

The Disk Controller Attribute Table is a cross reference displaying the controller related attributes for the currently supported disk controllers.

B

DISK CONTROLLER ATTRIBUTE TABLE	RWIN1	SASI5	SASI8	VM20 floppy IPC	VM21 UDC	VM22	VME315	VME320		
CHAN_ID	WIND	SAS5	SAS8	CFD1	CUD1	VM22	M315	M320		
ATT_MASK hard disk	0410	0015	0210	N/A	03FF	0490	0110	0210		
ATT_MASK floppy disk	001F	0015	0215	03FF	03FF	025F	021F	0215		
PAR_MASK	1AF3	02F3	02F3	FFF3	FFF3	5FF3	DBF3	53F3		
INTERLEAVE hard disk	1	4	4	n/a	0	1	5	11		
INTERLEAVE floppy disk	1	8	13	1	1	1	1	1		
ECC_LEN	0	0	0	0	0	0	11	0		

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX C

MODULE BASE ADDRESSES AND OFFSETS

The following defines the VERSAdos Base Addresses and Board Offsets for a variety of Motorola components.

C

Base Addresses

System	IPL.SY	I/O Channel	SIO Address	STARTRMS	CRASHSAV	SYSPAR
EXORmacs	$0	N/A	$FF0000	$C00	$400	$900
VM01	N/A	N/A	$FF0000	$F00	$800	$C00
VM02	$10000	$F80000	$FF0000	$F00	$800	$C00
VM03	$10000	$FA0000	$FF0000	$1200	$800	$F00
VM04	$10000	N/A	$FFFF0000	$2700	$2000	$2400
VME101	$10000	$FFE000	$FF0000	$10100	$800	$C00
VME110	$40000	$FE6000	$FF0000	$40100	$800	$C00
VME115	$10000	$FFE000 (VME316)	$FF0000	$10100	$800	$D00
VME120	$10000	$FFE000 (VME316)	$FF0000	$1100	$800	$D00
VME122	$10000	$FFE000 (VME316)	$FF0000	$1100	$800	$D00
VME128	$10000	$FFE000 (VME316)	$FF0000	$1100	$800	$D00
VMES10	$10000	$F1C000	$FF0000	$1100	$A00	$E00

Board Offsets

I/O Channel

Board	Offset
VME400 #1	$1CD
#2	$1AD
VME410 #1	$1E1
#2	$121
VME420 #1	$0F1
#2	$0E1
VME435 #1	$3E9
#2	$3A9
VME600 #1	$1E03
#2	$1C03
VME605	$1601
VME610 #1	$005
#2	$009
VME615	$003
VME625	$001
RAD	$E01
RIO	$021
RWIN #1	$003
#2	$0E3

SIO Address

Board	Offset
VME050	$1000
VME300 #1	$400
#2	$440
VME315 #1	$000
#2	$200
VME320 #1	$8000
#2	$AC00
VME331 #1	$3000
#2	$3100
#3	$3200
#4	$3300
#5	$3400
#6	$3500
VME333 #1	$3800
#2	$3900
#3	$3A00
#4	$3B00
#5	$3C00
#6	$3D00
VM30 #1	$1000
#2	$1200
#3	$1400
#4	$1600

C

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX D

RS-232C INTERFACE SPECIFICATION FOR SERIAL DRIVERS

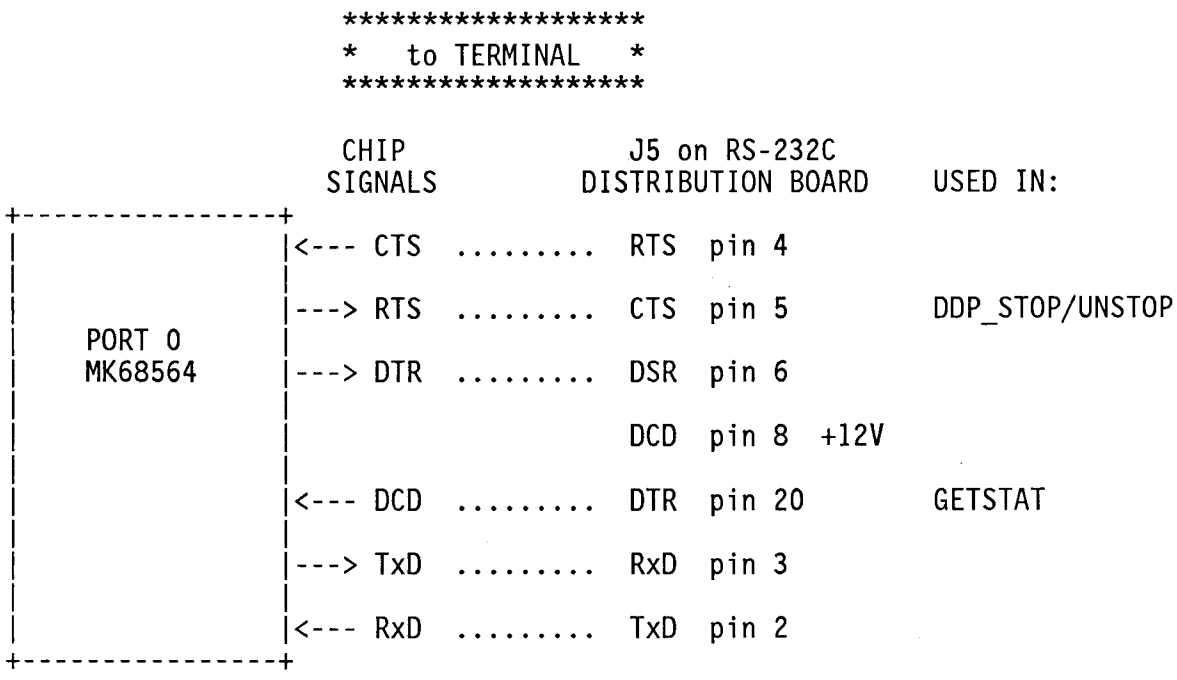
D.1 RS-232C INTERFACE SPECIFICATION

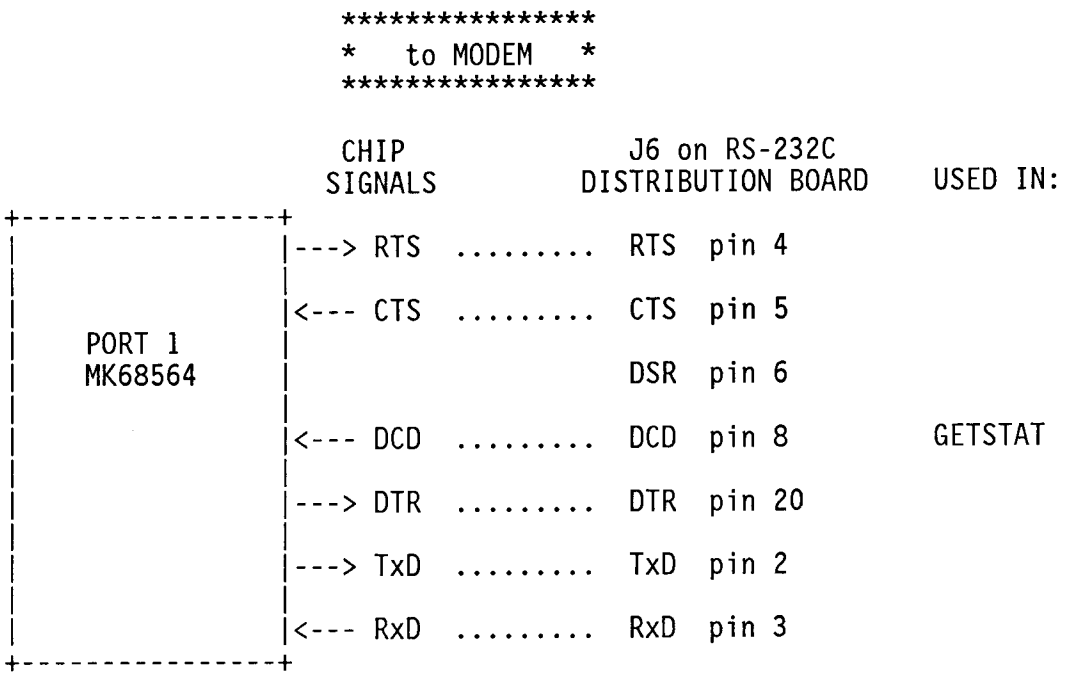
This appendix provides configuration information for most serial RS-232C devices implemented on various Motorola components.

