



[Home](#)



OS-9[®] for 68K Processors MVME Board Guide

Version 8.8



RadiSys.
THE POWER OF WE

www.radisys.com

Revision C • January 2008

Copyright and publication information

This manual reflects version 8.8 of Microware OS-9 for 68K. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

January 2008
Copyright ©2008 by RadiSys Corporation
All rights reserved.

EPC and RadiSys are registered trademarks of RadiSys Corporation. ASM, Brahma, DAI, DAQ, MultiPro, SAIB, Spirit, and ValuePro are trademarks of RadiSys Corporation.

DAVID, MAUI, OS-9, OS-9000, and SoftStax are registered trademarks of RadiSys Corporation. FasTrak, Hawk, and UpLink are trademarks of RadiSys Corporation.

† All other trademarks, registered trademarks, service marks, and trade names are the property of their respective owners.

Table of Contents

Chapter 1: MVME147 Reference **7**

- 8 Overview of the MVME147 CPU Module
- 11 ROM Configuration and Organization
- 12 Installing the MVME147 CPU Module
- 15 Configuring 147Bug to Boot RomBug
- 15 To configure 147Bug to boot RomBug
- 18 Board Specific OS-9 Modules

Chapter 2: MVME162 Reference **19**

- 20 MVME162 Quick Reference
- 21 Overview of the Original and FX MVME162 CPU Models
- 28 Overview of the MVME162 LX CPU Models
- 38 ROM Configuration and Organization
- 40 Board Specific OS-9 Modules
- 41 Installing the Original or FX Series MVME162 CPU Module (EPROM Method)
- 43 Installing the MVME162LX CPU Module (EPROM method)
- 44 Installing the MVME162 CPU Module (Flash programming method)
- 46 Configuring 162Bug to Boot RomBug
- 46 Configuring 162Bug to Boot RomBug
- 50 162Bug's NIOT configuration
- 51 Download Boot Image using NIOP Command
- 52 Program OS-9 ROM Image into Flash Memory

Chapter 3: MVME167 Reference **53**

- 54 MVME167 Quick Reference

55	Overview of the MVME167 CPU Module
58	ROM Configuration and Organization
59	Board Specific OS-9 Modules
60	Installing the MVME167 CPU Module
63	Configuring 167Bug to Boot RomBug
63	Configuring 167Bug to Boot RomBug

Chapter 4: MVME172 Reference

67

68	MVME172 Quick Reference
69	Overview of the MVME172 CPU Models
75	ROM Configuration and Organization
76	Board Specific OS-9 Modules
77	Installing the MVME172 CPU Module (EPROM Method)
80	Installing the MVME172 CPU Module (Flash programming method)
82	Configuring 172Bug to Boot RomBug
82	Configuring 172Bug to Boot RomBug

Chapter 5: MVME177 Reference

87

88	MVME177 Quick Reference
89	Overview of the MVME177 CPU Module
92	ROM Configuration and Organization
94	Board Specific OS-9 Modules
95	Installing the MVME177 CPU Module
99	Installing RomBug into Flash
99	Downloading RomBug from a Host over an Ethernet Network
101	Configuring 177Bug to Boot RomBug
101	Configuring 177Bug to Boot RomBug

Chapter 6: Peripheral and Transition Module Configurations

105

106	Overview of the MVME712 Transition Module
-----	---

108	Configuring the MVME712 Transition Module	
110	Configuring the MVME050 System Controller Module for I/O Operations	
113	Configuring the MVME701 I/O Transition Module	
115		

Appendix A: Downloading RomBug from a Host over a Serial Interface 117

118	Downloading RomBug from a Host over a Serial Interface	
-----	--	--

Chapter 1: MVME147 Reference

This section contains information about various MVME147 models. It contains the following sections:

- **Overview of the MVME147 CPU Module**
- **ROM Configuration and Organization**
- **Installing the MVME147 CPU Module**
- **Configuring 147Bug to Boot RomBug**
- **Board Specific OS-9 Modules**



Overview of the MVME147 CPU Module

The MVME147 CPU module provides:

- 68030 processor running at either 16MHz (SRF model), 20MHz, 25MHz (Sx-1 models) or 32MHz (Sx-2 models), with on-chip Paged Memory Management Unit
- 4, 8 (SA-x models), 16 (SB-x models), or 32M (SC-x models) of on-board DRAM
- Four RS-232C Serial Ports
- Centronics Parallel Printer Port
- On-board SCSI Interface
- Ethernet Transceiver Interface (except for -RF/-SRF models)
- Time-of-day clock/calendar with battery backup
- 2Kx8 Battery backed CMOS RAM (NVRAM)
The OS-9 area of NVRAM is the first 256 bytes of the user area beginning at address FFFE0000.
- Four JEDEC EPROM sockets organized as two banks of 16-bit wide (high and low byte) memory:
 - MVME147:
 - Primary ROM Bank:** FF800000
High Byte U1 / Low Byte U2
 - Secondary ROM Bank:** FFA00000
High Byte U16 / Low Byte U18
 - MVME147S:
 - Primary ROM Bank:** FF800000
High Byte U22 / Low Byte U30
 - Secondary ROM Bank:** FFA00000
High Byte U1 / Low Byte U15
- 68882 Floating Point Coprocessor

Figure 1-1 MVME147 Jumper Block Locations

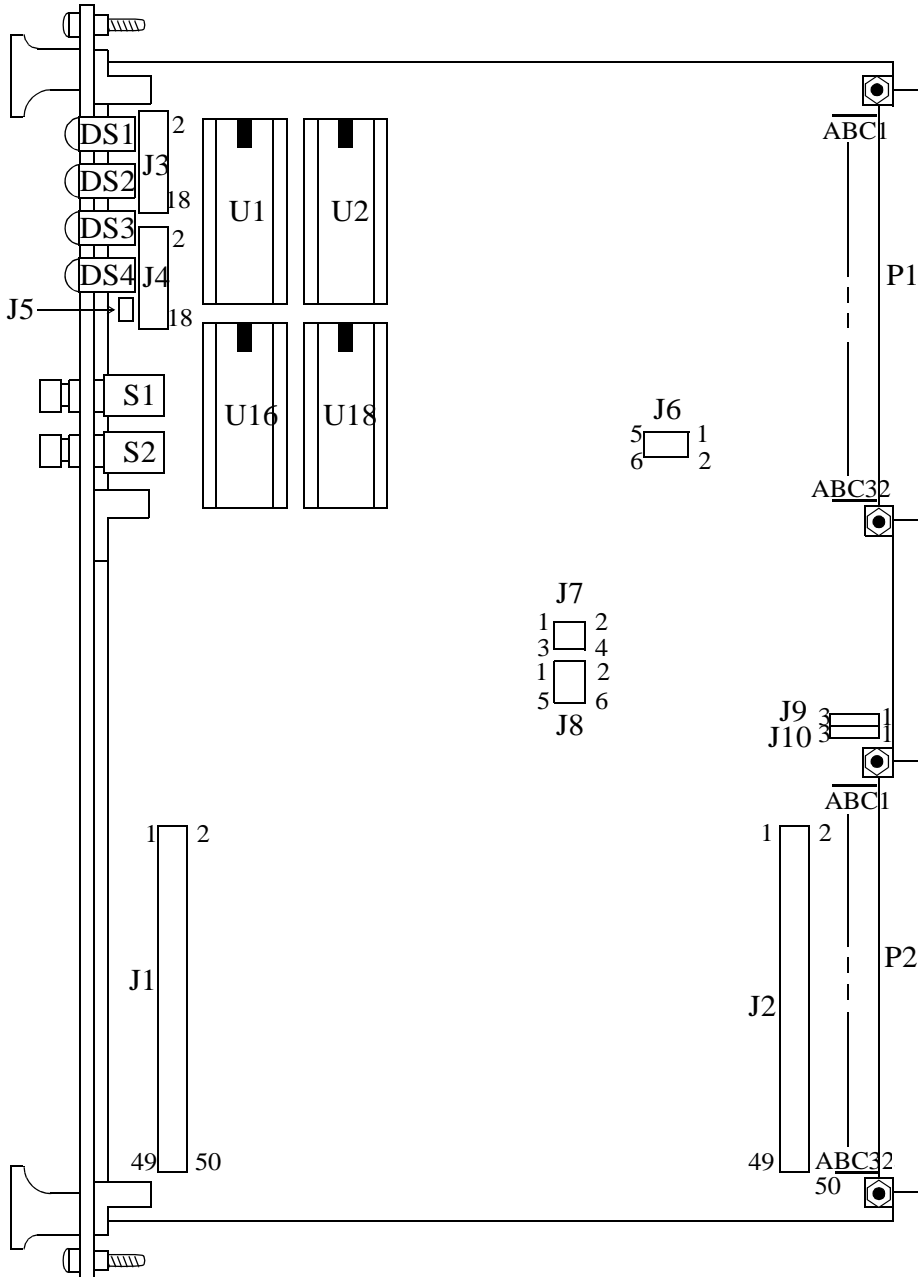
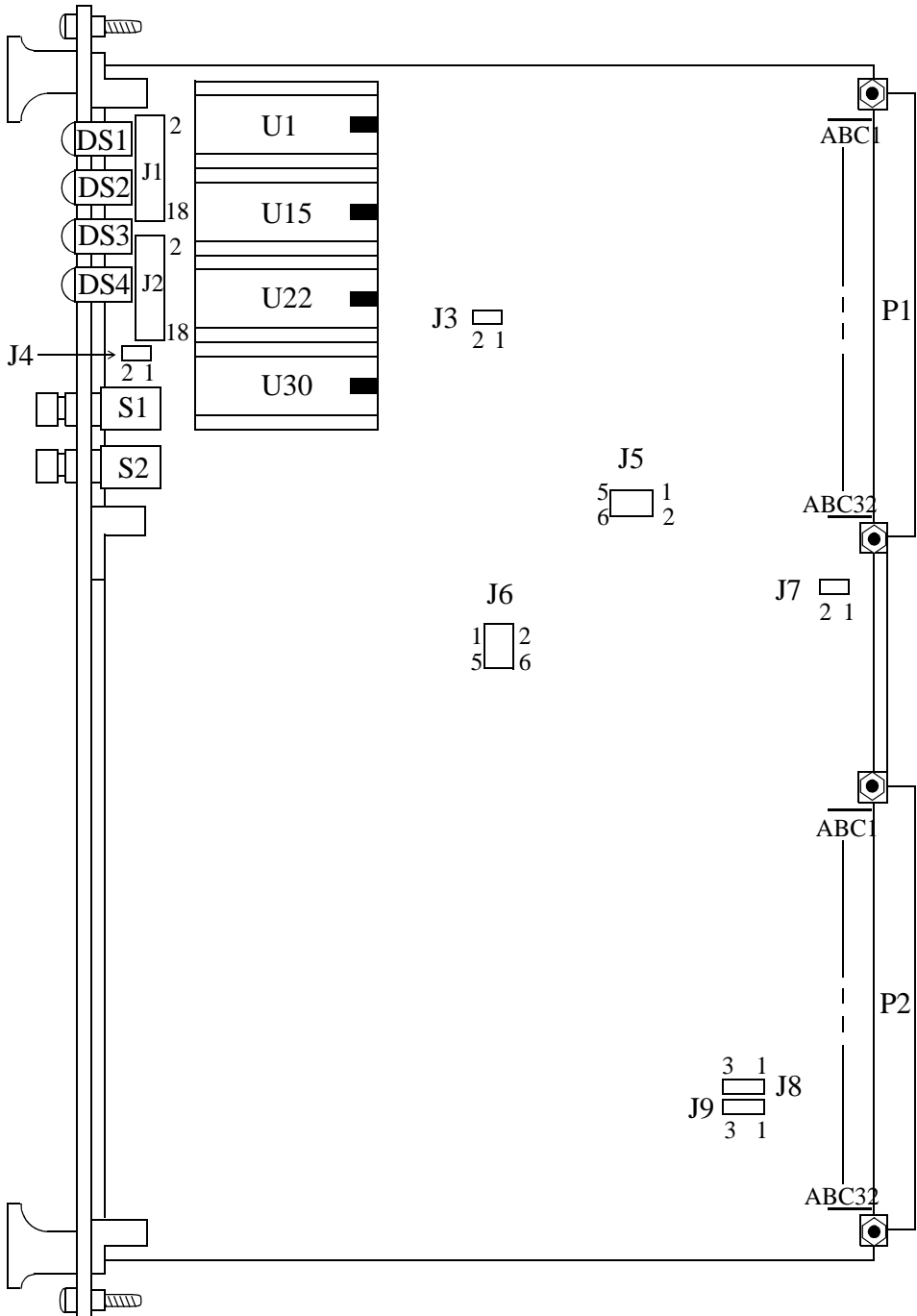


Figure 1-2 MVME 147S Jumper Block Location

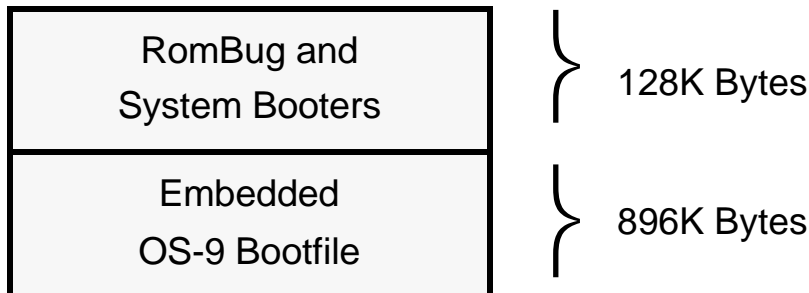


ROM Configuration and Organization

There are two possible ROM configurations for the MVME147. As the system installer, you must choose which ROM configuration to adopt, depending on your system application.

- The first retains the 147Bug ROMs in the primary bank (for ease of hardware diagnostic testing) and adds the OS-9 RomBug EPROMs in the secondary bank. In this configuration, 147Bug gets control on a reset and “boots” RomBug in the secondary bank (see the [Configuring 147Bug to Boot RomBug](#) section).
- The second configuration replaces the 147Bug EPROMs with the OS-9 RomBug EPROMs in the primary bank. In this configuration, RomBug always gets control on a reset.

Figure 1-3 MVME 147 RomBug EPROM Address Space



Installing the MVME147 CPU Module

Step 1. Remove power from the system, and eject the CPU module.

Step 2. Install the EPROMs on the MVME147 module.

For the MVME147 (not 'S') models, the primary bank of sockets are U1 (high or even byte) and U2 (low or odd byte). The secondary bank of sockets are U16 (high byte) and U18 (low byte).

For the MVME147S models, the primary bank of sockets are U22 (high or even byte) and U30 (low or odd byte). The secondary bank of sockets are U1 (high byte) and U15 (low byte).

Insert the RomBug EPROMs into the appropriate bank(s) as desired, keeping watch for the correct part (high or low) in the correct socket of the bank. The notches in the EPROMs should face the top edge of the card (for non-"S"-models) or the backplane (for "S"-models).

Step 3. Examine and change the jumpers on the MVME147 module.

In general, the jumper settings for the MVME147 module should remain the same as the factory default settings (or configured specifically to your needs), with the exception of the ROM size jumpers for the bank(s) containing RomBug EPROMs. If the jumpers are set for a size too small, "Boot from ROM" will not operate correctly, if RomBug even operates at all. The jumper configurations are identical across all models of the MVME147 and across banks.

Figure 1-4 EPROM Size Select, Banks 1 and 2

Banks 1 (J3 for 147; J2 for 147S) and 2 (J4 for 147; J1 for 147S):

This jumper selects the size of the EPROMs installed for the Bank used by the OS-9 RomBug* EPROMs. For 27C040 type EPROMs (256K x 8), connect: 2 to 4, 3 to 5, 6 to 8, 7 to 9, 13 to 15, and 14 to 16.

Jumper Blocks					
2-4	6-8	10	12	14-16	18
1-3	5	7-9	11	13-15	17



Note

The MotBug EPROMs will use a different jumper setting because they are not 27C040 EPROMs.

Figure 1-5 System Controller Enable Header (J5 for 147; J3 for 147S)

J5 for 147; J3 for 147S

This jumper indicates whether the MVME147 is the system controller. If installed, the MVME147 is the system controller.

Jumper Block J5/J3	
1	2

Figure 1-6 Serial Port 4 Clock Configuration Select

(J9, J10):

J9 and J10 (along with J15 of the MVME712M Breakout Module) configure the serial port 4 use of the clock signals.

Connect **J9:** 2 to 3 Receives RTXC4
J10: 1 to 2 Drives TRXC4

Jumper Block J9	
1	2-3

Jumper Block J10	
1-2	3

Step 4. Replace the CPU module in the system.

Once the EPROM and jumper installation is done, carefully read the ***MVME147 User Manual*** (supplied by Motorola) to complete the hardware installation steps required for the P2 Adaptor Card (if used) and the MVME712M Transition Module. These instructions also detail the various options available for cabling SCSI devices to the MVME147 module.

Jumper installation details for the MVME712M module are supplied in **Chapter 6** of the ***OS-9 for 68K Processors MVME Board Guide***. However, carefully read the Motorola supplied documentation to ensure that all hardware cabling requirements are correctly satisfied.