To assist in the hardware configuration of a serial device, this appendix describes the chip signal relationship to the RS-232C connector for both terminal and modem connections.

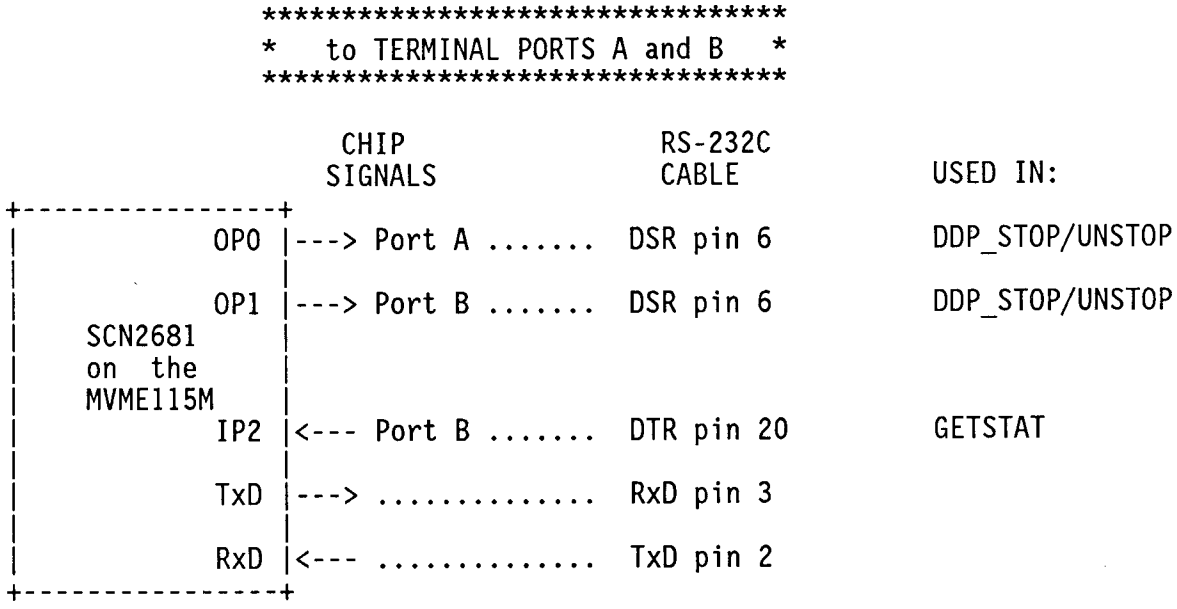
D

D.1.1 SIODRV/MK68564 on the M68KVM04





D.1.2 DARTDRV/SCN2681 on the MVME115M



NOTE: Port A is always "READY"

```
                    **********************************
                    *       to MODEM Port B          *
                    **********************************

                        CHIP                RS-232C
                       SIGNALS              CABLE            USED IN:
   +----------------+
   |            OP0 |---> Port A .......  DSR pin 6
   |                 |
   |            OP1 |---> Port B........  DSR pin 6
   |    SCN2681      |
   |    on  the      |
   |    MVME115M     |
   |            IP2 |<--- Port B .......  DTR pin 20        GETSTAT
   |                 |
   |            TxD |---> ..............  TxD pin 2
   |                 |
   |            RxD |<--- ..............  RxD pin 3
   +----------------+
```

NOTE: Port A is always "READY"

D.1.3 SCCDRV/Z8530 on the MVME117

```
                    ***************************
                    *   to TERMINAL Port A    *
                    ***************************

                        CHIP              FRONT PANEL
                       SIGNALS            RS-232C PORT          USED IN:
   +----------------+
   |                |<--- RxD  .........  TxD  pin 2
   |                |
   |                |---> TxD  .........  RxD  pin 3
   |    Z8530       |
   |    on  the     |<--- DCD  .........  RTS  pin 4
   |    MVME117     |
   |                |          .........  CTS  pin 5 +12V
   |                |
   |                |---> RTS  .........  DSR  pin 6        DDP_STOP/UNSTOP
   |                |
   |                |          .........  DCD  pin 8 +12V
   |                |
   |                |<--- CTS  .........  DTR  pin 20       GETSTAT
   +----------------+
```

D

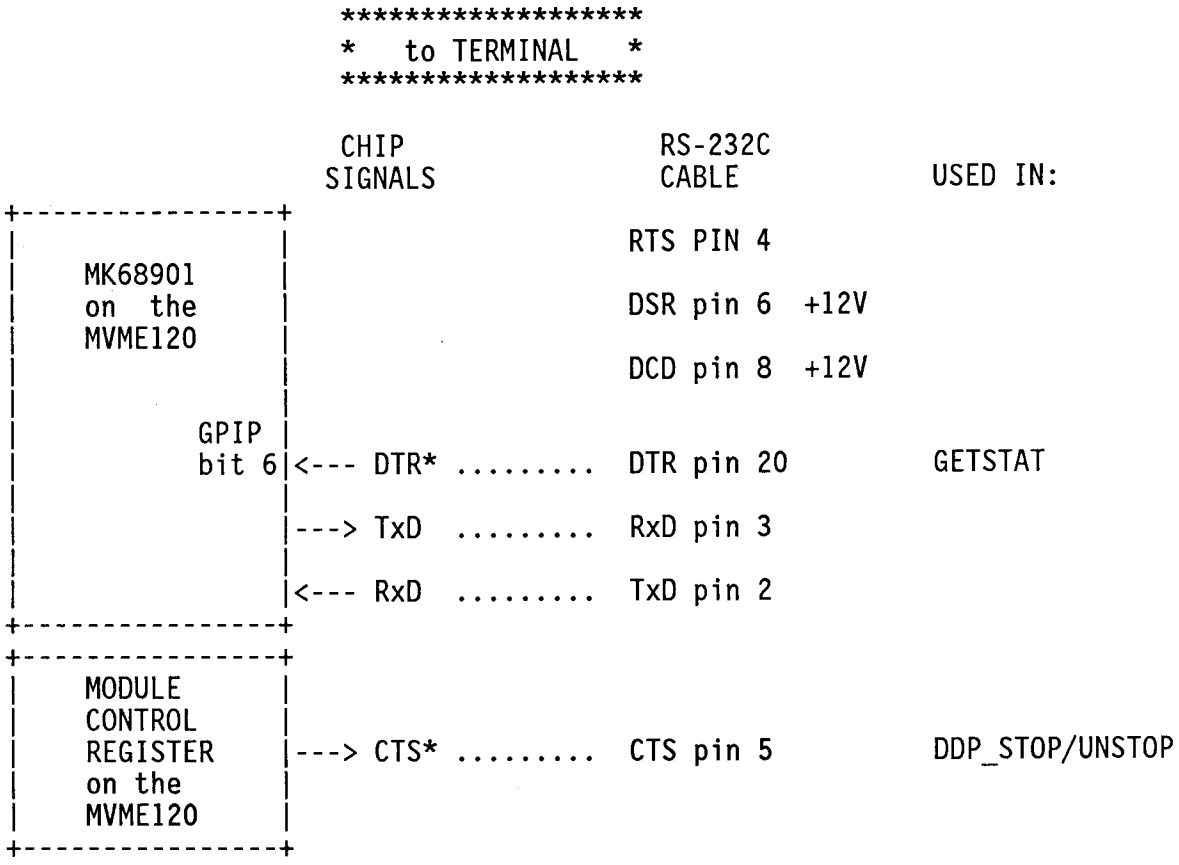
```
**************************
*   to TERMINAL Port B   *
**************************
```