Configuring 147Bug to Boot RomBug

Motorola's 147Bug, which resides in EPROM, can be configured to automatically boot RomBug using its ROMboot feature. This enables coldstarting from the 147Bug in order to keep the diagnostics handy, then automatically transferring control to RomBug to bring up OS-9. However, this method is much slower than coldstarting directly into RomBug.

To configure 147Bug to boot RomBug

Step 1. Ensure that 147Bug EPROMs are installed in the primary bank; that RomBug EPROMs are installed in the secondary bank; and that the ROM size jumper settings for each bank are correctly set.

Step 2. Reinstall the CPU module and reset the system to bring up the 147-Bug> prompt:

```
Copyright Motorola Inc. 1988, 1989, 1990, 1991, 1992 All
Rights Reserved
```

```
MVME147 Monitor/Debugger Release 2.43 - 6/30/92
CPU running at 25 MHz
```

```
FPC passed test
MMU passed test
```

```
COLD Start
```

```
Onboard RAM start = $00000000, stop = $003FFFFFFF
```

```
147-Bug>
```

Step 3. Enter the RB command at the 147Bug> prompt entering the following values at the configuration field prompt:

```
147-Bug> RB <return>
Boot at Power-up only or any board Reset [P,R] = P? R
<return>
Enable search of VMEbus [Y,N] = N? <return>
```

```

Boot direct address = $FF800000? FFA00400 <return>
ROM boot enabled
147-Bug>

```

Step 4. Enter the RESET command to force 147Bug to initiate the ROMboot sequence:

```

147Bug> RESET<return>
Reset Local SCSI Bus [Y,N] N? <return>
Automatic reset of known SCSI Buses on RESET [Y,N] = N?
<return>
Cold/Warm Reset flag [C,W] = C? <return>

```

```
Execute Soft Reset [Y,N] N? Y<return>
```

The system will reset twice after responding to the last prompt:

```

Copyright Motorola Inc. 1988, 1989, 1990, 1991, 1992 All Rights Reserved

MVME147 Monitor/Debugger Release 2.43 - 6/30/92
CPU running at 25 MHz

FPC passed test
MMU passed test

COLD Start

Onboard RAM start = $00000000, stop = $003FFFFFF

147-Bug>G FFA00416
Effective address: FFA00416}

Copyright Motorola Inc. 1988, 1989, 1990, 1991, 1992 All Rights Reserved

MVME147 Monitor/Debugger Release 2.43 - 6/30/92
CPU running at 25 MHz

FPC passed test
MMU passed test

COLD Start

Onboard RAM start = $00000000, stop = $003FFFFFF

147-Bug>G FFA00416
Effective address: FFA00416
OS-9/68K System Bootstrap

<Called>
Searching special memory list for symbol modules...

dn: 000000FF 00002000 00000000 00000000 00000000 00000001 FFFF0000 000069F0
an: FFA00A6E FFA00500 FFA40000 FFA40000 00007A00 00000400 00000000 000069F0
pc: FFA00970 sr:2708 (--SI-7-N---)t:OFF msp:00005C18 usp:00000000 ^isp^
0xFFA00970 >43FAFBDA lea.l 0xFFA0054C(pc),a1
RomBug:

```




Note

The ROMboot feature is not as robust on the MVME147 as it is on the MVME162 and MVME167. The only way to interrupt or disable a ROMboot sequence, once it has been enabled, is to force an ABORT-RESET sequence with the front panel switches. This is done by pressing both ABORT and RESET and letting go of RESET while still depressing ABORT until the 147-Bug> prompt appears. This also disables the ROMboot feature on subsequent resets. To reenble the ROMboot feature, reenter the RB command/dialogue, followed by the RESET command/dialogue as shown above.



Note

Since the ABORT-RESET sequence disables an automatic ROMboot sequence, it cannot be used alone to force a reconfiguration session when 147Bug is in the primary ROM Bank. The only way to force a reconfiguration session is to use the ABORT-RESET sequence to enter 147Bug, manually enter a value in the OS-9 area of NVRAM to invalidate the checksum of that area, re-enable ROM Booting and reset the CPU

The following 147Bug command will invalidate the NVRAM checksum in the same way as the ABORT-RESET sequence would if RomBug were in the primary ROM bank:

```
147-Bug> MW fffe0004 01;B<return>
Effective address: FFFE0004
Effective data    : 01
147-Bug> RB<return>
...
```

Board Specific OS-9 Modules

The software environment for the MVME147 is built in its `PORTS` directory structure which can be found at `/h0/MWOS/OS9/68030/PORTS/MVME147`. All descriptors, init modules, and bootfiles are configured and built in this directory and its subdirectories.

The board-specific drivers that are available for the MVME147 are:

<code>tk147</code>	system ticker
<code>rtc147</code>	real-time clock driver
<code>sc147</code>	SCF/serial port driver
<code>scp147</code>	SCF/parallel port driver
<code>scsi147</code>	SCSI low-level driver
<code>sp147</code>	SoftStax® Ethernet driver

Other system modules available for use on the board which reside in common directories are:

<code>cache030</code>	cache control module
<code>ssm851</code>	SSM for the 68030 PMMU
<code>fpu</code>	floating point emulation for CPU modules without 68882 FPU chip



Note

When starting the SoftStax/LAN Communications Pak TCP/IP networking on the MVME147 using the `startspf.ndbmod` example shell script, the device name should be changed from `spie0` to `sple0` as shown below:

```
ndbmod interface add enet0 address 192.168.0.5 netmask 255.255.255.0
binding /sple0/enet
```

Chapter 2: MVME162 Reference

This section contains information about the MVME162 original, FX, and LX models. It contains the following sections:

- **MVME162 Quick Reference**
- **Overview of the Original and FX MVME162 CPU Models**
- **Overview of the MVME162 LX CPU Models**
- **ROM Configuration and Organization**
- **Board Specific OS-9 Modules**
- **Installing the Original or FX Series MVME162 CPU Module (EPROM Method)**
- **Installing the MVME162LX CPU Module (EPROM method)**
- **Installing the MVME162 CPU Module (Flash programming method)**
- **Configuring 162Bug to Boot RomBug**



For More Information

This chapter details the jumper and processor locations for the MVME162 original, FX, and LX models. For information on the Petra model, refer to the ***Motorola MVME VME Embedded Controller Installation and Use Guide*** shipped with your hardware.



MVME162 Quick Reference

Ports Directory: <mwos>/OS9/68040/PORTS/MVME162

Console Port Location: Faceplate, connector labeled “console”

OS-9 EPROM Size: 1 megabyte X 8 bits

Original and FX models:

- On some models PROM size may be restricted to 512K and the OS-9 Rombug image must be programmed into Flash.
- A user-programmed EPROM may be installed on models supporting 1Mbit EPROM.

LX models:

- OS-9 ROM image can be programmed into Flash

ROM Booting Search Order:

FF800400:FF808400

FFA00400:FFA08400

BootP Ethernet booting device: ie

SPF Ethernet descriptor name: spie0

CMDS Build sequence:

<mwos>/OS9/68000/CMDS

<mwos>/OS9/68020/CMDS

<mwos>/OS9/68040/CMDS

<mwos>/OS9/68040/PORTS/MVME162/CMDS

Overview of the Original and FX MVME162 CPU Models

The original and FX MVME162 models provide:

- 68040 or 68LC040 processor running at 25MHz or 32 MHz with on-chip 8K cache, MMU, and FPU (68040 only)
- 1M, 4M, 8M, or 16M Bytes of Dynamic RAM
- 512K Bytes of battery-backed Static RAM
- 2 RS-232 serial ports
- On-board SCSI I/O processor (Optional)
- On-board Ethernet processor (Optional)
- Time-of-day clock/calendar with battery backup
- 8Kx8 battery backed CMOS RAM (NVRAM)
The OS-9 for 68K area of NVRAM is the first 256 bytes of the user area beginning at FFFC0000.
- 1M Bytes of flash memory as primary ROM bank (at FF800000) or secondary ROM bank (at FFA00000, jumper selectable)
- One 32-pin PLCC PROM socket as primary or secondary ROM bank (jumper selectable - alternates with flash) Size 4/8 M Bits. Early board may be restricted to 4M Bits.
- 4 Industry Pack (IP) slots (A - D). The default IP setup by RomBug is detailed in [Table 2-1](#). If a particular application requires a different setup, RomBug can be rebuilt with a customized `initext` module to fit the application's requirements.

Table 2-1 MVME162 Original and FX Model IP Slots

IP Slot	Memory Base	Interrupt Level
A	C0000000	5
B	C0800000	4
C	C1000000	3
D	C1800000	2

All slots are set for:

Memory Size	8M Bytes
Memory Width	16 Bits
Interrupts	Level sensitive, Normal Polarity
Recovery Time	0 microseconds

The original models differ by the amount of RAM memory installed, the type of the processor, and support for SCSI and/or Ethernet on the board. Original series boards operate at 25 MHz while the FX boards operate at 32 MHz. A representative list of model numbers are listed in [Table 2-2 MVME162 Original and FX Models and Processor Information](#).

Table 2-2 MVME162 Original and FX Models and Processor Information

Model	Processor	DRAM Size and Type	I/O Options
MVME162-001	LC040	1MB Parity	
MVME162-002	68040	1MB Parity	
MVME162-003	LC040	1MB Parity	No VME interface
MVME162-010(A)	LC040	4MB Parity	
MVME162-011(A)	LC040	4MB Parity	SCSI
MVME162-012(A)	LC040	4MB Parity	Enet
MVME162-013(A)	LC040	4MB Parity	Enet,SCSI
MVME162-014(A)	LC040	4MB Parity	No VME interface
MVME162-020(A)	040	4MB Parity	
MVME162-021(A)	040	4MB Parity	SCSI
MVME162-022(A)	040	4MB Parity	Enet
MVME162-023(A)	040	4MB Parity	Enet,SCSI
MVME162-026(A)	040	4MB Parity	Enet, No VME interface
MVME162-030	LC040	8MB Parity	
MVME162-031	LC040	8MB Parity	SCSI
MVME162-032	LC040	8MB Parity	Enet
MVME162-033	LC040	8MB Parity	Enet,SCSI
MVME162-040	040	8MB Parity	
MVME162-041	040	8MB Parity	SCSI
MVME162-042	040	8MB Parity	Enet
MVME162-043	040	8MB Parity	Enet, SCSI
MVME162-510	040	4MB No Parity	

**Table 2-2 MVME162 Original and FX Models and Processor Information
(continued)**

Model	Processor	DRAM Size and Type	I/O Options
MVME162-511	040	4MB No Parity	SCSI
MVME162-512	040	4MB No Parity	Enet
MVME162-513	040	4MB No Parity	Enet, SCSI
MVME162-520	040	8MB No Parity	
MVME162-521	040	8MB No Parity	SCSI
MVME162-522	040	8MB No Parity	Enet
MVME162-523	040	8MB No Parity	Enet, SCSI
MVME162-530	040	16MB No Parity	
MVME162-531	040	16MB No Parity	SCSI
MVME162-532	040	16MB No Parity	Enet
MVME162-533	040	16MB No Parity	Enet, SCSI

Figure 2-1 MVME162 Jumper Block Locations (-0xx and -5xx series)

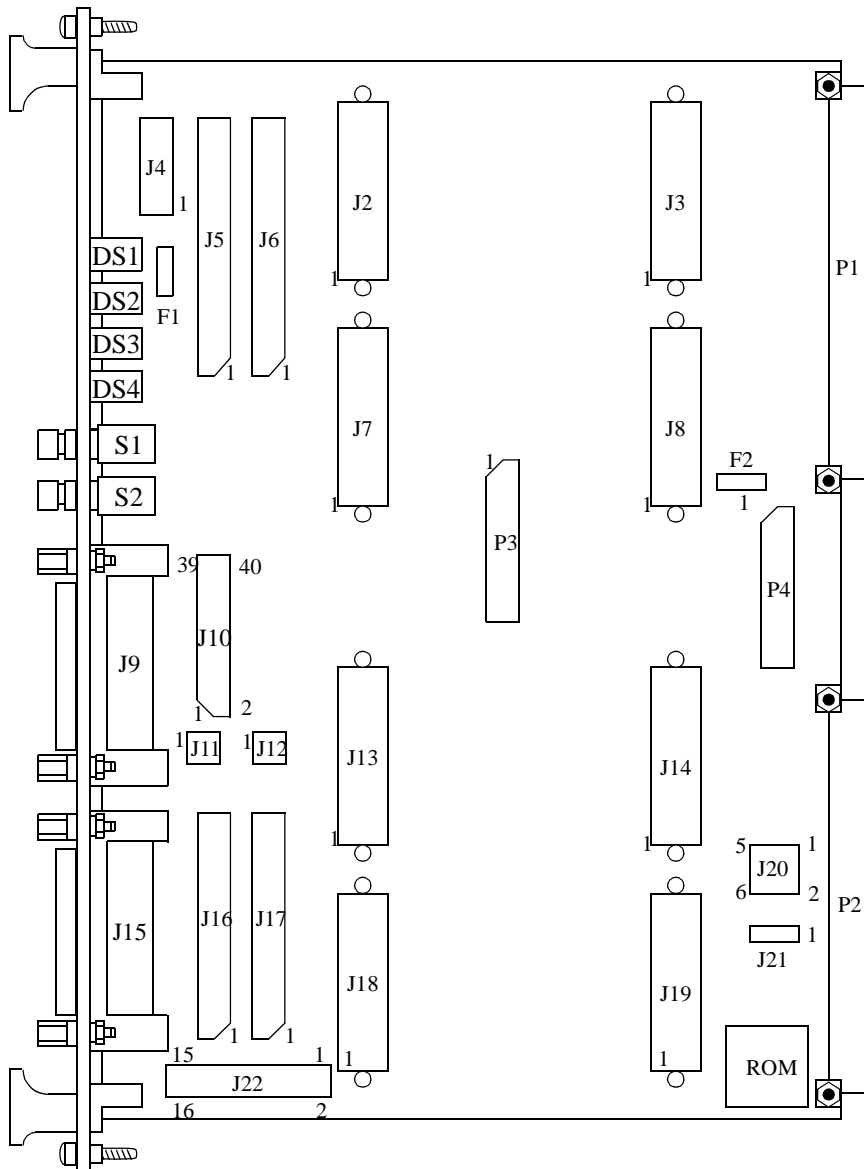


Figure 2-2 Jumper Locations

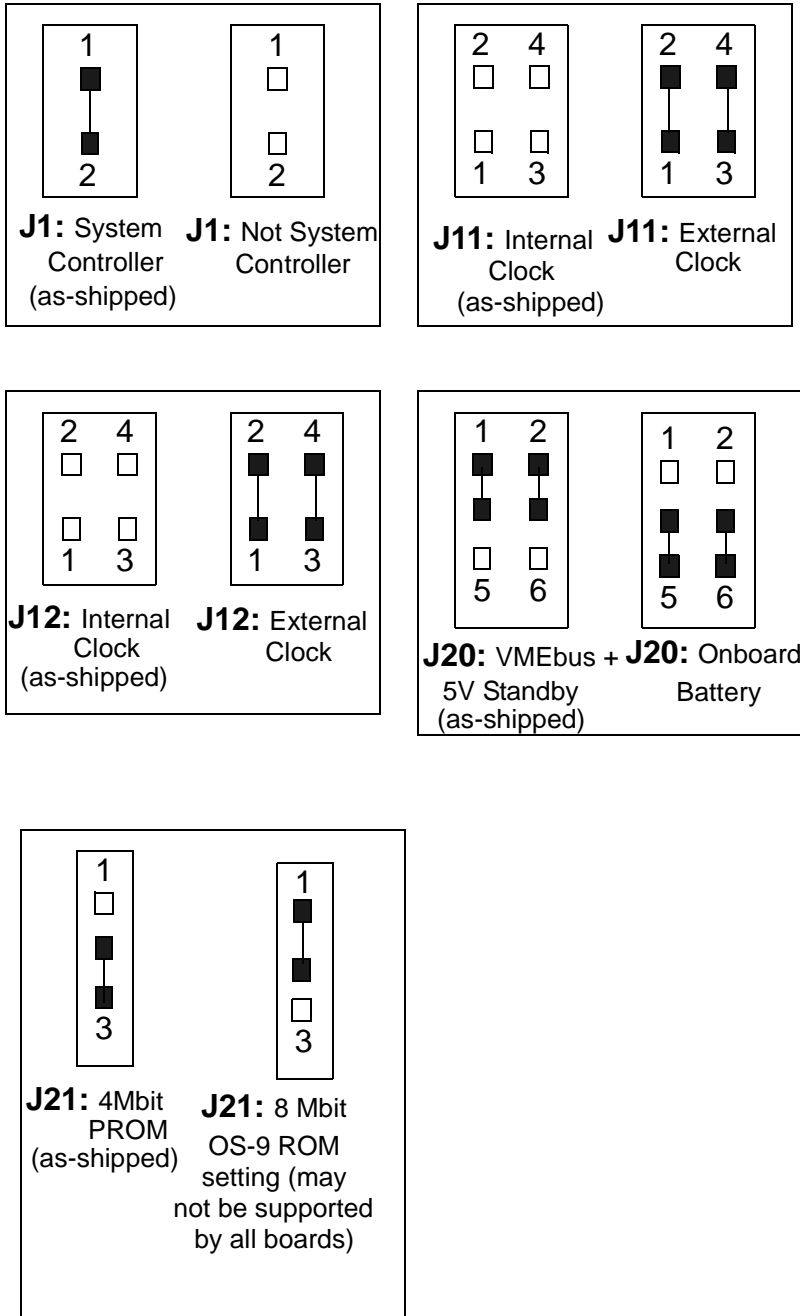
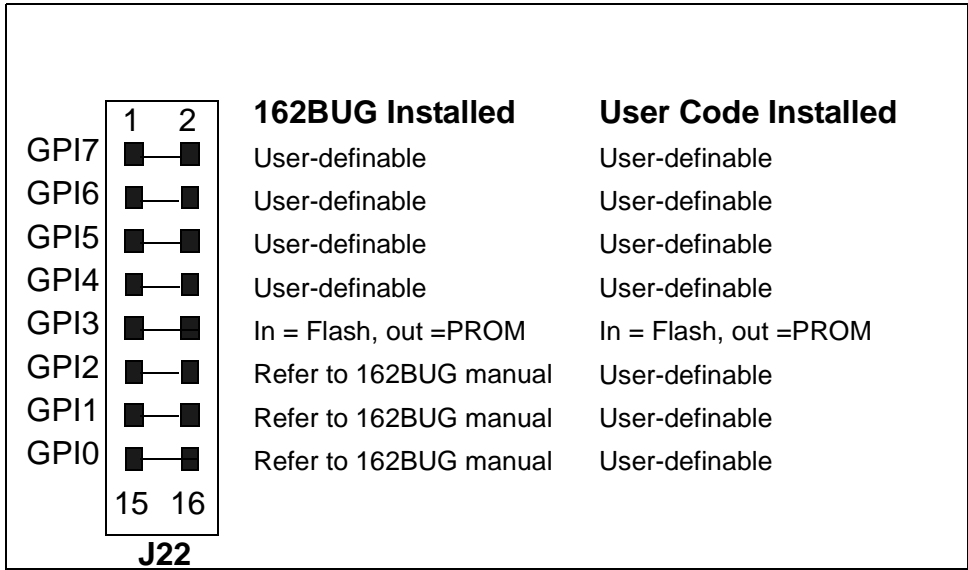


Figure 2-3 Jumper Locations (continued)



Overview of the MVME162 LX CPU Models

The MVME162 LX CPU models provide:

- 68040 or 68LC040 processor running at 33MHZ with on-chip 8K cache, MMU and FPU (68040 only).
- 4M of Parity Dynamic (DRAM), 4MB, 16MB or 32MB of ECC DRAM, the combination of 16MB ECC and 4MB of Parity DRAM, or no DRAM.
- 128K bytes or 2M bytes of battery-backed static RAM.
- 4 RS-232 serial ports.
- On-board SCSI I/O processor (optional).
- On-board Ethernet processor (optional).
- Time-of-day clock/calendar with battery backup.
- 8Kx8 battery backed CMOS RAM (NVRAM)
The OS-9 for 68K area of NVRAM is the first 256 bytes of the user area beginning at FFFC0000.
- 1M bytes of flash memory as primary ROM bank (at FF800000) or secondary ROM bank (at FFA00000, jumper selectable).
- Four 32-pin DIP PROM sockets as primary or secondary ROM banks (jumper selectable - alternates with flash).
- 2 Industry Pack (IP) slots (A - B), unless otherwise specified. The default IP setup by RomBug is detailed in [Table 2-3](#). If a particular application requires a different setup, RomBug can be rebuilt with a customized `initext` module to fit the application's requirements.

Table 2-3 MVME162 LX Model IP Slots

IP Slot	Memory Base	Interrupt Level
A	C0000000	5
B	C0800000	4

Both slots are set for:

Memory Size	8MB
Memory Width	16 bits
Interrupts	Level sensitive, Normal Polarity
Recovery Time	0 microseconds

The MVME162 LX models differ by the type and amount of RAM memory installed, the type of the processor, and support for SCSI and/or Ethernet on the board. Currently all boards operate at 25 MHz. The models known at the time of publication are listed in [Table 2-4](#).

Table 2-4 MVME162 LX Models and Processor Information

Flash	Model	Processor	DRAM Size & Type	SRAM Size	I/O Options
	MVME162-200	LC040	1MB Parity	128KB	no IP interface
	MVME162-201	LC040	1MB Parity	128KB	
	MVME162-202	040	1MB Parity	128KB	SCSI
	MVME162-203	LC040	1MB Parity	128KB	no VME interface
1MB	MVME162-210	LC040	4MB Parity	128KB	
1MB	MVME162-211	LC040	4MB Parity	128KB	SCSI
1MB	MVME162-212	LC040	4MB Parity	128KB	Enet
1MB	MVME162-213	LC040	4MB Parity	128KB	Enet, SCSI
1MB	MVME162-214	LC040	4MB Parity	128KB	no VME interface
1MB	MVME162-215	LC040	4MB Parity	128KB	SCSI, no VME
1MB	MVME162-220	040	4MB Parity	128KB	

Table 2-4 MVME162 LX Models and Processor Information (continued)

Flash	Model	Processor	DRAM Size & Type	SRAM Size	I/O Options
1MB	MVME162-221	040	4MB Parity	128KB	SCSI
1MB	MVME162-222	040	4MB Parity	128KB	Enet
1MB	MVME162-223	040	4MB Parity	128KB	Enet, SCSI
1MB	MVME162-233	LC040	4MB ECC	128KB	Enet, SCSI
1MB	MVME162-250	LC040	16MB ECC	128KB	
1MB	MVME162-251	LC040	16MB ECC	128KB	SCSI
1MB	MVME162-252	LC040	16MB ECC	128KB	Enet
1MB	MVME162-253	LC040	16MB ECC	128KB	Enet, SCSI
1MB	MVME162-260	040	16MB ECC	128KB	
1MB	MVME162-261	040	16MB ECC	128KB	SCSI
1MB	MVME162-262	040	16MB ECC	128KB	Enet
1MB	MVME162-263	040	16MB ECC	128KB	Enet, SCSI

Table 2-4 MVME162 LX Models and Processor Information (continued)

Flash	Model	Processor	DRAM Size & Type	SRAM Size	I/O Options
1MB	MVME162-270	LC040	none	2MB	
1MB	MVME162-271	LC040	none	2MB	SCSI
1MB	MVME162-272	LC040	none	2MB	Enet
1MB	MVME162-273	LC040	none	2MB	Enet, SCSI
1MB	MVME162-280	040	none	2MB	
1MB	MVME162-281	040	none	2MB	SCSI
1MB	MVME162-282	040	none	2MB	Enet
1MB	MVME162-283	040	none	2MB	Enet, SCSI

Figure 2-4 MVME162 Switch, Header, Connector, Fuse and LED Locations (-2xx series)

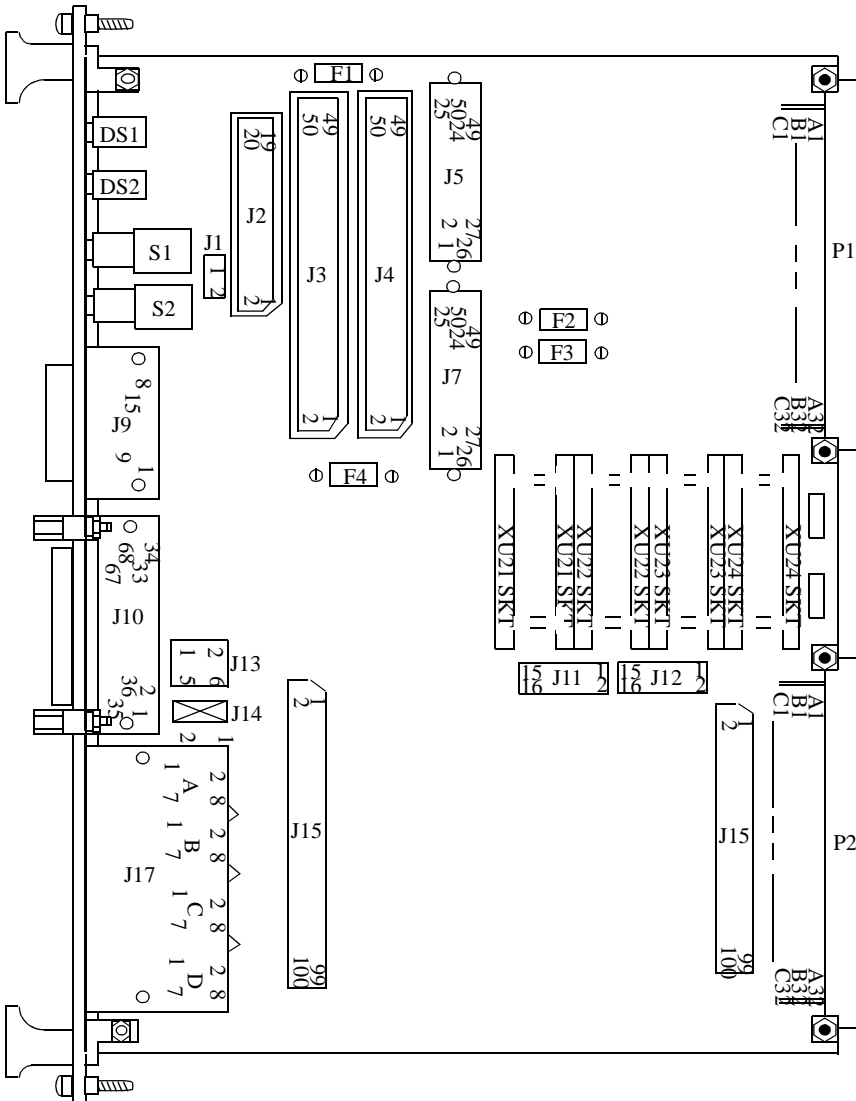


Figure 2-5 EPROM/Flash Configuration Header (J12)

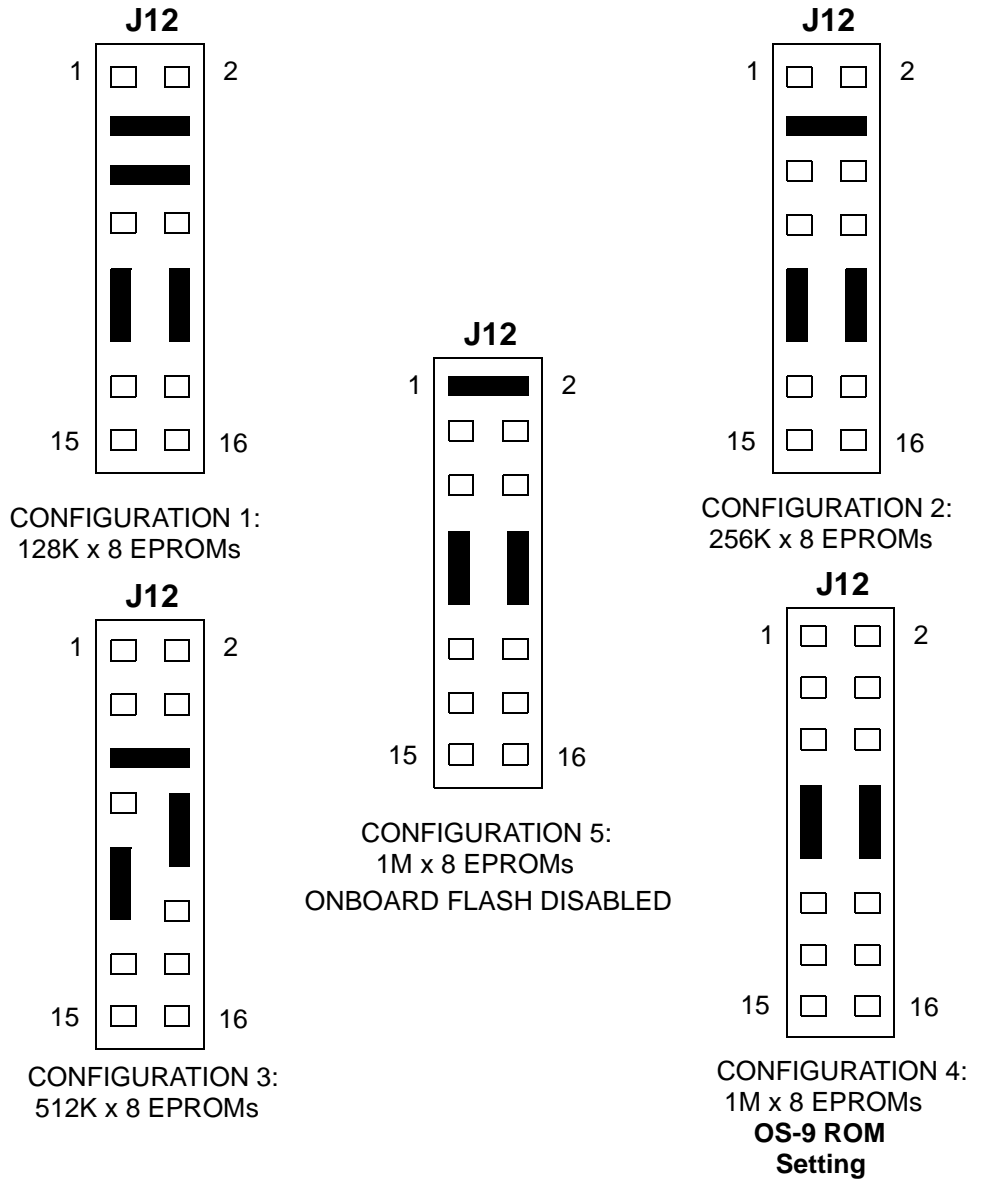






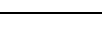


Figure 2-6 General Purpose Readable Jumpers Header (J11)

		J11	162BUG INSTALLED	USER CODE INSTALLED
GP100	1		2 REFER TO 162BUG MANUAL	USER-DEFINABLE
GPIO1			REFER TO 162BUG MANUAL	USER-DEFINABLE
GPIO2			REFER TO 162BUG MANUAL	USER-DEFINABLE
GPIO3	7	<input type="checkbox"/> <input type="checkbox"/>	8 IN=FLASH; OUT=EPROM	IN=FLASH; OUT=EPROM
GPIO4			USER-DEFINABLE	USER-DEFINABLE
GPIO5			USER-DEFINABLE	USER-DEFINABLE
GPIO6			USER-DEFINABLE	USER-DEFINABLE
GPIO7	15		16 USER-DEFINABLE	USER-DEFINABLE

EPROMs Selected (factory configuration)

Figure 2-7 System Controller Select Header (J1)

The MVME162LX is factory-configured as a VMEbus system controller (a jumper is installed across pins 1 and 2 of header J1). Remove the J1 jumper if the MVME162LX is not to be the system controller.

Note: when the MVME162LX is functioning as system controller, the `SCON LED` is turned on.

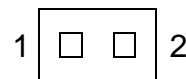
Note: For MVME162LXs without the optional VMEbus interface (i.e., with no VMEchip2), the jumper may be installed or removed without affecting normal operation.

Jumper Block J1



System Controller
(factory configuration)

Jumper Block J1

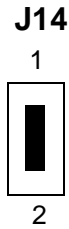


Not System Controller

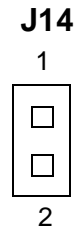
Figure 2-8 SCSI Terminator Enable Header (J14)

The MVME162LX provides terminators for the SCSI bus. The SCSI terminators are enabled/disabled by a jumper on header J14. The SCSI terminators may be configured as follows.

NOTE: If the MVME162LX is to be used at one end of the SCSI bus, the SCSI bus terminators must be enabled.



Onboard SCSI Bus Terminator Enabled
(Factory configuration)



Onboard SCSI Bus Terminator Disabled

Figure 2-9

Header J13 determines the source for onboard static RAM backup power on the MVME162LX main module. Header J1 determines the source for backup power on the 2MB SRAM mezzanine board (if installed).

The following backup power configurations are available for onboard SRAM through header J13. In the factory configuration, the VMEbus +5V standby voltage serves as primary and secondary power source (the onboard battery is disconnected).

NOTE: For MVME162LXs without the optional VMEbus interface (for instance, without the VMEchip2 ASIC), you must select the onboard battery as the backup power source.

CAUTION: Removing all jumpers may temporarily disable the SRAM. Do not remove all jumpers from J13, except for storage.