	CHIP SIGNALS		TRANSITION BOARD		USED IN:
Z8530 on the MVME117	<--- RxD	.....	TxD	pin 2	
	---> TxD	.....	RxD	pin 3	
	<--- DCD	.....	RTS	pin 4	
		.....	CTS	pin 5 +12V	
	---> RTS	.....	DSR	pin 6	DDP_STOP/UNSTOP
		.....	DCD	pin 8 +12V	
	<--- CTS	.....	DTR	pin 20	GETSTAT

```
************************
*   to MODEM Port B    *
************************
```

	CHIP SIGNALS		TRANSITION BOARD		USED IN:
Z8530 on the MVME117	---> TxD	.....	TxD	pin 2	
	<--- RxD	.....	RxD	pin 3	
	---> RTS	.....	RTS	pin 4	
	<--- CTS	.....	CTS	pin 5	GETSTAT
	<--- DCD	.....	DSR	pin 6	
		.....	DCD	pin 8	
		.....	DTR	pin 20 +12V	

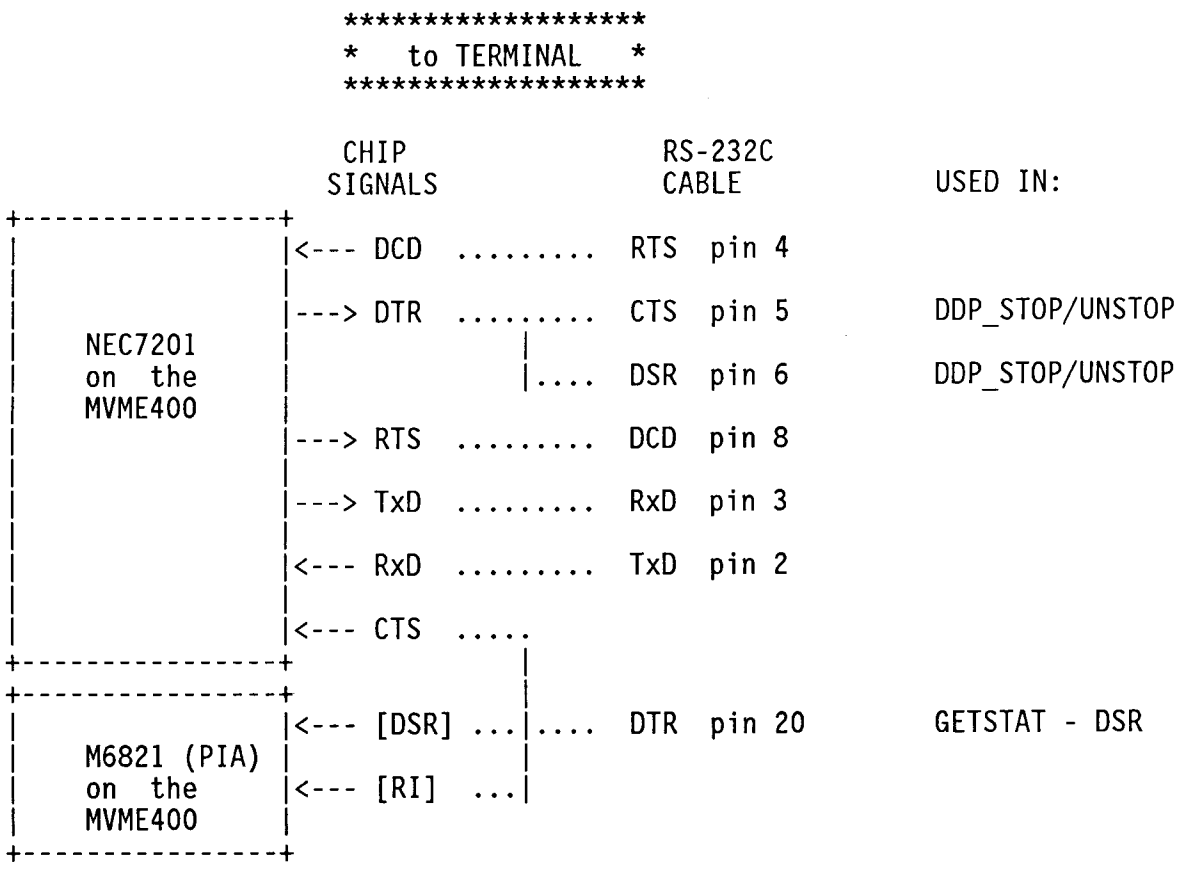
D.1.4 MFPRV/MK68901 on the MVME120



D.1.5 MPCCDRV/R68560 on the MVME050

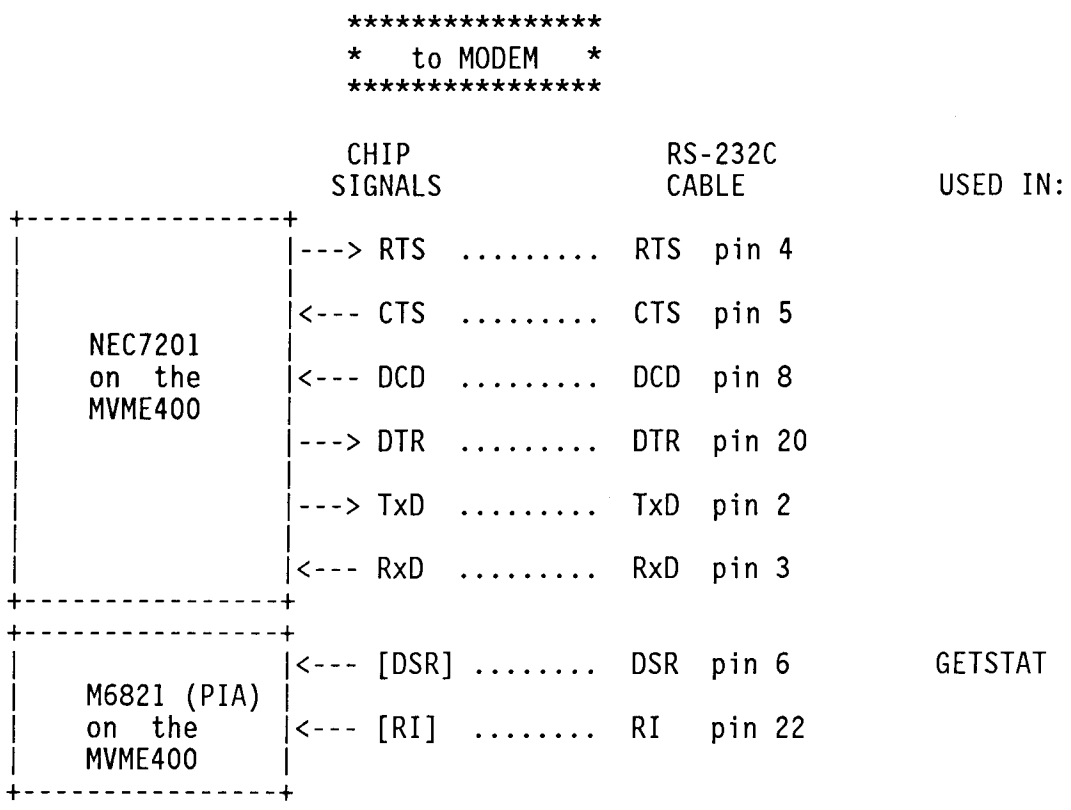
```
                    ********************
                    *   to TERMINAL    *
                    ********************

                     CHIP                RS-232C
                    SIGNALS              CABLE              USED IN:
+----------------+
|                |<--- DSR  .........  RTS  pin 4
|                |
|                |---> DTR  .........  CTS  pin 5          DDP_STOP/UNSTOP
|    R68560      |
|    on the      |---> RTS  .........  DSR  pin 6          DDP_STOP/UNSTOP
|    MVME050     |
|                |<--- DCD -- 0v       DCD  pin 8  +12V
|                |
|                |<--- CTS  .........  DTR  pin 20         GETSTAT
|                |
|                |---> TxD  .........  RxD  pin 3
|                |
|                |<--- RxD  .........  TxD  pin 2
+----------------+


                    ****************
                    *   to MODEM   *
                    ****************

                     CHIP                RS-232C
                    SIGNALS              CABLE              USED IN:
+----------------+
|                |---> RTS  .........  RTS  pin 4
|                |
|                |<--- CTS  .........  CTS  pin 5          GETSTAT
|    R68560      |
|    on the      |<--- DSR  .........  DSR  pin 6
|    MVME050     |
|                |<--- DCD  .........  DCD  pin 8
|                |
|                |---> DTR  .........  DTR  pin 20
|                |
|                |---> TxD  .........  TxD  pin 2
|                |
|                |<--- RxD  .........  RxD  pin 3
+----------------+
```