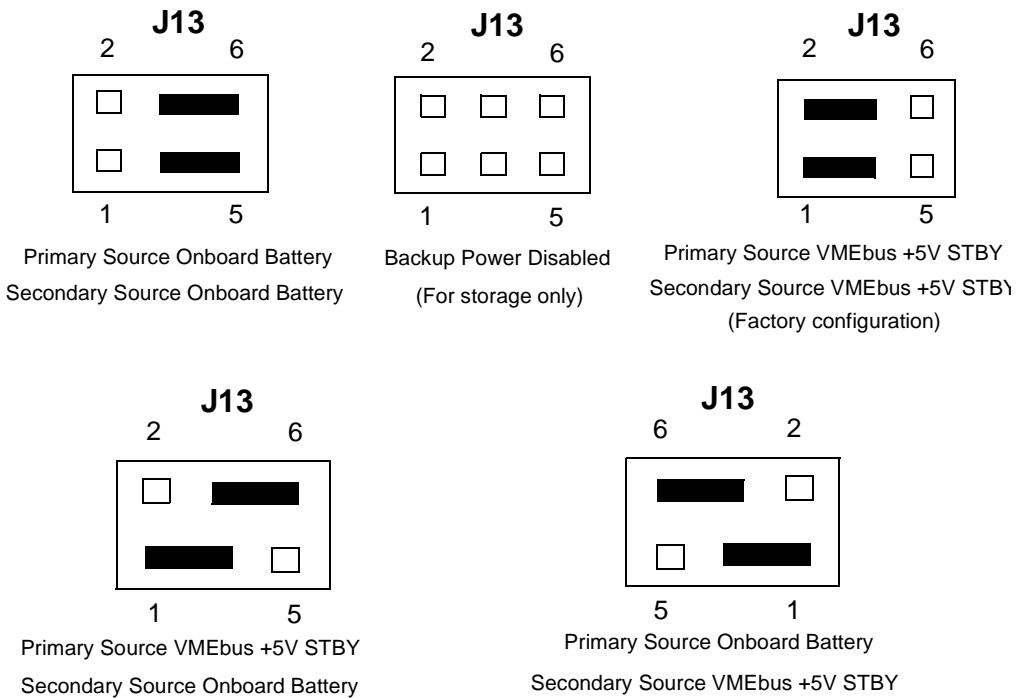
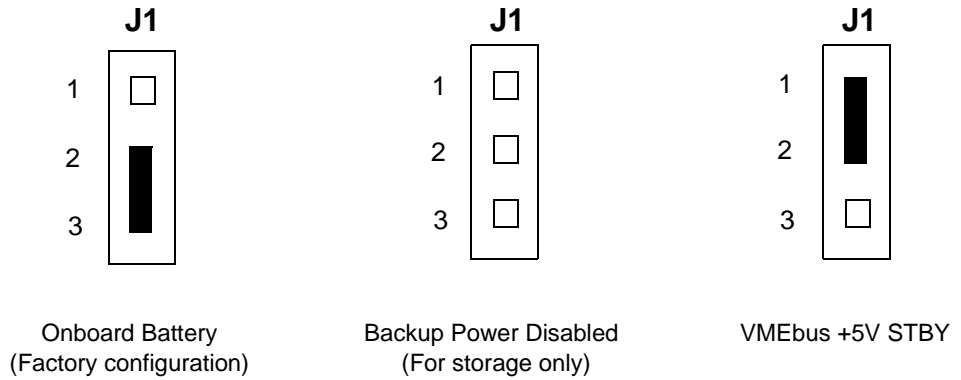


Figure 2-10 Backup Power Configurations

The following backup power configurations are available for the 2MB mezzanine SRAM through header J1 (located on the mezzanine). In the factory configuration, the onboard battery serves as secondary power source.

CAUTION: Removing the jumper may temporarily disable the SRAM. Do not remove the jumpers from J1, except for storage.



ROM Configuration and Organization

On the original and FX MVME162, either flash memory (when available) or the PROM socket can be selected as the primary ROM bank with a jumper. Typically, Motorola ships the MVME162 with a PROM containing 162Bug and with 162Bug residing in flash. The 162Bug includes a facility for reprogramming flash, but it must run from the PROM (as the primary bank), not flash, to operate properly. MVME162 BLS packages include a RomBug image in the file:

```
MWOS/OS9/68040/PORTS/MVME162/CMD5/BOOTOBSJS/ROMBUG/rombugger
```

Some original and FX boards may restrict the PROM size to 512K bytes which will require the OS-9 ROM image to be programmed into the Flash memory for use. The image may optionally be modified and reconstructed from the materials provided in the distribution and downloaded into the board to be programmed into flash.

There are two methods available for installing OS-9's Rombug/Boot on the board. The suggested method retains MotBug in PROM and programs the OS-9 image into the onboard Flash memory. This method limits possible damage to the MotBug PROM and makes it easier to make changes to the embedded boot. The specifics are covered later in the section. The second method involves removing the 162Bug PROM and installing an OS-9 PROM in its place. Take care when removing the 162Bug PROM not to damage any pins and be sure to retain the PROM for use if you ever wish to reprogram the Flash memory.

Once you've decided where to have the OS-9 ROM image reside, the next step is to decide whether you wish to have 162Bug get control on a reset and boot RomBug in the secondary bank or if you want to have the OS-9 ROM code take control directly from reset. If you do not anticipate needing occasional diagnostic testing, or if you desire a faster reset, you can choose to reset directly into the OS-9 ROM code. To start the board via 162Bug set the jumper: **Reset from Flash/ROM**, to the location holding the 162Bug code. Additional configuration is required to have 162Bug transfer control to the OS-9 ROM code.



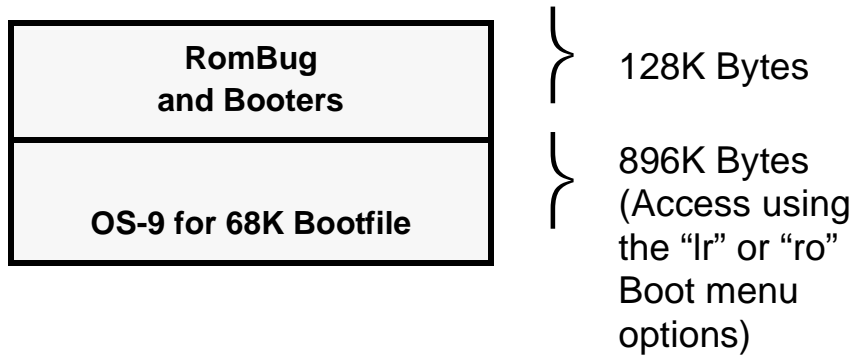
For More Information

For more information see [Configuring 162Bug to Boot RomBug](#).

To reset directly into the OS-9 ROM code, set the jumper to the location where the OS-9 code resides. If an OS-9 PROM is installed, the PROM size jumpers must be reset from the factory setting (512KB) to the setting for 1MB x 8bits. On the MVME162LX, the OS-9 PROM is installed in XU24. Socket U23 can be used for additional OS-9 modules.

The address space of the MVME162 RomBug PROM is divided into the two areas illustrated in [Figure 2-11](#):

Figure 2-11 MVME162 RomBug PROM Address Space



Board Specific OS-9 Modules

The software environment for the MVME162 is built in its `PORTS` directory structure found at `/h0/MWOS/OS9/68020/PORTS/MVME162`. All descriptors, `init` modules, and bootfiles are configured and built in this directory and its subdirectories.

The board-specific drivers available for the MVME162 are:

<code>tk162</code>	System ticker
<code>rtc162</code>	Real-time clock driver
<code>sc162</code>	SCF/serial port driver
<code>scsi162</code>	SCSI low-level driver
<code>sp162</code>	Ethernet driver

Other system modules available for use on the board residing in common directories include:

<code>cache040</code>	Cache control module
<code>ssm040</code>	SSM for the 68040 PMMU
<code>ssm040_cbsup</code>	Same as <code>ssm040</code> with cache copyback for supervisor state
<code>fpsp040</code>	Floating point support package for 68040 board versions
<code>fpu</code>	Floating point emulation for 68LC040 board versions

Installing the Original or FX Series MVME162 CPU Module (EPROM Method)

- Step 1. Remove power from the system and eject the CPU module.
- Step 2. Install the PROM containing the OS-9 ROM image.



Note

Set EPROM Jumper J21 to short pins 1-2 for 8M Bit setting.

Some boards may not support 8M Bit EPROMS. Check the Motorola documentation that came with your board to verify 8M bit EPROM support.



Note

The 32-pin PROM used with the MVME162 requires special considerations when inserting or removing and during handling. A special extraction tool is required to remove the device from its socket. When handling and inserting the PROM, be careful not to damage or bend the pins on the PROM.

- Step 3. (Optional) Select the PROM as the reset bank.
Remove the jumper at J22 pins 9 & 10. Save the jumper by installing it only on pin 10.



Note

The MVME162 can be reset from either the 162Bug in flash memory or the RomBug in PROM. The jumper at J22 pins 9 & 10 switches the address space between these two devices. Once 162Bug has been coldstarted, RomBug can be **booted** from the 162Bug by configuring 162Bug's environment as described in a later section of this manual.

Step 4. Examine and change jumpers on the CPU module.

In general, the jumper settings for the MVME162 module should remain the same as the factory default settings (or configured specifically for your needs).

Step 5. Reinstall the CPU module.

Step 6. Complete hardware installation.

Once the PROM is installed and jumper configuration is completed, carefully read the ***MVME162 Embedded Controller User's Manual*** to complete the hardware installation steps required for the P2 Adaptor card (if used) and the MVME712M Transition Module.



Note

Use of the second serial port requires the installation of a serial line driver module at J10 on the CPU module. Consult your Motorola representative for details.

Installing the MVME162LX CPU Module (EPROM method)

- Step 1. Remove power from the system and eject the CPU module.
- Step 2. Install the OS-9 PROM in XU24 and configure J12 for configuration 4. See [Figure 2-5](#).
- IMPORTANT:** Retain the 162Bug PROM for future use.
- Step 3. (Optional) Select the PROM as the reset bank. Remove the jumper at J11 pins 7 & 8. Save the jumper by installing it only on pin 8.



Note

The MVME162LX can be reset from the code in either flash memory or PROM. The jumper at J11 pins 7 & 8 switches the address space between these two devices. Once 162Bug has been coldstarted, RomBug can be **booted** from the 162Bug by configuring 162Bug's environment as described in a later section of this chapter.

- Step 4. Examine and change jumpers on the CPU module. In general, the jumper settings for the MVME162LX module should remain the same as the factory default settings (or configured specifically for your needs).
- Step 5. Reinsert the CPU module into the system.
- Step 6. Complete hardware installation.
- Once the PROM is installed and jumper configuration is completed, carefully read the ***MVME162LX Embedded Controller User's Manual*** to complete the hardware installation steps required for any I/O connections.
-

Installing the MVME162 CPU Module (Flash programming method)



Note

If you are downloading RomBug over a serial interface, do not perform all of the steps below. Instead, perform step three; once the file transfer is complete, perform step six to complete the installation. For more information on this process, refer to [Appendix A: Downloading RomBug from a Host over a Serial Interface](#).

-
- Step 1. Install ROM Flash image onto a machine configured with a TFTP server. ROM image:
`MWOS/OS9/68040/PORTS/MVME162/CMDS/BOOTOBSJS/ROMBUG/rombugger`
- Step 2. Reset target machine
- Step 3. Use 162Bug's ENV command to configure autobooting to ROMBUG ROM Image (Not required if step 6 option is selected)
See command sequence at [Configuring 162Bug to Boot RomBug](#).
- Step 4. Use 162BUG's NIOT command to setup networking information.
See command sequence at [162Bug's NIOT configuration](#).
- Step 5. Use 162BUG's NIOP command to load the OS-9 ROM image into RAM.
See command sequence at [Download Boot Image using NIOP Command](#).
- Step 6. Program Flash memory with OS-9 ROM image.
See command sequence a [Program OS-9 ROM Image into Flash Memory](#).

- Step 7. (Optional) Select the PROM as the reset bank. Remove the jumper at J11 pins 7 & 8. Save the jumper by installing it only on pin 8.



Note

The MVME162LX can be reset from the code in either flash memory or PROM. The jumper at J11 pins 7 & 8 switches the address space between these two devices. Once 162Bug has been coldstarted, RomBug can be **booted** from the 162Bug by configuring 162Bug's environment as described in a later section of this manual.

- Step 8. Examine and change jumpers on the CPU module. In general, the jumper settings for the MVME162LX module should remain the same as the factory default settings (or configured specifically for your needs).
- Step 9. Reinstall the CPU module.
- Step 10. Complete hardware installation. Once the PROM is installed and jumper configuration is completed, carefully read the ***MVME162LX Embedded Controller User's Manual*** to complete the hardware installation steps required for any I/O connections.
-

Configuring 162Bug to Boot RomBug

Motorola's 162Bug, which resides in flash on the original and FX series and PROM on the LX series, can be configured to automatically boot RomBug using its ROMboot feature. This allows resetting into the 162Bug in order to keep the diagnostics handy, then automatically transferring control to RomBug to bring up OS-9. However, this method is slower than resetting directly into RomBug.

Configuring 162Bug to Boot RomBug

Step 1. Verify reset into 162Bug.

On the original and FX series, ensure 162Bug is installed in EPROM and the jumper at J22 pins 9 & 10 is removed.

On the LX series, ensure the 162Bug PROM is installed in XU24 and the jumper at J11 pins 7 & 8 is removed.

Step 2. Reset the system to bring up the 162-Bug> prompt:

```
Copyright Motorola Inc. 1988 - 1993, All Rights Reserved
```

```
MVME162 Debugger/Diagnostics Release Version 1.2 - 02/14/93  
COLD Start
```

```
Local Memory Found =00400000 (&4194304) (may vary with other memory  
configurations)
```

```
MPU Clock Speed =25Mhz
```

```
162-Bug>
```

Step 3. Enter the ENV command at the 162-Bug> prompt entering the following values at the configuration field prompt:

```
162-Bug> ENV<return>  
Bug or System environment [B/S] = B? <return>  
Field Service Menu Enable [Y/N] = N? <return>  
Remote Start Method Switch [G/M/B/N] = B? <return>  
Probe System for Supported I/O Controllers [Y/N] = Y? Y<return>  
Negate VMEbus SYSFAIL* Always [Y/N] = N? <return>
```

```

Local SCSI Bus Reset on Debugger Startup [Y/N] = N? <return>
Local SCSI Bus Negotiations Type [A/S/N] = A? <return>
Industry Pack Reset on Debugger Startup [Y/N] = N? <return>
Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y? <return>
Auto Boot Enable [Y/N] = N? <return>
Auto Boot at power-up only [Y/N] = Y? <return>
Auto Boot Controller LUN = 00? <return>
Auto Boot Device LUN = 00? <return>
Auto Boot Abort Delay = 15? <return>
Auto Boot Default String [NULL for a empty string] = ? <return>
ROM Boot Enable [Y/N] = N? Y<return>
ROM Boot at power-up only [Y/N] = Y? N<return>
ROM Boot Enable search of VMEbus [Y/N] = N? <return>
ROM Boot Abort Delay = 0? 3<return>
For the original and FX series boards, the following two entries apply:
ROM Boot Direct Starting Address = FF800000? FFA00400<return>
ROM Boot Direct Ending Address = FFDFFFFC? FFA00500<return>
For the LX series boards, the following two entries apply:
ROM Boot Direct Starting Address = FF800000? FF900400<return>
ROM Boot Direct Ending Address = FFDFFFFC? FF900500<return>
Network Auto Boot Enable [Y/N] = N? <return>
Network Auto Boot at power-up only [Y/N] = Y? <return>
Network Auto Boot Controller LUN = 00? <return>
Network Auto Boot Device LUN = 00? <return>
Network Auto Boot Abort Delay = 5? <return>
Network Auto Boot Configuration Parameters Pointer (NVRAM) = 00000000?
FFE10000<return>
Memory Search Starting Address = 00000000? FFE00000<return>
Memory Search Ending Address = 00010000? FFE10000<return>
Memory Search Increment Size = 00010000? <return>
Memory Search Delay Enable [Y/N] = N? <return>
Memory Search Delay Address = FFFFD20F? <return>
Memory Size Enable [Y/N] = Y? <return>
Memory Size Starting Address = 00000000? <return>
Memory Size Ending Address = 00400000? <return> (value may vary with
installed memory)
Base Address of Dynamic Memory = 00000000? <return>
Size of Parity Memory = 00400000? <return>
Size of ECC Memory Board #0 = 00000000? <return>
Size of ECC Memory Board #1 = 00000000? <return>
Base Address of Static Memory = FFE00000? <return>
Size of Static Memory = 00800000? <return>
Slave Enable #1 [Y/N] = Y? N<return>
Slave Starting Address #1 = 00000000? <return>
Slave Ending Address #1 = 000FFFFFF? <return>
Slave Address Translation Address #1 = 00000000? <return>
Slave Address Translation Select #1 = 00000000? <return>
Slave Control #1 = 03FF? <return>
Slave Enable #2 [Y/N] = N? <return>
Slave Starting Address #2 = 00000000? <return>
Slave Ending Address #2 = 00000000? <return>
Slave Address Translation Address #2 = 00000000? <return>
Slave Address Translation Select #2 = 00000000? <return>
Slave Control #2 = 0000? <return>

```

```

Master Enable #1 [Y/N] = Y? N<return>
Master Starting Address #1 = 02000000? <return>
Master Ending Address #1 = EFFFFFFF? <return>
Master Control #1 = 0D? <return>
Master Enable #2 [Y/N] = N? <return>
Master Starting Address #2 = 00000000? <return>
Master Ending Address #2 = 00000000? <return>
Master Control #2 = 00? <return>
Master Enable #3 [Y/N] = Y? N<return>
Master Starting Address #3 = 00000000? <return>
Master Ending Address #3 = 00000000? <return>
Master Control #3 = 00? <return>
Master Enable #4 [Y/N] = N? <return>
Master Starting Address #4 = 00000000? <return>
Master Ending Address #4 = 00000000? <return>
Master Address Translation Address #4 = 00000000? <return>
Master Address Translation Select #4 = 00000000? <return>
Master Control #4 = 00? <return>
Short I/O (VMEbus A16) Enable [Y/N] = Y? N<return>
Short I/O (VMEbus A16) Control = 01? <return>
F-Page (VMEbus A24) Enable [Y/N] = Y? N<return>
F-Page (VMEbus A24) Control = 02? <return>
ROM Access Time Code = 03? <return>
FLASH Access Time Code = 02? <return>
MCC Vector Base = 05? <return>
VMEC2 Vector Base #1 = 06? <return>
VMEC2 Vector Base #2 = 07? <return>
VMEC2 GCSR Group Base Address = D2? <return>
VMEC2 GCSR Board Base Address = 00? <return>
VMEbus Global Time Out Code = 01? <return>
Local Bus Time Out Code = 00? <return>
VMEbus Access Time Out Code = 02? <return>
IP A Base Address = 00000000? <return>
IP B Base Address = 00000000? <return>
IP C Base Address = 00000000? <return>
IP D Base Address = 00000000? <return>
IP D/C/B/A Memory Size = 00000000? <return>
IP D/C/B/A General Control = 00000000? <return>
IP D/C/B/A Interrupt 0 Control = 00000000? <return>
IP D/C/B/A Interrupt 1 Control = 00000000? <return>

Update Non-Volatile RAM (Y/N)? Y<return>

Reset Local System (CPU) (Y/N)? Y<return>

```

The system resets twice after responding to the last prompt (each time with a three second delay during which the operator may abort the ROMboot process by pressing <break>), then RomBug is entered. Resetting from RomBug brings the operator back to 162Bug with

another opportunity to interrupt the automatic ROMboot. Once configured, RomBug can manually be booted from 162Bug by entering the RB command at the 162-Bug> prompt.