D.1.6 MPSCDRV/NEC7201 on the MVME400



NOTE: For Port 1 to talk to a TERMINAL the board is jumpered as:

J14 is open.
J15 is shorted.
J16 1-3 and 2-4 are shorted (for hardware handshaking).



Note: For Port 2 to talk to a MODEM the board is jumpered as:

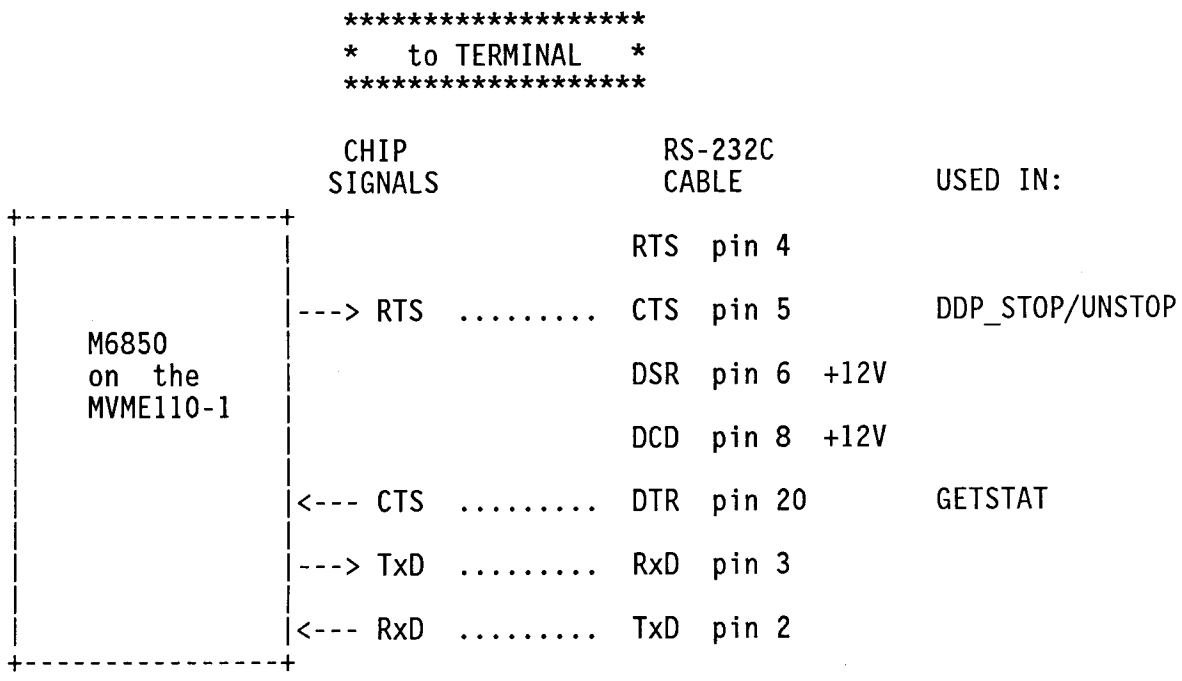
J8 is shorted.
J9 is open.
J7 5-7 and 6-8 are shorted.

D.1.7 MPSCDRV/NEC7201 on the M68KVM02/M68KVM03

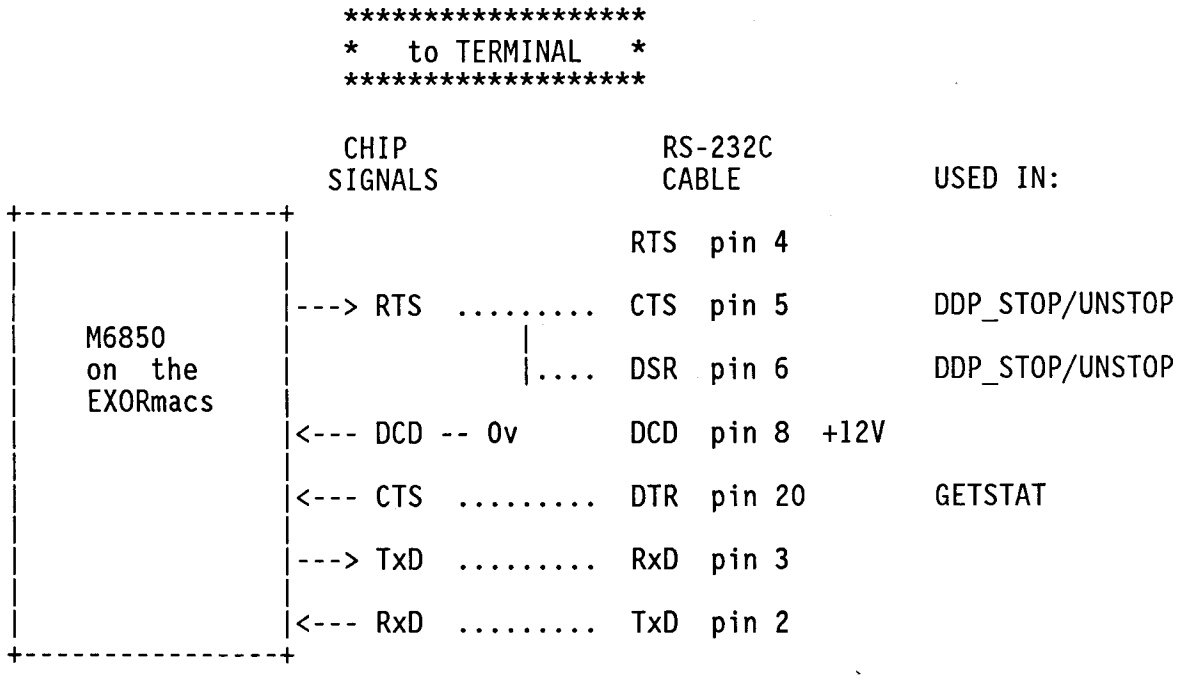
```
                    ********************
                    *   to TERMINAL    *
                    ********************

                     CHIP                RS-232C
                    SIGNALS              CABLE          USED IN:
+-----------------+
|                 |<--- DCD  .........  RTS  pin 4
|                 |
|                 |---> DTR  .........  CTS  pin 5      DDP_STOP/UNSTOP
|    NEC7201      |                |
|    on  the      |                |....  DSR  pin 6    DDP_STOP/UNSTOP
|    M68KVM02/    |
|    M68KVM03     |---> RTS  .........  DCD  pin 8
|                 |
|                 |<--- CTS  .........  DTR  pin 20     ALWAYS "READY"
|                 |
|                 |---> TxD  .........  RxD  pin 3
|                 |
|                 |<--- RxD  .........  TxD  pin 2
+-----------------+


                    ****************
                    *   to MODEM   *
                    ****************

                     CHIP                RS-232C
                    SIGNALS              CABLE          USED IN:
+-----------------+
|                 |---> RTS  .........  RTS  pin 4
|                 |
|                 |<--- CTS  .........  CTS  pin 5
|    NEC7201      |
|    on  the      |                     DSR  pin 6     ALWAYS "READY"
|    M68KVM02/    |
|    M68KVM03     |<--- DCD  .........  DCD  pin 8
|                 |
|                 |---> DTR  .........  DTR  pin 20
|                 |
|                 |---> TxD  .........  TxD  pin 2
|                 |
|                 |<--- RxD  .........  RxD  pin 3
+-----------------+
```