**Note**

All of the above 162Bug entries are explained in the *162Bug User's Manual*.

**Note**

An ABORT-RESET sequence disables an automatic ROM boot sequence. As a result, it cannot be used alone to force an OS-9 RomBug reconfiguration session when 162Bug is in the primary ROM bank. To force a reconfiguration session from 162Bug, press and hold the ABORT switch, enter the RB command, wait at least as long as the ROMboot abort delay, and then release the ABORT switch.

162Bug's NIOT configuration

Step 1. Verify reset into 162Bug and that 162Bug is running from EPROM.

Step 2. Reset the system to bring up the 162-Bug> prompt:
 Note: you may need to press <break> if 162Bug has been previously configured to ROMBoot

```
Copyright Motorola Inc. 1988 - 1993, All Rights Reserved
```

```
MVME162 Debugger/Diagnostics Release Version 1.2 - 02/14/93
COLD Start
```

```
Local Memory Found =00400000 (&4194304) (may vary with other memory
configurations)
```

```
MPU Clock Speed =25Mhz
```

```
ROMboot in progress... To abort hit <BREAK> <break>
--Break Detected--
162-Bug>
```

Step 3. Enter appropriate Data in NIOT command.

```
162-Bug>niot
Controller LUN =00?
Device LUN      =00?
Node Control Memory Address =FFE10000?
Client IP Address      =0.0.0.0?  ENTER DATA HERE
Server IP Address      =0.0.0.0?  ENTER DATA HERE
Subnet IP Address Mask =0.0.0.0?  ENTER DATA HERE
Broadcast IP Address   =0.0.0.0?
Gateway IP Address     =0.0.0.0?
Boot File Name ("NULL" for None)      =?  ENTER DATA HERE
Argument File Name ("NULL" for None) =?
Boot File Load Address      =00000000?  00100000
Boot File Execution Address  =00000000?  00100000
Boot File Execution Delay    =00000010?
Boot File Length             =00000000?
Boot File Byte Offset        =00000000?
BOOTP/RARP Request Retry     =00?
TFTP/ARP Request Retry       =00?
Trace Character Buffer Address =00000000?
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?
BOOTP/RARP Reply Update Control: Yes/No (Y/N)           =Y?
```

Update Non-Volatile RAM (Y/N)? **y**

```
162-Bug>
```

Download Boot Image using NIOP Command

Step 1. Verify reset into 162Bug and that 162Bug is running from EPROM.

Step 2. (OPTIONAL) Reset the system to bring up the 162-Bug> prompt:
 Note: you may need to press <break> if 162Bug has been previously configured to ROMBoot

```
Copyright Motorola Inc. 1988 - 1993, All Rights Reserved
```

```
MVME162 Debugger/Diagnostics Release Version 1.2 - 02/14/93
COLD Start
```

```
Local Memory Found =00400000 (&4194304) (may vary with other memory
configurations)
```

```
MPU Clock Speed =25Mhz
```

```
ROMboot in progress... To abort hit <BREAK> <break>
--Break Detected--
162-Bug>
```

Step 3. Ensure the time-of-day clock is started with the SET command (SET <yymmddhhmm>). The TIME command can confirm the correct setting and operation.

Step 4. 162-Bug>niop

```
Controller LUN =00?
Device LUN      =00?
Get/Put        =G?
File Name       =? rombugger
Memory Address  =FFE0E000? 100000
Length         =00000000?
Byte Offset    =00000000?
```

```
Bytes Received =&1048576, Bytes Loaded =&1048576
Bytes/Second   =&80659, Elapsed Time =13 Second(s)
162-Bug>
```

Program OS-9 ROM Image into Flash Memory

Program 1 MB flash image from ROM Image downloaded into RAM
 Flash memory is at FFA00000

Step 1. Perform NIOP command above just before PFLASH command. Do Not reset between commands.

Step 2. 162-Bug>**pflash 10000:10000 FFA00000**

```

Source Starting/Ending Addresses      =00100000/001FFFFFF
Destination Starting/Ending Addresses =FFA00000/FFAFFFFFF
Number of Effective Bytes             =00100000 (&1048576)
Program FLASH Memory (Y/N)? y
Erasing Block Number      =00 ($FFA00000)
Erasing Block Number      =01 ($FFA10000)
Erasing Block Number      =02 ($FFA20000)
Erasing Block Number      =03 ($FFA30000)
Erasing Block Number      =04 ($FFA40000)
Erasing Block Number      =05 ($FFA50000)
Erasing Block Number      =06 ($FFA60000)
Erasing Block Number      =07 ($FFA70000)
Erasing Block Number      =08 ($FFA80000)
Erasing Block Number      =09 ($FFA90000)
Erasing Block Number      =10 ($FFAA0000)
Erasing Block Number      =11 ($FFAB0000)
Erasing Block Number      =12 ($FFAC0000)
Erasing Block Number      =13 ($FFAD0000)
Erasing Block Number      =14 ($FFAE0000)
Erasing Block Number      =15 ($FFAF0000)
Programming Block Number =00 ($FFA00000)
Programming Block Number =01 ($FFA10000)
Programming Block Number =02 ($FFA20000)
Programming Block Number =03 ($FFA30000)
Programming Block Number =04 ($FFA40000)
Programming Block Number =05 ($FFA50000)
Programming Block Number =06 ($FFA60000)
Programming Block Number =07 ($FFA70000)
Programming Block Number =08 ($FFA80000)
Programming Block Number =09 ($FFA90000)
Programming Block Number =10 ($FFAA0000)
Programming Block Number =11 ($FFAB0000)
Programming Block Number =12 ($FFAC0000)
Programming Block Number =13 ($FFAD0000)
Programming Block Number =14 ($FFAE0000)
Programming Block Number =15 ($FFAF0000)
FLASH Memory Programming Complete
162-Bug>

```

Chapter 3: MVME167 Reference

This section contains information about the MVME167 models. It contains the following sections:

- **MVME167 Quick Reference**
- **Overview of the MVME167 CPU Module**
- **ROM Configuration and Organization**
- **Board Specific OS-9 Modules**
- **Configuring 167Bug to Boot RomBug**



MVME167 Quick Reference

Ports Directory: <mwos>/OS9/68040/PORTS/MVME167

Console Port Location: MVME712, connector labeled “console”

OS-9 EPROM Size: (2) 256K X 16 bits (1 Megabyte total)
No flash Option.

ROM Booting Search Order:

FF800400:FF808400

FFA00400:FFA08400

BootP Ethernet booting device: ie

SPF Ethernet descriptor name: spie0

CMDS Build sequence:

<mwos>/OS9/68000/CMDS

<mwos>/OS9/68020/CMDS

<mwos>/OS9/68040/CMDS

<mwos>/OS9/68040/PORTS/MVME167/CMDS

Overview of the MVME167 CPU Module

The MVME167 CPU module provides:

- 68040 processor running at 25MHz or 33Mhz with on-chip 8K cache, MMU, and FPU
- 4, 8, 16 or 32M bytes of on-board DRAM
- 128K Bytes of on-board Static RAM
- 4 RS-232C serial ports
- Centronics parallel printer port
- On-board SCSI I/O processor
- On-board Ethernet processor
- Time-of-day clock/calendar with battery backup 8Kx8 battery backed CMOS RAM (NVRAM).

The OS-9 for 68K area of NVRAM is the first 256 bytes of the user area beginning at FFFC0000.

- Four 44-pin PLCC (16-bit) EPROM sockets organized as two banks of 32-bit wide (high and low word) memory:

Primary Bank: FF800000

High Word XU1

Low Word XU2

Secondary Bank: FFA00000

High Word XU3

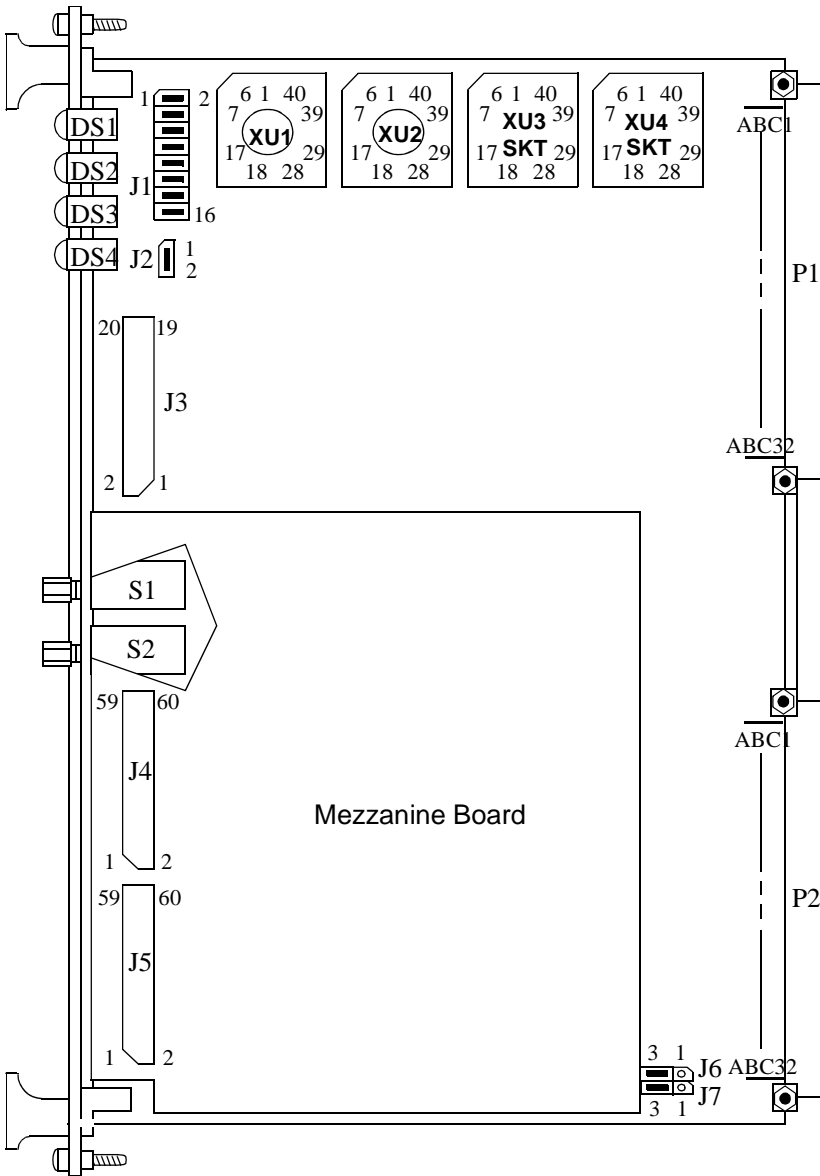
Low Word XU4

Motorola has numerous variations of the board. These boards are differentiated by the amount of RAM memory installed, the speed of the processor, and the design revisions of the board. A representative list of module numbers are listed in [Table 3-1](#).

Table 3-1 MVME167 Models and Processor Information

CPU Model	Memory Size	Processor Speed
MVME167	4M	25mhz
MVME167A	8M	25mhz
MVME167B	16M	25mhz
MVME167C	32M	25mhz
MVME167-01	4M	25mhz
MVME167-02	8M	25mhz
MVME167-03	16M	25mhz
MVME167-04	32M	25mhz
MVME167-31	4M	33mhz
MVME167-32	8M	33mhz
MVME167-33	16M	33mhz
MVME167-34	32M	33mhz

Figure 3-1 MVME167 Jumper Block Locations



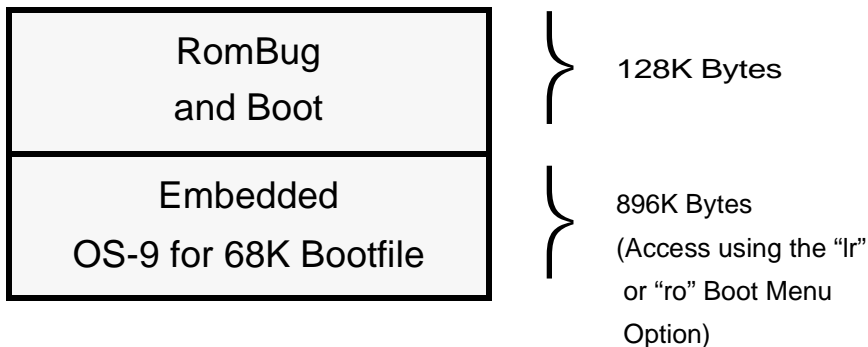
ROM Configuration and Organization

There are two possible ROM configurations for the MVME167. As the system installer, you must choose which ROM configuration to adopt, depending on your system application.

- The first retains the 167Bug ROMs in the primary bank (for ease of hardware diagnostic testing) and adds the OS-9 RomBug EPROMs in the secondary bank. In this configuration, 167Bug gets control on a reset and **boots** RomBug in the secondary bank (see [Configuring 167Bug to Boot RomBug](#)).
- The second configuration replaces the 167Bug EPROMs with the OS-9 RomBug EPROMs in the primary bank. In this configuration, RomBug always gets control on a reset.

The address space of the MVME167 RomBug EPROMS is divided into the two areas illustrated in [Figure 3-1](#).

Figure 3-2 Address Space of MVME167 RomBug EPROMS



Board Specific OS-9 Modules

The software environment for the MVME167 is built in its `PORTS` directory structure found at `/h0/MWOS/OS9/68040/PORTS/MVME167`. All descriptors, `init` modules, and bootfiles are configured and built in this directory and its subdirectories.

The board specific drivers available for the MVME167 include:

<code>tk167</code>	System ticker
<code>rtc167</code>	Real-time clock driver
<code>sc167</code>	SCF/serial port driver
<code>scp167</code>	SCF/parallel port driver
<code>scsi167</code>	SCSI low-level driver
<code>sp167</code>	Ethernet driver

Other system modules available for use on the board residing in common directories include:

<code>cache040</code>	Cache control module
<code>ssm040</code>	SSM for the 68040 PMMU
<code>ssm040_cbsup</code>	Same as above with cache copyback for supervisor state
<code>fppsp040</code>	Floating point support package for 68040 CPU chips

Installing the MVME167 CPU Module

Step 1. Remove power from the system and eject the CPU module.

Step 2. Install the EPROM labeled HIGH in socket XU1. Install the EPROM labeled LOW in socket XU2.

If you desire, you may leave the Motorola PROMs in U1 and U2. The OS-9 RomBug ROMs can co-exist with the Motorola PROMs. Install the EPROM labeled HIGH in socket XU3 and the EPROM labeled LOW in XU4.



Note

If you choose this method you must also complete the **Configuring 167Bug to Boot RomBug** section.



WARNING

The 44-pin EPROMs used with the MVME167 require special considerations when inserting or removing and during handling. A special extraction tool is required to remove the devices from their sockets. When handling and inserting the EPROMs, be careful not to damage or bend the pins on the EPROMs.

Step 3. Set jumpers.

In general, the jumper settings for the MVME167 module should remain the same as the factory default settings (or configured specifically for your needs).

Figure 3-3 MVME167 Jumper Settings

Software Readable Header:

(Note: Unused by Microware)

J1: 1 to 2, 3 to 4, 5 to 6, 7 to 8, 9 to 10,
11 to 12, 13 to 14, 15 to 16

Jumper Block J1

2	4	6	8	10	12	14	16
1	3	5	7	9	11	13	15

System Controller Selection:

J2: System Controller



(Factory Default)

Not System Controller



Serial Port 4 Clock Configuration Select Headers:

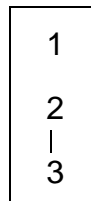
J6: Jumper 2-3 to receive RTXC4 (Factory Configuration)

J7: Jumper 2-3 to receive TRXC4 (Factory Configuration)

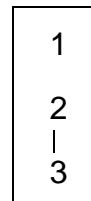
J6: Jumper 1-2 to drive RTXC4

J7: Jumper 1-2 to drive TRXC4

Jumper Block J6



Jumper Block J7



Step 4. Reinsert the CPU module into the system.

Step 5. Complete hardware installation.

Once the EPROM and jumper installation is done, carefully read the **MVME167 User Manual** (supplied by Motorola) to complete the hardware installation steps required for the P2 Adaptor Card (if used)

and the MVME712 Transition Module(s). These instructions also detail the various options available for cabling SCSI devices to the MVME167 module.



Note

Jumper installation details for the MVME712 module(s) are supplied in the support module appendix. However, carefully read the Motorola supplied documentation to ensure all hardware cabling requirements are correctly satisfied.

Configuring 167Bug to Boot RomBug

Motorola's 167Bug which resides in EPROM can be configured to automatically boot RomBug using its ROMboot feature. This allows coldstarting from the 167Bug in order to keep the diagnostics handy, then automatically transferring control to RomBug to bring up OS-9. However, this method is much slower than coldstarting directly into RomBug.

Configuring 167Bug to Boot RomBug

Step 1. Verify PROM installation.

Ensure 167Bug EPROMs are installed in the primary bank and the RomBug EPROMs are installed in the secondary bank.

Step 2. Start up the system.

Reinstall the CPU module and reset the system to bring up the 167-Bug> prompt:

```
Copyright Motorola Inc. 1988 - 1993, All Rights Reserved
```

```
MVME167 Debugger/Diagnostics Release Version 1.5 - 02/14/93  
COLD Start
```

```
Local Memory Found =03000000 (&50331648)
```

```
MPU Clock Speed =33Mhz
```

```
167-Bug>
```

Step 3. Enable the ROMboot feature.

Enter the ENV command at the 167-Bug> prompt entering the following values at the configuration field prompt:

```
167-Bug>ENV<return>  
Bug or System environment [B/S] = B? <return>  
Field Service Menu Enable [Y/N] = N? <return>  
Remote Start Method Switch [G/M/B/N] = B? <return>  
Probe System for Supported I/O Controllers [Y/N] = Y? N<return>
```

```

Negate VMEbus SYSFAIL* Always [Y/N] = N? <return>
Local SCSI Bus Reset on Debugger Startup [Y/N] = N? <return>
Local SCSI Bus Negotiations Type [A/S/N] = A? <return>
Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y? <return>
Auto Boot Enable [Y/N] = N? <return>
Auto Boot at power-up only [Y/N] = Y? <return>
Auto Boot Controller LUN = 00? <return>
Auto Boot Device LUN = 00? <return>
Auto Boot Abort Delay = 15? <return>
Auto Boot Default String [NULL for a empty string] = ? <return>
ROM Boot Enable [Y/N] = N? Y<return>
ROM Boot at power-up only [Y/N] = Y? N<return>
ROM Boot Enable search of VMEbus [Y/N] = N? <return>
ROM Boot Abort Delay = 0? 3<return>
ROM Boot Direct Starting Address = FF800000? FFA00400<return>
ROM Boot Direct Ending Address = FFDFFFFC? FFA00500<return>
Network Auto Boot Enable [Y/N] = N? <return>
Network Auto Boot at power-up only [Y/N] = Y? <return>
Network Auto Boot Controller LUN = 00? <return>
Network Auto Boot Device LUN = 00? <return>
Network Auto Boot Abort Delay = 5? <return>
Network Auto Boot Configuration Parameters Pointer (NVRAM) = 00000000? <return>
Memory Search Starting Address = 00000000? FFE00000<return>
Memory Search Ending Address = 00010000? FFE10000<return>
Memory Search Increment Size = 00010000? <return>
Memory Search Delay Enable [Y/N] = N? <return>
Memory Search Delay Address = FFFFCE0F? <return>
Memory Size Enable [Y/N] = Y? <return>
Memory Size Starting Address = 00000000? <return>
Memory Size Ending Address = 0xx00000? <return>
Base Address of Local Memory = 00000000? <return>
Size of Local Memory Board #0 = 0xx00000? <return>
Size of Local Memory Board #1 = 0xx00000? <return>
Slave Enable #1 [Y/N] = Y? N<return>
Slave Starting Address #1 = 00000000? <return>
Slave Ending Address #1 = 0xxFFFFFF? <return>
Slave Address Translation Address #1 = 00000000? <return>
Slave Address Translation Select #1 = Fxx00000? <return>
Slave Control #1 = 03FF? <return>
Slave Enable #2 [Y/N] = N? <return>
Slave Starting Address #2 = 00000000? <return>
Slave Ending Address #2 = 00000000? <return>
Slave Address Translation Address #2 = 00000000? <return>
Slave Address Translation Select #2 = 00000000? <return>
Slave Control #2 = 0000? <return>
Master Enable #1 [Y/N] = Y? N<return>
Master Starting Address #1 = 0xx00000? <return>
Master Ending Address #1 = 00000000? <return>
Master Control #1 = 00? <return>
Master Enable #2 [Y/N] = N? <return>
Master Starting Address #2 = 00000000? <return>
Master Ending Address #2 = 00000000? <return>
Master Control #2 = 00? <return>
Master Enable #3 [Y/N] = N? <return>

```



```

Master Starting Address #3 = 00000000? <return>
Master Ending Address #3   = 00000000? <return>
Master Control #3 = 00? <return>
Master Enable #4 [Y/N] = N? <return>
Master Starting Address #4 = 00000000? <return>
Master Ending Address #4   = 00000000? <return>
Master Address Translation Address #4 = 00000000? <return>
Master Address Translation Select #4 = 00000000? <return>
Master Control #4 = 00? <return>
Short I/O (VMEbus A16) Enable [Y/N] = Y? N<return>
Short I/O (VMEbus A16) Control      = 00? <return>
F-Page (VMEbus A24) Enable [Y/N]   = Y? N<return>
F-Page (VMEbus A24) Control        = 00? <return>
ROM Speed Bank A Code               = 00? <return>
ROM Speed Bank B Code               = 00? <return>
Static RAM Speed Code               = 00? <return>
PCC2 Vector Base                    = 05? <return>
VMEC2 Vector Base #1                = 06? <return>
VMEC2 Vector Base #2                = 07? <return>
VMEC2 GCSR Group Base Address       = CC? <return>
VMEC2 GCSR Board Base Address       = 00? <return>
VMEbus Global Time Out Code         = 01? <return>
Local Bus Time Out Code             = 00? <return>
VMEbus Access Time Out Code         = 02? <return>

Update Non-Volatile RAM (Y/N)? Y

Reset Local System (CPU) (Y/N)? Y

```

The system resets twice after responding to the last prompt (each time with a three second delay during which the operator may abort the ROMboot process by pressing <break>), then RomBug is entered. Resetting from RomBug brings the operator back to 167Bug with another opportunity to interrupt the automatic ROMboot. Once configured, RomBug can manually be booted from 162Bug by entering the RB command at the 167-Bug> prompt.



Note

All of the above 167Bug entries are explained in the *167Bug User's Manual*.



Note

An `ABORT-RESET` sequence disables an automatic ROMboot sequence. As a result, it cannot be used alone to force an OS-9 RomBug reconfiguration session when 167Bug is in the primary ROM bank. To force a reconfiguration session from 167Bug, press and hold the `ABORT` switch, enter the `RB` command, wait at least as long as the ROM boot abort delay, and then release the `ABORT` switch.

Chapter 4: MVME172 Reference

This section contains information about the MVME172 models. It contains the following sections:

- **MVME172 Quick Reference**
- **Overview of the MVME172 CPU Models**
- **ROM Configuration and Organization**
- **Board Specific OS-9 Modules**
- **Installing the MVME172 CPU Module (EPROM Method)**
- **Configuring 172Bug to Boot RomBug**



For More Information

This chapter details the jumper and processor locations for the MVME172 FX and LX models. For information on the Petra model, refer to the *Motorola MVME VME Embedded Controller Installation and Use Guide* shipped with your hardware.



MVME172 Quick Reference

Ports Directory: <mwos>/OS9/68060/PORTS/MVME172

Console Port Location: Faceplate, connector labeled “console”

OS-9 EPROM Size: 1 megabyte X 8 bits

May be programmed onto EPROM or can be programmed into Flash

ROM Booting Search Order:

FF800400:FF808400

FFA00400:FFA08400

BootP Ethernet booting device: ie

SPF Ethernet descriptor name: spie0

CMDS Build sequence:

<mwos>/OS9/68000/CMDS

<mwos>/OS9/68020/CMDS

<mwos>/OS9/68060/CMDS

<mwos>/OS9/68060/PORTS/MVME172/CMDS

Overview of the MVME172 CPU Models

The MVME172 models provide:

- 68060 or 68LC060 processor running at 60MHz or 64 MHz with on-chip 8K cache, MMU and FPU (68060 only)
- 4M to 64M Bytes of Dynamic RAM
- 512K Bytes of battery-backed Static RAM
- 4 RS-232 serial ports
- On-board SCSI I/O processor (Optional)
- On-board Ethernet processor (Optional)
- Time-of-day clock/calendar with battery backup
- 8Kx8 battery backed CMOS RAM (NVRAM)
The OS-9 for 68K area of NVRAM is the first 256 bytes of the user area beginning at FFFC0000.
- 2M Bytes of flash memory as primary ROM bank (at FF800000) or secondary ROM bank (at FFA00000, jumper selectable)
- FX: One 32-pin PLCC JEDEC EPROM socket
LX: Two 32-pin DIP EPROM sockets are selectable as primary or secondary ROM bank. (
EPROMS are jumper selectable to alternate with flash as primary ROM bank
- 2 Industry Pack (IP) slots (A - D). The default IP setup by RomBug is detailed in [Table 4-1](#). If a particular application requires a different setup, RomBug can be rebuilt with a customized `initext` module to fit the application's requirements.

Table 4-1 MVME172 Model IP Slots

IP Slot	Memory Base	Interrupt Level
A	C0000000	5
B	C0800000	4

All slots are set for:

Memory Size	8M Bytes
Memory Width	16 Bits
Interrupts	Level sensitive, Normal Polarity
Recovery Time	0 microseconds

Table 4-2 MVME172 FX Models and Processor Information

Mode	Processor Type and Speed	DRAM Size
MVME172-413y	64 MHz MC68LC060	4MB DRAM
MVME172-433y	64 MHz MC68LC060	8MB DRAM
MVME172-453y	64 MHz MC68LC060	16MB DRAM
MVME172-513y	60 MHz MC68060	4MB DRAM
MVME172-523y	60 MHz MC68060	8MB DRAM
MVME172-533y	60 MHz MC68060	16MB DRAM

Motorola has numerous variations of the MVME172board. These boards are differentiated by FX vs. LX model, the amount and type of RAM memory installed, the type of processor, and support for SCSI and Ethernet on the board. A representative list of model numbers are listed in [Table 4-3](#).

Table 4-3 MVME172LX Models and Processor Information

Model	Processor Type and Speed	DRAM Size and Type
MVME172-213	68LC060 64MHz	4MB Parity
MVME172-223	68060 60MHz	4MB Parity
MVME172-233	68LC060 64MHz	4MB ECC
MVME172-243	68060 60MHz	4MB ECC
MVME172-253	68LC060 64MHz	16MB ECC
MVME172-263	68060 60MHz	16MB ECC
MVME172-303	68060 60MHz	8MB Parity
MVME172-313	68LC060 64MHz	8MB Parity
MVME172-323	68060 60MHz	8MB ECC
MVME172-333	68LC060 64MHz	8MB ECC
MVME172-343	68060 60MHz	32MB ECC
MVME172-353	68LC060 64MHz	32MB ECC
MVME172-363	68060 60MHz	16MB Parity
MVME172-373	68060 60MHz	16MB Parity