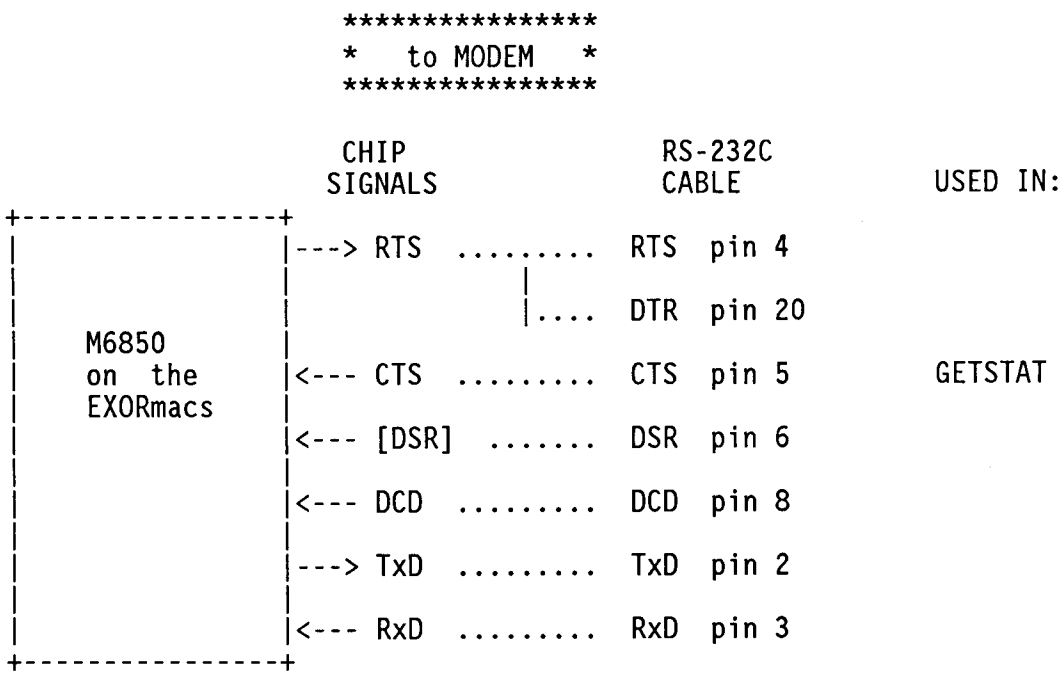
D

D.1.8 ACIADRV/M6850 on the MVME110-1



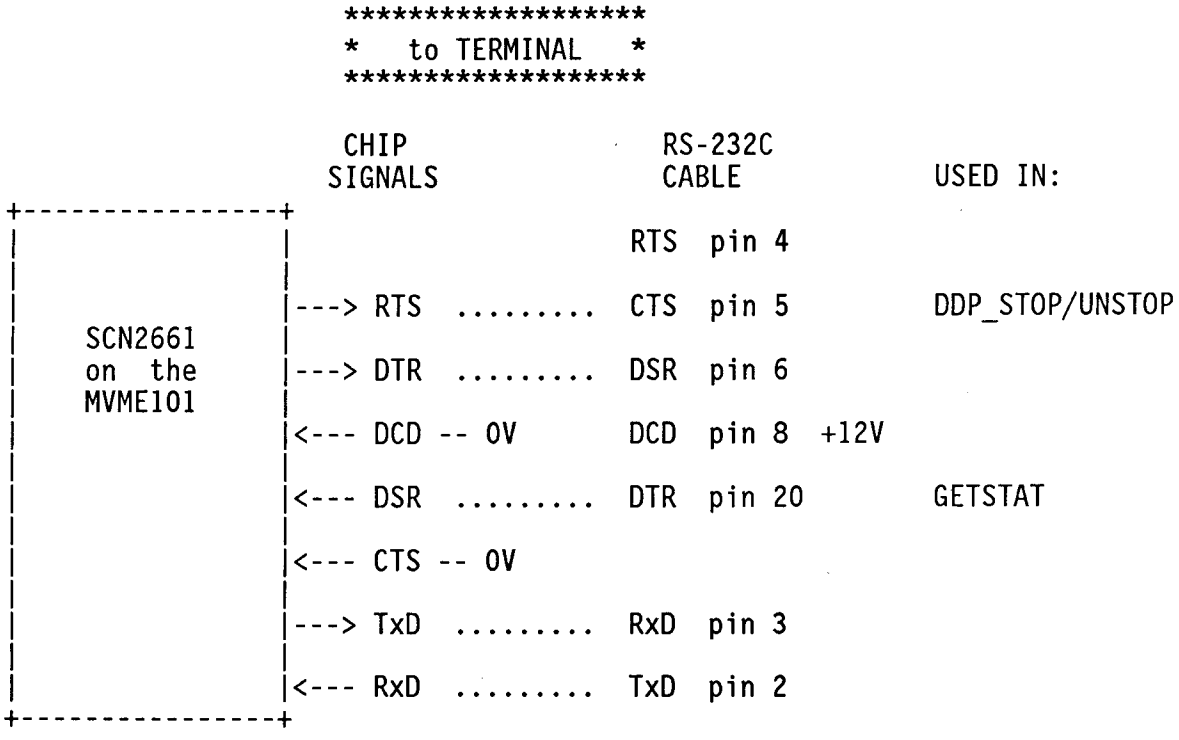
D.1.9 ACIADRV/M6850 on the EXORmacs





D

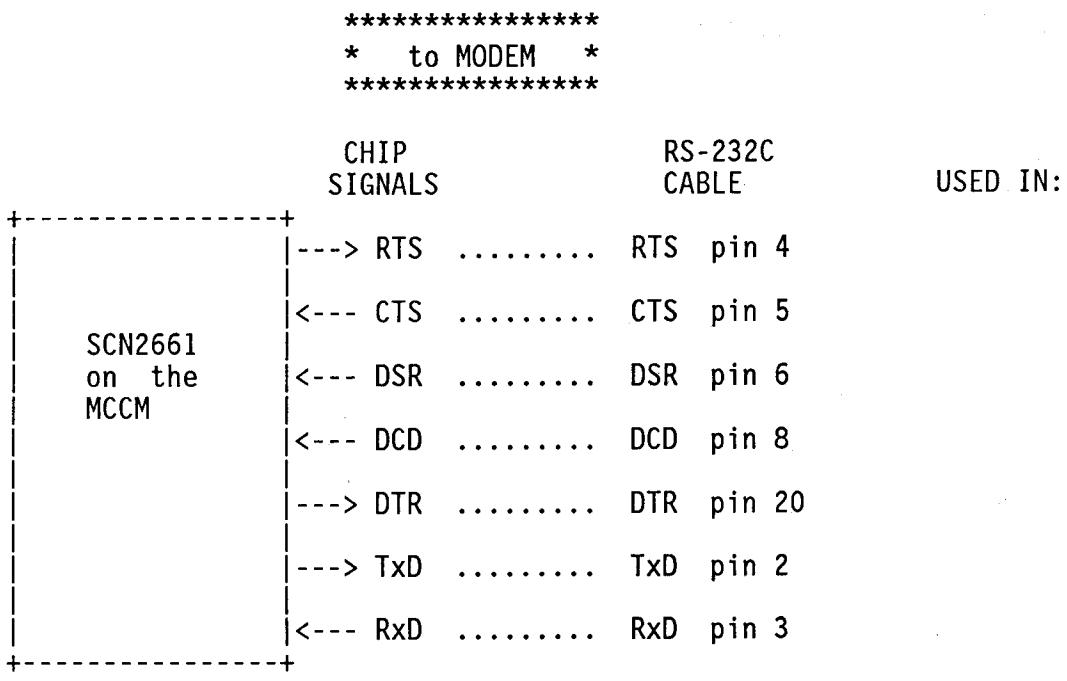
D.1.10 EPCIDRV/SCN2661 on the MVME101



```
                    ****************
                    *   to MODEM   *
                    ****************

                     CHIP                RS-232C
                    SIGNALS              CABLE              USED IN:
+----------------+
|                |---> RTS  .........  RTS  pin 4
|                |
|                |<--- CTS -- 0V       CTS  pin 5
|    SCN2661     |
|    on  the     |<--- DSR  .........  DSR  pin 6          GETSTAT
|    MVME101     |
|                |<--- DCD -- 0V       DCD  pin 8
|                |
|                |---> DTR  .........  DTR  pin 20
|                |
|                |---> TxD  .........  TxD  pin 2
|                |