Figure 4-1 MVME172FX Jumper Block Locations

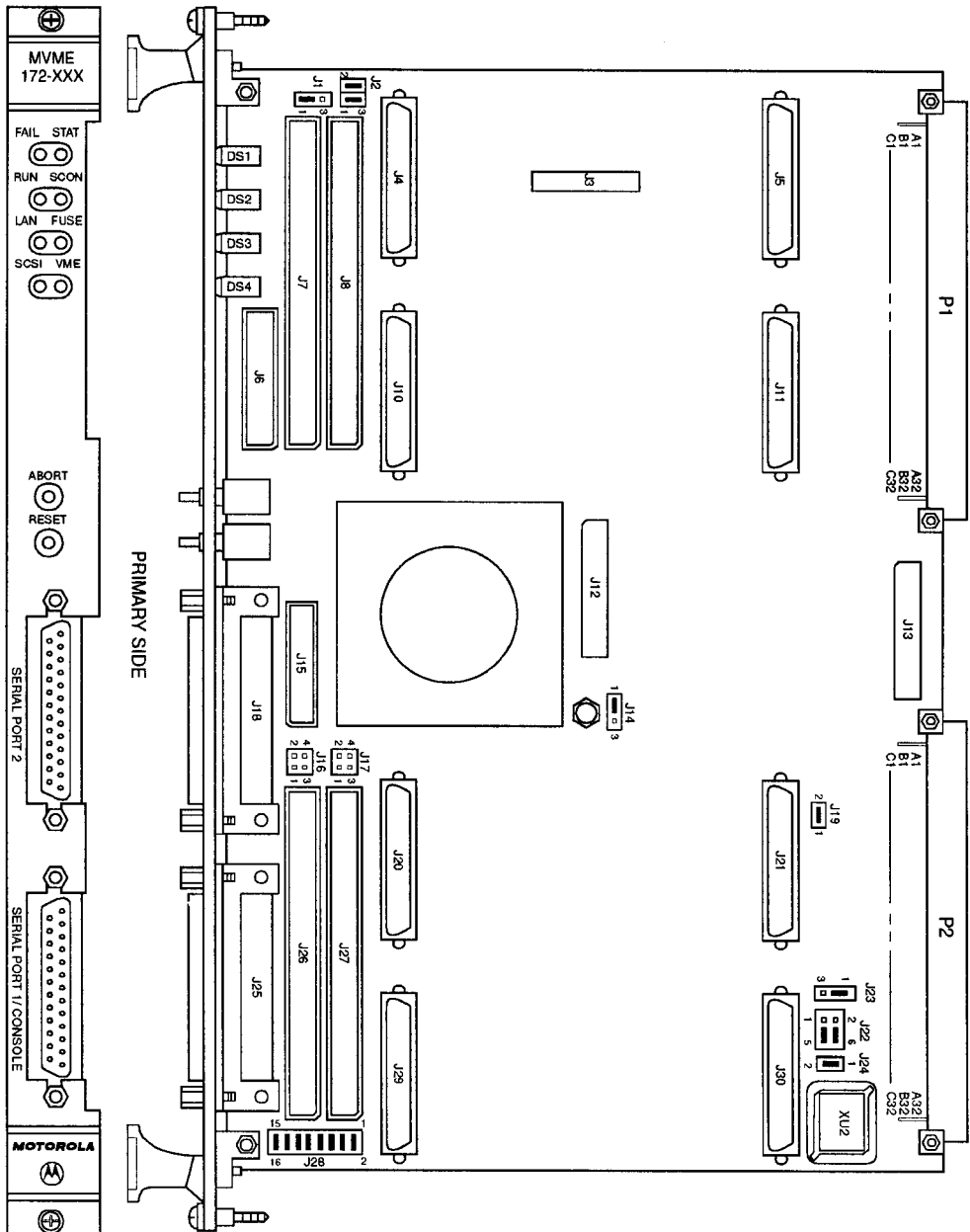


Figure 4-2 MVME172LX Jumper Block Locations

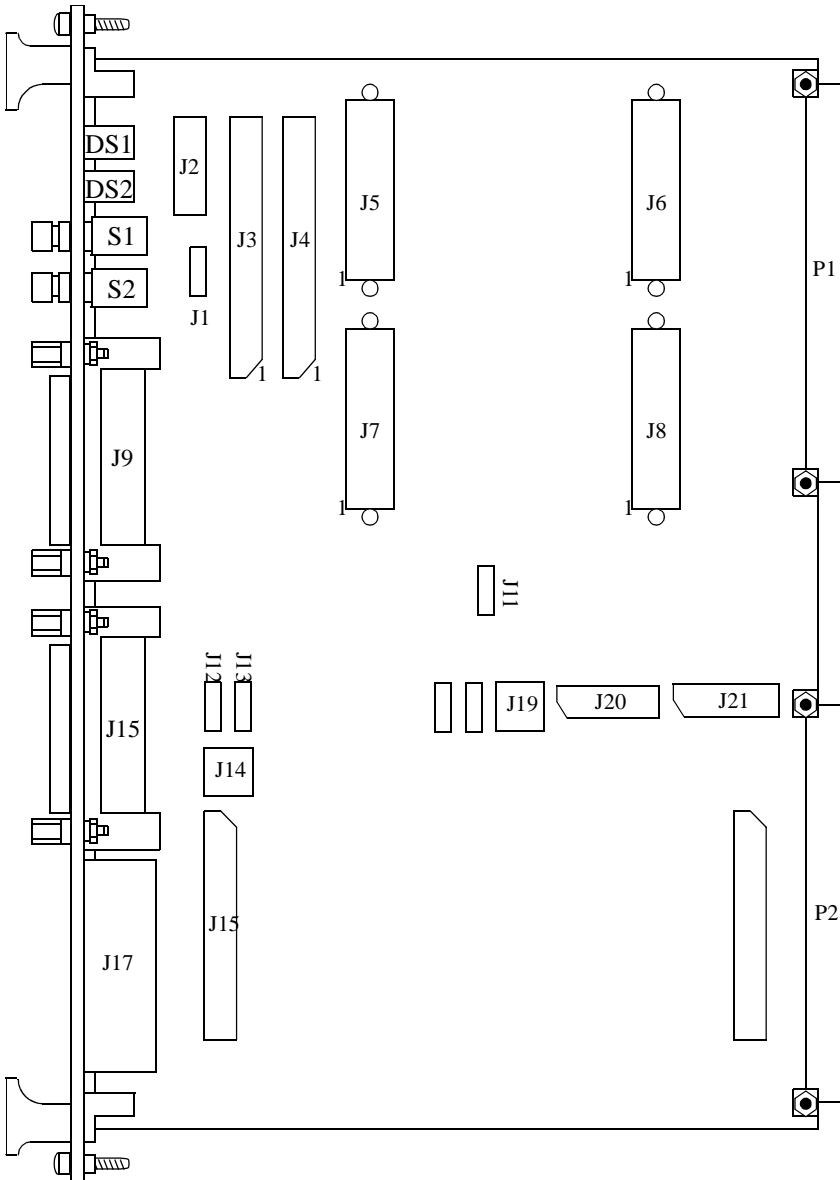
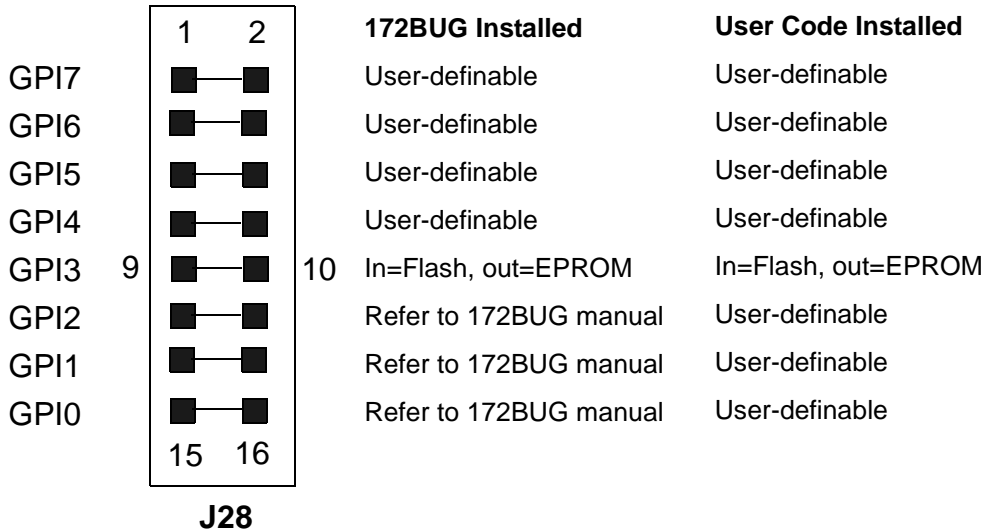
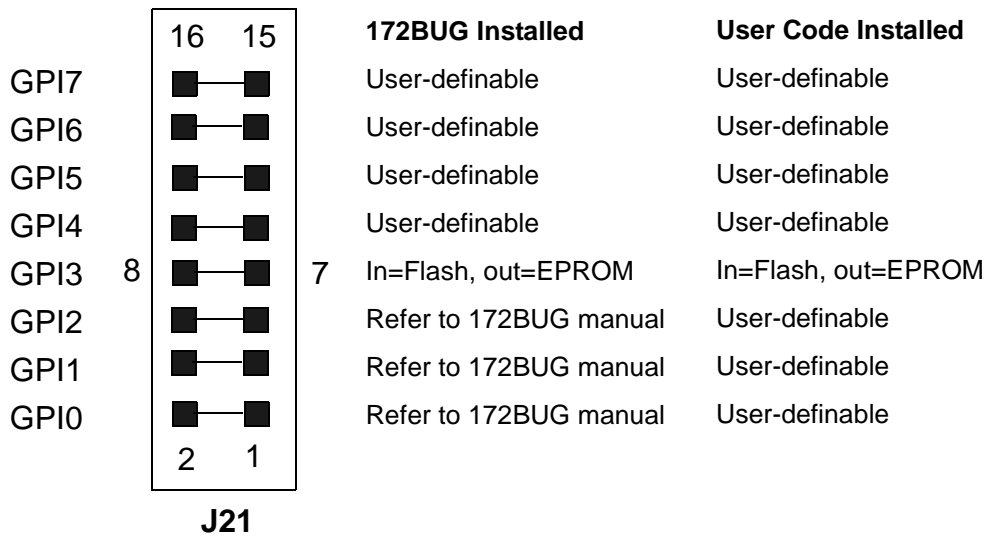


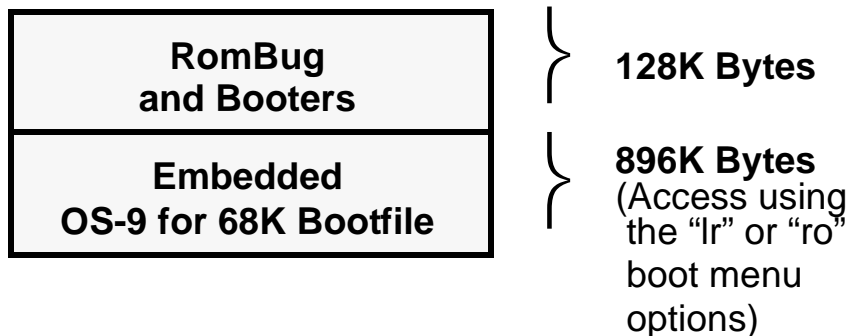
Figure 4-3 MVME172FX Jumper Settings

Figure 4-4 MVME172LX Jumper Settings


ROM Configuration and Organization

On the MVME172, either flash memory or the PROM socket can be selected as the primary ROM bank with a jumper. Typically, Motorola ships the MVME172 with a PROM containing 172Bug and with 172Bug residing in flash. The 172Bug includes a facility for reprogramming flash, but it must run from the PROM (as the primary bank), not flash, to operate properly. All MVME172 BLSs include a RomBug image to put in flash or programmed to an EPROM that can be installed onto the board in socket XU2.

The address space of the MVME172 RomBug PROM is divided into the two areas illustrated in [Figure 4-5](#).

Figure 4-5 Address Space of the MVME172 RomBug PROM



Board Specific OS-9 Modules

The software environment for the MVME172 is built in its `PORTS` directory structure found at `/h0/MWOS/OS9/68060/PORTS/MVME172`. All descriptors, `init` modules, and bootfiles are configured and built in this directory and its subdirectories.

The board-specific drivers available for the MVME172 include:

<code>tk172</code>	System ticker
<code>rtc172</code>	Real-time clock driver
<code>sc172</code>	SCF/serial port driver
<code>scsi172</code>	SCSI low-level driver
<code>sp172</code>	Ethernet driver

Other system modules available for use on the board residing in common directories include:

<code>cache060</code>	Cache control module
<code>ssm060</code>	SSM for the 68060 PMMU
<code>ssm060_cbsup</code>	Same as above with cache copyback for supervisor state
<code>fpsp060</code>	Floating point support package for 68060 board versions
<code>fpu</code>	Floating point emulation for 68LC060 board versions

Installing the MVME172 CPU Module (EPROM Method)

- Step 1. Remove power from the system and eject the MVME172 CPU module.
- Step 2. Install the PROM(s).
On FX models insert the EPROM in socket XU2.
On LX models insert the EPROM in socket XU1
IMPORTANT: Retain the 172Bug EPROM for future use.
On the LX172, the EPROM can be placed in XU2.

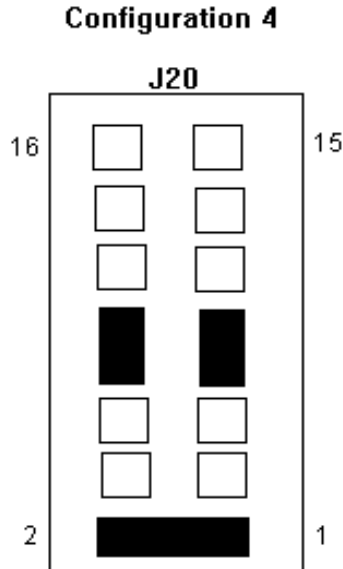


Note

The 32-pin PROM used with the MVME172FX requires special considerations when inserting or removing and during handling. A special extraction tool is required to remove the device from its socket. When handling and inserting the PROM, be careful not to damage or bend the pins on the PROM.

- Step 3. Set EPROM Jumpers:
 On FX models set jumper J23 to connect pins 1 and 2
 On LX models set jumpers J20 to Configuration 4, shown in the figure below:

Figure 4-6 Configuration 4



1Mx8 EPROMs On Board Flash Disabled

- Step 4. (Optional) Select the PROM as the reset bank.
 On FX models Remove the jumper at J22 pins 9 & 10. Save the jumper by installing it only on pin 10.



Note

The MVME172 can be reset from either the 172Bug in flash memory or the RomBug in PROM. The jumper at J28 pins 8 & 9 on FX models and at J21 pins 9 & 10 on LX models switches the address space between these two devices. Once 172Bug has been coldstarted, RomBug can be **booted** from the 172Bug by configuring 172Bug's environment as described in a later section of this manual.

Step 5. Examine and change jumpers on the CPU module.

In general, except for EPROM size, the jumper settings for the MVME172 module should remain the same as the factory default settings (or configured specifically for your needs).

Step 6. Reinstall the MVME172CPU module.

Step 7. Complete hardware installation.

Once the PROM is installed and jumper configuration is completed, carefully read the ***MVME172 Embedded Controller User's Manual*** to complete the hardware installation steps required for the P2 Adaptor card (if used) and the MVME712M Transition Module.



Note

Read the Motorola supplied documentation to ensure all hardware cabling requirements are correctly satisfied.

Installing the MVME172 CPU Module (Flash programming method)



Note

If you are downloading RomBug over a serial interface, do not perform all of the steps below. Instead, perform steps one through six; once the file transfer is complete, perform steps 12 and 13 to complete the installation. For more information on this process, refer to [Appendix A: Downloading RomBug from a Host over a Serial Interface](#).

Step 1. Remove power from the system and eject the MVME172 CPU module.

Step 2. Verify EPROM installation.

Motorola's 172Bug EPROMS are shipped with the MVME172 board. 172Bug will be used to install the OS-9 RomBug image into the MVME172 board's flash memory. The EPROMs may be disabled with jumper J8, once the RomBug image is stored in flash.

Step 3. Set jumpers

For installation, the jumper settings for the MVME172 should be set to boot from EPROM (FX Models J28--pins 9 and 10, LX models J21--pins 7 and 8). Reinsert the MVME172 CPU module into the system.

Step 4. Reset the system by powering up or pressing the reset switch. This brings up the 172Bug prompt. (You may need to press **break** to bring up the 172Bug prompt.)

Step 5. Ensure the 172Bug is configured to **Probe system for supported I/O controllers**. (**Y**) through the `ENV` command.

- Step 6. Ensure the time-of-day clock is started with the `SET` command (for example, `SET <mmddyymm>`). The `TIME` command can confirm the correct setting and operation.
- Step 7. Obtain an IP address for the MVME172 board. Determine the assigned IP address for the cross-development `host/bootp/tftp` server.
- Step 8. Use the `NPING` command to confirm connectivity between the MVME172 board and the host system.
- Step 9. Transfer the file:

```
<mwos>/OS9/68060/PORTS/MVME172/CMS/BOOTOBS/ROMBUG/rombugger
```

to the `tftp` server.
- Step 10. Use the `NIOT` (Network I/O Teach) command to configure the network driver with the correct IP addresses.
- Step 11. Initiate transfer of the RomBug image with the `NIOP` (Network I/O Physical) command. Specify a read with the `rombugger` file path to a load address of 100000. If an error occurs, repeat the transfer until successful.
- Step 12. When the previous command is complete, enter the command to program the image into flash:

```
172-Bug>PFLASH 100000:100000 FFA00000;B
```
- Step 13. When programming is complete, enable ROM booting by entering the `ENV` command. Specify the starting address as `FFA00400` and the ending address as `FFA00500`. You should include a delay count of three (3) seconds to give yourself time to interrupt an automatic ROM boot to regain access to the 172Bug debugger.



For More Information

This is explained in more detail in the next section, [Configuring 172Bug to Boot RomBug](#).

Configuring 172Bug to Boot RomBug

Motorola's 172Bug, which resides in flash can be configured to automatically boot RomBug using its ROMboot feature. This allows resetting into the 172Bug in order to keep the diagnostics handy, then automatically transferring control to RomBug to bring up OS-9. However, this method is slower than resetting directly into RomBug.

Configuring 172Bug to Boot RomBug

Step 1. Verify reset into 172Bug.

Ensure 172Bug is installed in flash, the jumper at J21 pins 7 & 8 is installed, and the RomBug PROM is installed in the socket.

Step 2. Start up the system.

Reinstall the CPU module and reset the system to bring up the 172-Bug> prompt:

```
Copyright Motorola Inc. 1988 - 1997, All Rights Reserved
```

```
MVME172 Debugger/Diagnostics Release Version 1.2 - 01/21/97  
COLD Start
```

```
Local Memory Found =02000000 (&33554432)
```

```
WARNING: Local Memory Configuration Status (00000000)
```

```
MPU Clock Speed =60Mhz
```

```
172-Bug>
```

Step 3. Enable the ROMboot feature.