|                |<--- RxD  .........  RxD  pin 3
+----------------+
```

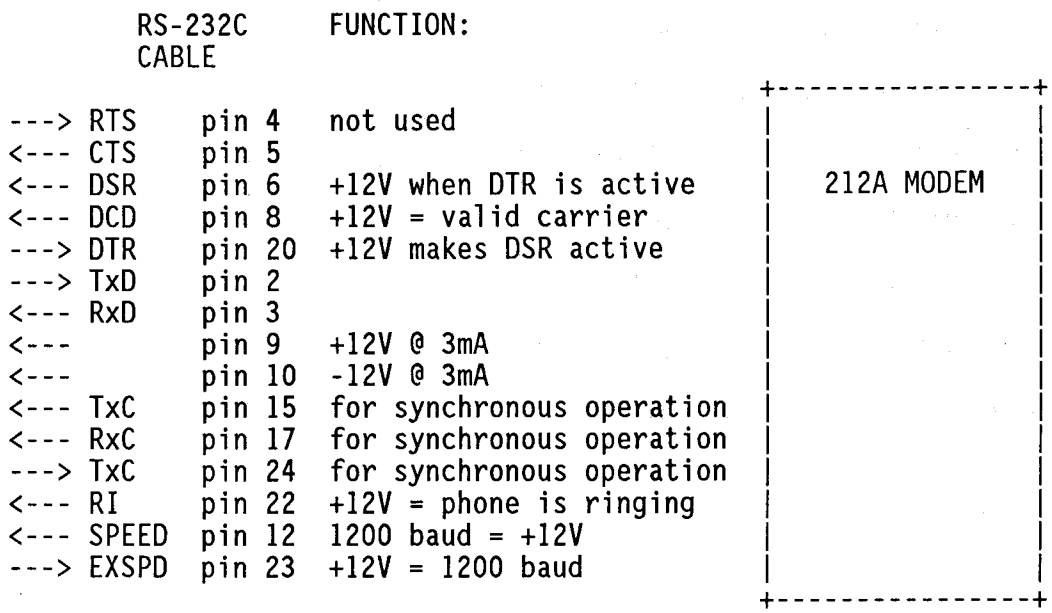
D.1.11 IPCDRV/SCN2661 on the MCCM (M68KVM30)

```
                    ********************
                    *   to TERMINAL    *
                    ********************

                     CHIP                RS-232C
                    SIGNALS              CABLE              USED IN:
+----------------+
|                |<--- DSR  .........  RTS  pin 4
|                |
|                |---> DTR  .........  CTS  pin 5
|    SCN2661     |
|    on  the     |---> RTS  .........  DSR  pin 6
|    MCCM        |
|                |<--- DCD -- 0V       DCD  pin 8  +12V
|                |
|                |<--- CTS  .........  DTR  pin 20
|                |
|                |---> TxD  .........  RxD  pin 3
|                |
|                |<--- RxD  .........  TxD  pin 2
+----------------+
```

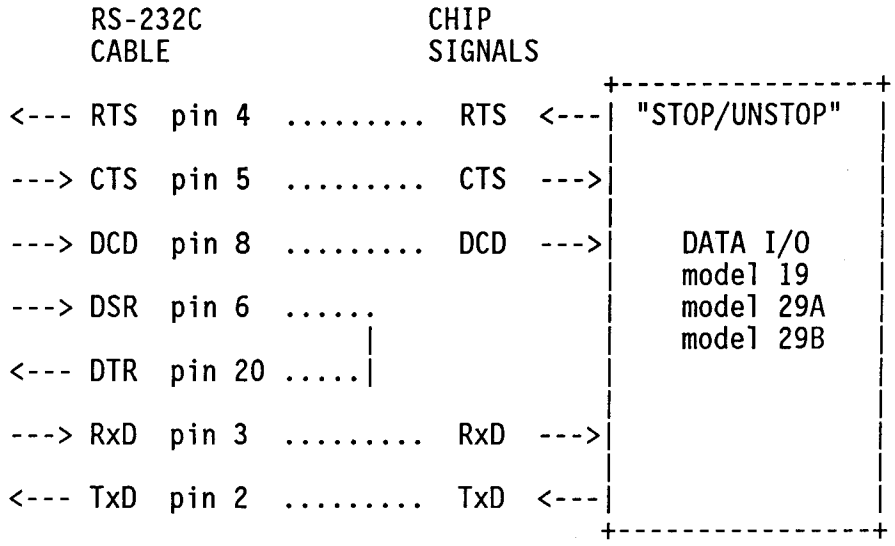


D.2 SELECTED RS-232C DEVICES

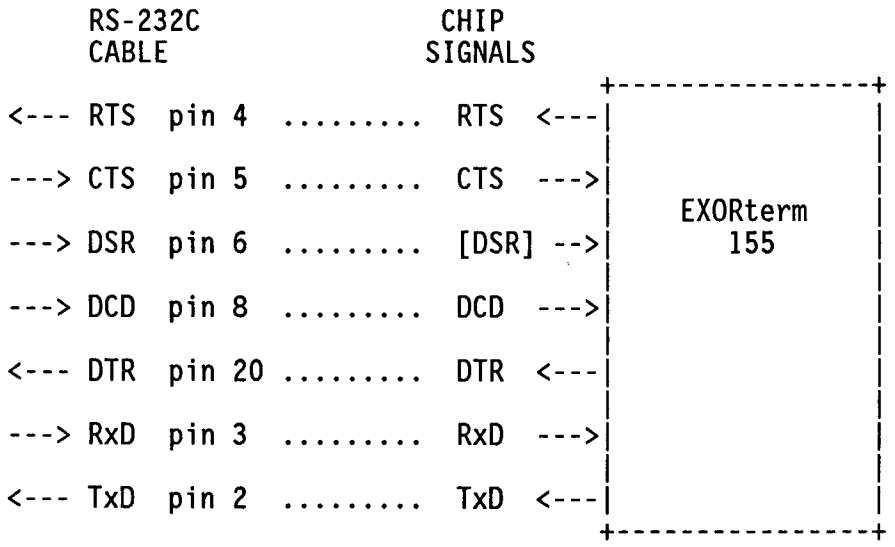
D.2.1 UDS 212A MODEM



D.2.2 DATA I/O PROM PROGRAMMER



D.2.3 EXORterm 155 TERMINAL



D.3 DEFINITION OF RS-232C SIGNALS

RS-232C PIN	SIGNAL NAME		DESCRIPTION
1		Not used.	
2	TxD	TRANSMIT DATA	Data to be transmitted is furnished on this line to the modem from the terminal.
3	RxD	RECEIVE DATA	Data on the receive line is presented to the terminal from the modem.
4	RTS	REQUEST TO SEND	RTS is supplied by the terminal to the modem when required to transmit a message.
5	CTS	CLEAR TO SEND	CTS is supplied to the terminal by the modem which indicates that it is permissible to begin transmission of a message.
6	DSR	DATA SET READY	DSR is supplied to the terminal by the modem to indicate that the modem is ready to transmit data.
7	GND	SIGNAL GROUND	
8	DCD	DATA CARRIER DETECT	Sent by the modem to the terminal to indicate that a valid carrier is being received.
15	TxC	TRANSMIT CLOCK	This line clocks output data to the modem from the terminal.
17	RxC	RECEIVE CLOCK	This line clocks input data from a terminal to a modem.
20	DTR	DATA TERMINAL READY	A signal from the terminal to the modem indicating that the terminal is ready to send or receive data.
22	RI	RING INDICATOR	RI is sent by the modem to the terminal. This line indicates to the terminal that an incoming call is present.
24	TxC	TRANSMIT CLOCK	Refer to TxC on pin 15.

D.4 SELECTED ROUTINES USED BY SERIAL DRIVERS

D.4.1 Selecting Device Ready/Unready (Flow Control)

There are two methods of flow control. First, the default method used in VERSAdos between a computer and a terminal in close proximity is to change the level of a hardware output line. The port must be set with the "USE XOFF/ON" attribute off to allow hardware handshaking.

The second method of flow control is XON/XOFF. The industry standard is to use DC1 and DC3 for XON and XOFF, respectively (DC1 = $11 and DC3 = $13). The default values for VERSAdos are any character and DC3. This method of flow control requires no adjustments of hardware output lines and is the method used with modems. The port must be set with the "USE XOFF/ON" attribute ON to allow this method of flow control.

D.4.1.1 DDP-Unstop (Device-Dependent Unstop). This routine will assert the appropriate line(s) to allow the external device to continue transmitting. This routine is called when the type-ahead buffer has been emptied enough to allow more characters to be received.

D.4.1.2 DDP-Stop (Device-Dependent Stop). This routine will negate the appropriate line(s) to stop the external device from transmitting. This routine is called when the type-ahead buffer is full so that no more characters are received.

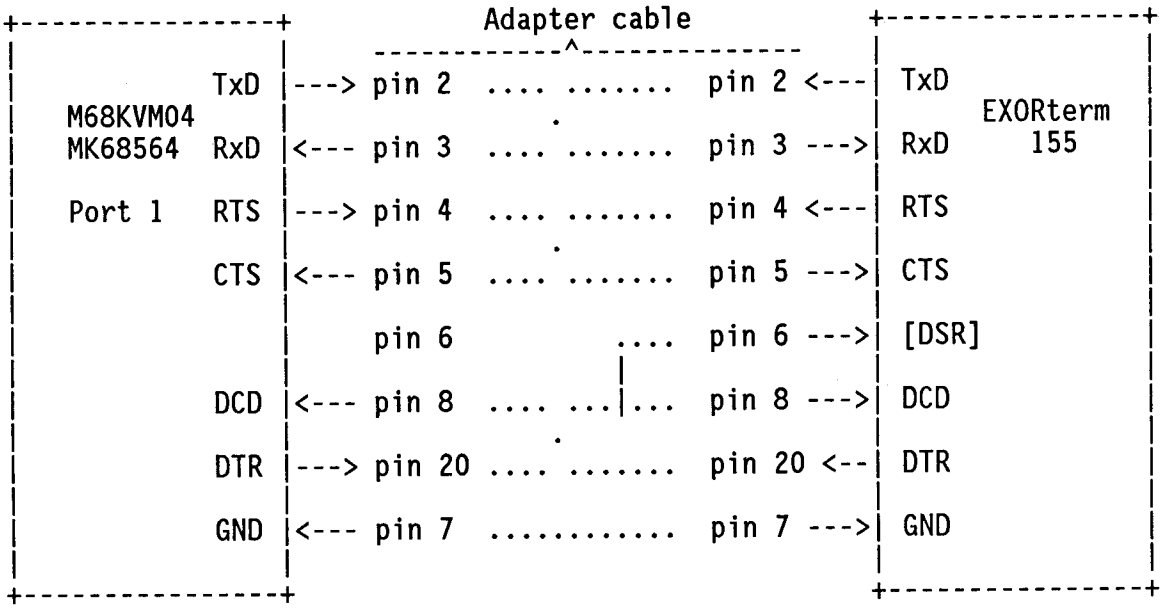
D.4.2 Detecting Device Ready/Unready (GETSTAT)

The GETSTAT routine is called to check the status of the external device. The proper signal for each driver is read and the result is returned.

D.5 CONFIGURATION EXAMPLES

D.5.1 M68KVM04 PORT 1 and an EXORterm 155 Terminal

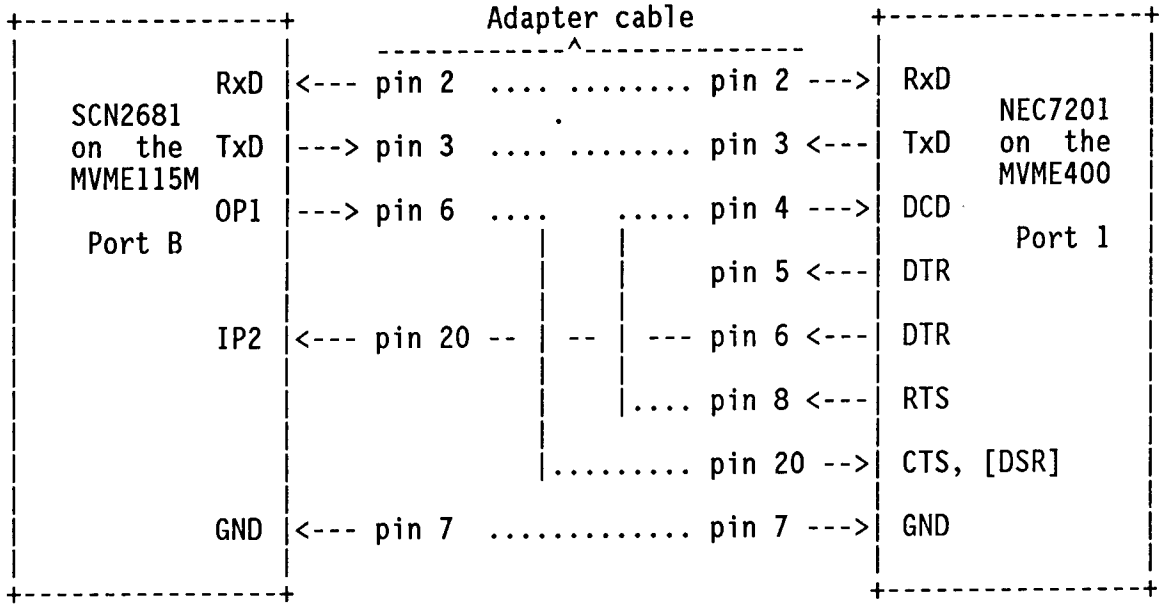
Port 1 on the M68KVM04 was designed to be interfaced with Data Communication Equipment (DCE) such as modems. It can be used with terminals, as in the following example.



D

D.5.2 MVME115M Combined with an MVME400

To connect the MVME115M to an MVME400 for an application such as UPLOADS, use the configuration:



The MVME115M is jumpered as a TERMINAL:

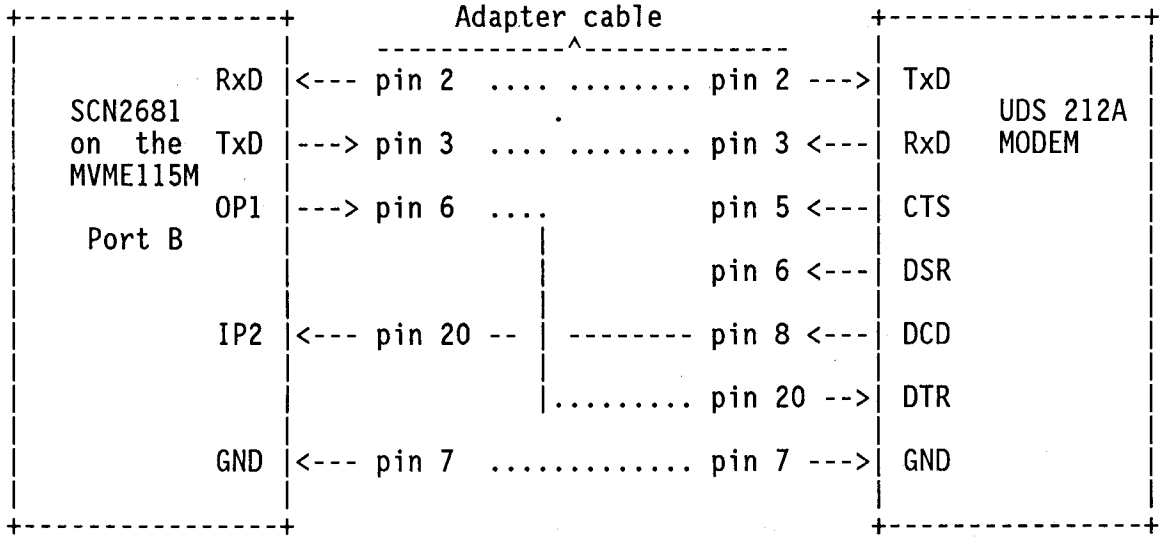
J121 3-4
J152 1-2, 3-4
J128 1-3

The MVME400 is jumpered as a TERMINAL:

J14 - all open
J15 - all shorted
J16 - 1-3, 2-4

D.5.3 MVME115M and a UDS 212A MODEM

To connect the MVME115M to a MODEM, such as a UDS 212A, use the configuration:

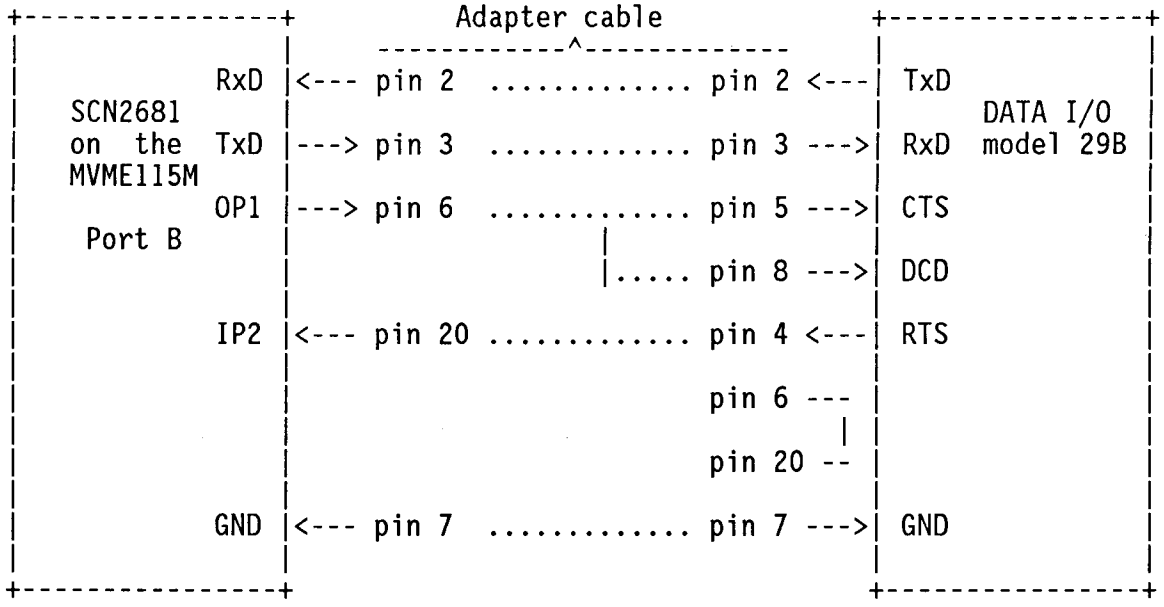


The MVME115M is jumpered as a TERMINAL:

J121 3-4
J152 1-2, 3-4
J128 1-3

D.5.4 MVME115M and a Data I/O PROM Programmer

To connect the MVME115M to a PROM programmer, such as the DATA I/O model 29B, use the configuration:



The MVME115M is jumpered as a TERMINAL:

J121 3-4
J152 1-2, 3-4
J128 1-3

D.6 MISCELLANEOUS NOTES

A SPACE = +3 to +15 volts.

A MARK = -3 to -15 volts.

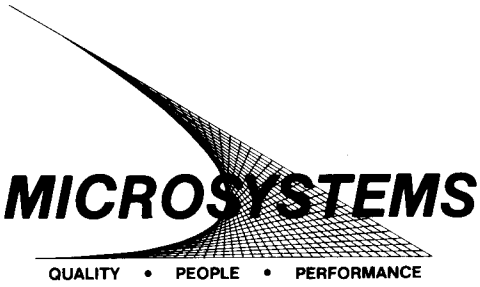
A TERMINAL will drive pins 4 and 20 and monitor pins 5, 6 and 8.

A MODEM will drive pins 5, 6 and 8 and will monitor pins 4 and 20.

D

THIS PAGE INTENTIONALLY LEFT BLANK.

SUGGESTION/PROBLEM REPORT



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.

18401 PRINTED IN USA (5/85) MESSENGER 7500