Enter the ENV command at the 172-Bug> prompt entering the following values at the configuration field prompt:

```
172-Bug> ENV<return>  
Bug or System environment [B/S] = B? <return>  
Field Service Menu Enable [Y/N] = N? <return>  
Remote Start Method Switch [G/M/B/N] = B? N<return>
```

```

Probe System for Supported I/O Controllers [Y/N] = Y? N<return>
Negate VMEbus SYSFAIL* Always [Y/N] = N? <return>
Local SCSI Bus Reset on Debugger Startup [Y/N] = N? <return>
Local SCSI Bus Negotiations Type [A/S/N] = A? <return>
Industry Pack Reset on Debugger Startup [Y/N] = Y? N<return>
Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y? <return>
Auto Boot Enable [Y/N] = N? <return>
Auto Boot at power-up only [Y/N] = Y? <return>
Auto Boot Controller LUN = 00? <return>
Auto Boot Device LUN = 00? <return>
Auto Boot Abort Delay = 15? <return>
Auto Boot Default String [NULL for a empty string] = ? <return>
ROM Boot Enable [Y/N] = N? Y<return>
ROM Boot at power-up only [Y/N] = Y? N<return>
ROM Boot Enable search of VMEbus [Y/N] = N? <return>
ROM Boot Abort Delay = 0? 5<return>
ROM Boot Direct Starting Address = FF800000? FFA00400<return>
ROM Boot Direct Ending Address = FFDF0000? FFA00500<return>
Network Auto Boot Enable [Y/N] = N? <return>
Network Auto Boot at power-up only [Y/N] = Y? <return>
Network Auto Boot Controller LUN = 00? <return>
Network Auto Boot Device LUN = 00? <return>
Network Auto Boot Abort Delay = 5? <return>
Network Auto Boot Configuration Parameters Pointer (NVRAM) = 00000000?
FFE10000<return>
Memory Search Starting Address = 00000000? FFE00000<return>
Memory Search Ending Address = 02000000? FFE10000<return>
Memory Search Increment Size = 00010000? <return>
Memory Search Delay Enable [Y/N] = N? <return>
Memory Search Delay Address = FFFFD20F? <return>
Memory Size Enable [Y/N] = Y? <return>
Memory Size Starting Address = 00000000? <return>
Memory Size Ending Address = 02000000? <return>
Base Address of Dynamic Memory = 00000000? <return>
Size of Parity Memory = 00000000? <return>
Size of ECC Memory Board #0 = 02000000? <return>
Size of ECC Memory Board #1 = 00000000? <return>
Base Address of Static Memory = FFE00000? <return>
Size of Static Memory = 00020000? <return>
Slave Enable #1 [Y/N] = Y? N<return>
Slave Starting Address #1 = 00000000? <return>
Slave Ending Address #1 = 01FFFFFF? <return>
Slave Address Translation Address #1 = 00000000? <return>
Slave Address Translation Select #1 = 00000000? <return>
Slave Control #1 = 03FF? <return>
Slave Enable #2 [Y/N] = N? <return>
Slave Starting Address #2 = 00000000? <return>
Slave Ending Address #2 = 00000000? <return>
Slave Address Translation Address #2 = 00000000? <return>
Slave Address Translation Select #2 = 00000000? <return>
Slave Control #2 = 0000? <return>
Master Enable #1 [Y/N] = Y? N<return>
Master Starting Address #1 = 02000000? <return>
Master Ending Address #1 = EFFFFFFF? <return>

```

```

Master Control #1 = 0D? <return>
Master Enable #2 [Y/N] = N? <return>
Master Starting Address #2 = 00000000? <return>
Master Ending Address #2 = 00000000? <return>
Master Control #2 = 00? <return>
Master Enable #3 [Y/N] = N? <return>
Master Starting Address #3 = 00000000? <return>
Master Ending Address #3 = 00000000? <return>
Master Control #3 = 00? <return>
Master Enable #4 [Y/N] = N? <return>
Master Starting Address #4 = 00000000? <return>
Master Ending Address #4 = 00000000? <return>
Master Address Translation Address #4 = 00000000? <return>
Master Address Translation Select #4 = 00000000? <return>
Master Control #4 = 00? <return>
Short I/O (VMEbus A16) Enable [Y/N] = Y? N<return>
Short I/O (VMEbus A16) Control = 01? <return>
F-Page (VMEbus A24) Enable [Y/N] = Y? N<return>
F-Page (VMEbus A24) Control = 02? <return>
ROM Access Time Code = 04? <return>
FLASH Access Time Code = 03? <return>
MCC Vector Base = 05? <return>
VMEC2 Vector Base #1 = 06? <return>
VMEC2 Vector Base #2 = 07? <return>
VMEC2 GCSR Group Base Address = D2? <return>
VMEC2 GCSR Board Base Address = 00? <return>
VMEbus Global Time Out Code = 01? <return>
Local Bus Time Out Code = 02? <return>
VMEbus Access Time Out Code = 02? <return>
IP A Base Address = 00000000? <return>
IP B Base Address = 00000000? <return>
IP C Base Address = 00000000? <return>
IP D Base Address = 00000000? <return>
IP D/C/B/A Memory Size = 00000000? <return>
IP D/C/B/A General Control = 00000000? <return>
IP D/C/B/A Interrupt 0 Control = 00000000? <return>
IP D/C/B/A Interrupt 1 Control = 00000000? <return>

Update Non-Volatile RAM (Y/N)? Y<return>

Reset Local System (CPU) (Y/N)? Y<return>

```

The system resets twice after responding to the last prompt (each time with a three second delay during which the operator may abort the ROMboot process by pressing **<break>**), then RomBug is entered. Resetting from RomBug brings the operator back to 172Bug with another opportunity to interrupt the automatic ROMboot. Once configured, RomBug can manually be booted from 172Bug by entering the RB command at the 172-Bug> prompt.



Note

All of the above 172Bug entries are explained in the *172Bug User's Manual*.



Note

An ABORT-RESET sequence disables an automatic ROM Boot sequence. As a result, it cannot be used alone to force an OS-9 RomBug reconfiguration session when 172Bug is in the primary ROM bank. To force a reconfiguration session from 172Bug, press and hold the ABORT switch, enter the RB command, wait at least as long as the ROMboot abort delay, and then release the ABORT switch.

Chapter 5: MVME177 Reference

This section contains information about the MVME177 models. It includes the following sections:

- **MVME177 Quick Reference**
- **Overview of the MVME177 CPU Module**
- **ROM Configuration and Organization**
- **Board Specific OS-9 Modules**
- **Installing RomBug into Flash**
- **Configuring 177Bug to Boot RomBug**



For More Information

This chapter details the jumper and processor locations for all of the MVME177 models, with the exception of Petra. For information on the Petra model, refer to the ***Motorola MVME VME Embedded Controller Installation and Use Guide*** shipped with your hardware.



MVME177 Quick Reference

Ports Directory: <mwos>/OS9/68060/PORTS/MVME177

Console Port Location: MVME712, connector labeled “console”

OS-9 Prom Size: 1 megabyte, always programmed into Flash.

ROM Booting Search Order:

FF800400:FF808400

FFA00400:FFA08400

BootP Ethernet booting device: ie

SPF Ethernet descriptor name: spie0

CMDS Build sequence:

<mwos>/OS9/68000/CMDS

<mwos>/OS9/68020/CMDS

<mwos>/OS9/68060/CMDS

<mwos>/OS9/68060/PORTS/MVME177/CMDS

Overview of the MVME177 CPU Module

The MVME177 CPU module provides:

- 68060 processor running at 50MHz
- 4, 8, 16, 32, 64, or 128MB of on-board DRAM
- 128KB of on-board Static RAM
- 4 RS-232 serial ports
- 8-bit bi-directional parallel port
- On-board SCSI bus interface with DMA
- On-board Ethernet transceiver interface with DMA
- Time-of-day clock/calendar with battery backup
- 8KB by 8 battery backed CMOS RAM (NVRAM)
The OS-9 for 68K area of NVRAM is the first 256 bytes of the user area beginning at FFFC0000.
- Two 44-pin PLCC (16-bit) EPROM sockets organized as one bank of 32-bit wide (high and low word) memory
- 4MB flash memory in four Intel 28F0085A chips with software control write protection

Motorola has numerous variations of the board. These boards are differentiated by the amount of RAM memory installed. A representative list of module numbers are listed in [Table 5-1](#).

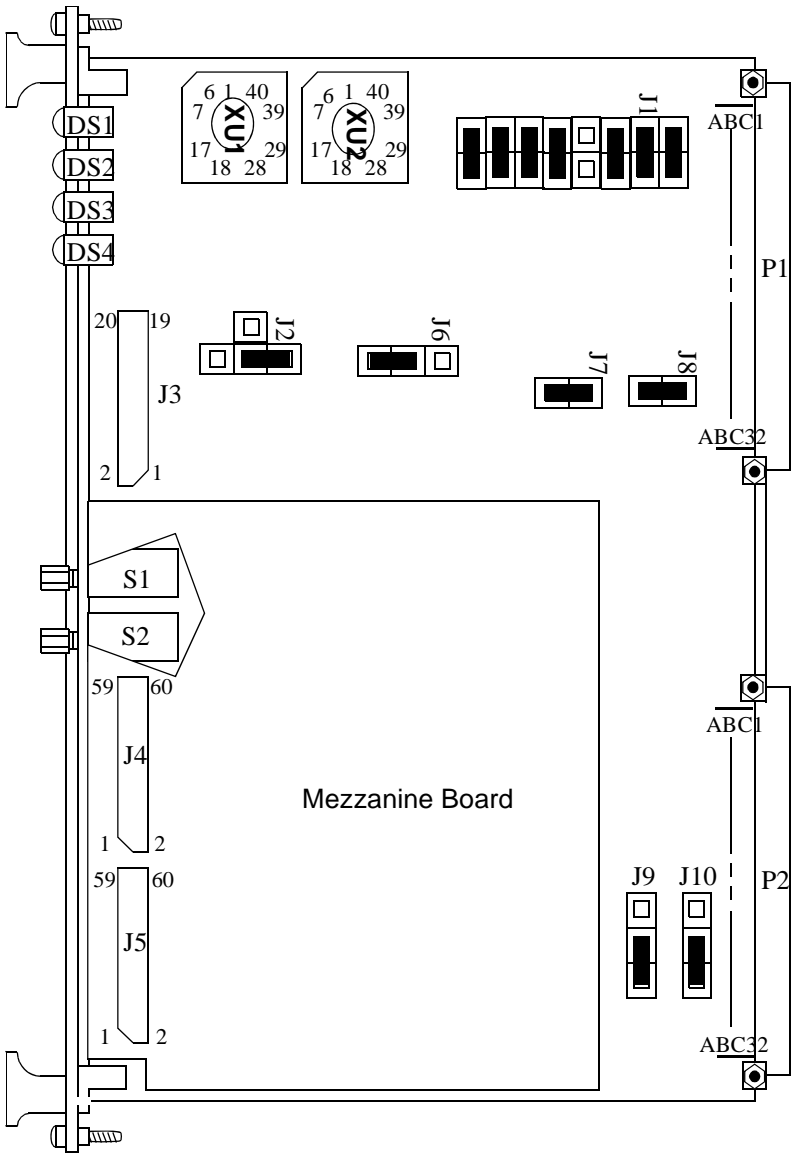
Table 5-1 MVME177 Models and Memory Size

CPU Model	CPU Speed	Memory Size
MVME177-001	50 MHz	4M
MVME177-002	50 MHz	8M
MVME177-003	50 MHz	16M

Table 5-1 MVME177 Models and Memory Size (continued)

CPU Model	CPU Speed	Memory Size
MVME177-004	50 MHz	32M
MVME177-005	50 MHz	64M
MVME177-006	50 MHz	128M
MVME177-011	60 MHz	4M
MVME177-012	60 MHz	8M
MVME177-013	60 MHz	16M
MVME177-014	60 MHz	32M
MVME177-015	60 MHz	64M
MVME177-016	60 MHz	128M

Figure 5-1 MVME177 Jumper Block Locations



ROM Configuration and Organization

There are two possible ROM configurations for the MVME177. As the system installer, you must choose which ROM configuration to adopt, depending on your system application.

MVME 177 has one bank of ROM sockets and a 4MB bank of flash memory split into two halves, upper and lower. You can set the jumpers to allow two different configurations:

1. The ROM responds as the primary ROM bank and one of the two flash halves (under software configuration responds as the secondary ROM bank).
2. The lower half of the flash responds as the primary ROM bank and the upper half of the flash responds as the secondary ROM bank.
 - The first retains the 177Bug ROMs in the primary bank (for ease of hardware diagnostic testing) and adds the OS-9 RomBug image in the secondary bank. In this configuration, 177Bug gets control on a reset and transfers control to RomBug in the secondary bank



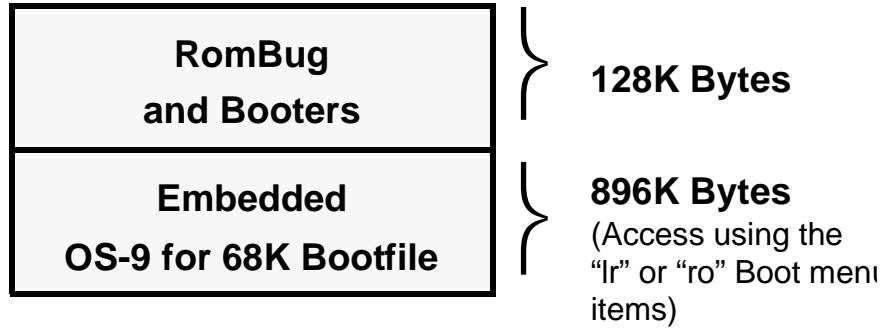
For More Information

See the [Configuring 177Bug to Boot RomBug](#) section in this chapter for more information.

- The second configuration is to program the OS-9 RomBug image into the primary flash bank. In this configuration, RomBug always gets control on a reset. The secondary flash bank can be used for additional boot modules or can be searched by the kernel for additional modules by adjusting the init module's search lists. The jumpers must be set to select flash memory for the primary ROM address.

The address space of the MVME177 RomBug image is divided into the three areas illustrated in **Figure 5-2**.

Figure 5-2 Address Space of the MVME177 RomBug



Board Specific OS-9 Modules

The board specific drivers available for the MVME177 include:

tk177	System ticker
rtc177	Real-time clock driver
sc177	SCF/serial port driver
scp177	SCF/parallel port driver
scsi177	SCSI low-level driver
sp177	Ethernet driver

Other system modules available for use on the board residing in common directories include:

cache060	Cache control module
ssm060	SSM for the 68060 PMMU
ssm060_cbsup	Same as above with cache copyback for supervisor state
fpsp060	Floating point support package for 68060 CPU chips

The MVME177 contains the I82595 Ethernet chip. The MVME177 BLS provides the functionality to communicate over the network to other systems.

System Ethernet modules can be found in the following directory:

MWOS/OS9/68060/PORTS/MVME177/CMDS/BOOTOBS/SPF

spie0	i82596 descriptor
sp177	i82595 driver
inetdb	Data module containing network information such as host, networks, services, protocols, inetd.conf, and resolv.conf

Installing the MVME177 CPU Module

Step 1. Remove power from the system and eject the CPU module.

Step 2. Verify EPROM installation.

Motorola's 177Bug is shipped with the MVME177 board and installed in EPROM in sockets XU1 and XU2. 177Bug is used to install the OS-9 RomBug image into the MVME177 board's flash memory. The EPROMs may be disabled with jumper J8, once the RomBug image is stored in flash.



For More Information

Refer to the [Installing RomBug into Flash](#) section of this manual for more information.

Verify the factory supplied EPROMs are installed in sockets XU1 and XU2.

Step 3. Verify jumper settings.

In general, the jumper settings for the MVME177 board should remain the same as the factory default settings (or configured specifically for your needs).

Figure 5-3 MVME177 Jumper Settings

Software Readable Header:

Reserved for
 177Bug: GPI0, GPI1, GPI2, GPI3
 Available: GPI4, GPI5, GPI6, GPI7
 Factory Default: GPI3 only removed

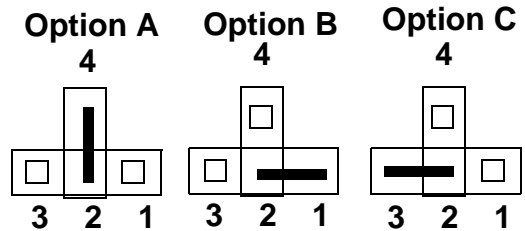
Jumper Block J1



SRAM Backup Power Source Select:

Possible configurations:
A = Backup Power Disable
B = VMEbus +5V STBY, backup off backplane, factory configuration
C = Backup from Battery

Jumper Block J2

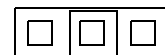
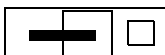


WARNING

Do not remove all of the jumpers from J2 or your SRAM becomes disabled.

System Control Selection

Jumper Block J6



System Controller
 (Factory Configuration)

Auto System Controller

Not System Controller

Figure 1-3 MVME177 Jumper Settings (continued)

Thermal Sensing Pins

Jumper Block J7

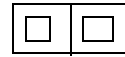
The thermal sensing pins, THERM1 and THERM2, are connected to an internal thermal resistor to provide information about the average temperature of the processor. Refer to the *M68000 Microprocessors User's Manual* for additional information on the use of these pins.

EPROM/Flash Configuration



1MB PROM and 2MB Flash enabled
(Factory Configuration)

Jumper Block J8:



4MB Flash enabled

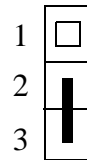
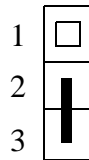
Serial Port 4 Clock Configuration Select Headers:

J9: Jumper 2-3 to receive RTXC4 (Factory Configuration)
J10: Jumper 2-3 to receive TRXC4 (Factory Configuration)

J9: Jumper 1-2 to drive RTXC4
J10: Jumper 1-2 to drive TRXC4

Jumper Block J9

Jumper Block J10



- Step 4. Reinsert the CPU module into the system.
- Step 5. Complete hardware installation.



For More Information

Carefully read the ***MVME177 Single Board Computer Installation and User Manual*** (supplied by Motorola) to complete the hardware installation steps required for the P2 Adaptor Card (if used) and the MVME712M Transition Module. These instructions also detail the various options available for cabling SCSI devices to the MVME712 module.



Note

Jumper installation details for the MVME712M module are supplied in the support module appendix. However, carefully read the Motorola supplied documentation to ensure all hardware cabling requirements are correctly satisfied.

Step 6. Install RomBug.

This BSP is supplied with a RomBug image to be installed in the flash memory. Follow the procedure in the next section, [Installing RomBug into Flash](#) if you are going to put the RomBug image in flash

Installing RomBug into Flash

Before booting OS-9 for 68K, a copy of Microware's RomBug debugger must be programmed into the MVME177 board's flash memory. The PROM-resident Motorola debugger (177Bug) has the capability to read the images off of magnetic media (or download them from cross-development platforms) and program the image into flash memory.

Downloading RomBug from a Host over an Ethernet Network



Note

If you are downloading RomBug over a serial interface, do not perform all of the steps listed below. Instead, perform step three; once the file transfer is complete, perform steps nine and ten to complete the installation. For more information on this process, refer to [Appendix A: Downloading RomBug from a Host over a Serial Interface](#).

- Step 1. Reset the system by powering up or pressing the reset switch. This brings up the 177Bug prompt. (You may need to press <break> to bring up the 177Bug prompt.)
- Step 2. Ensure 177Bug is configured to **Probe system for supported I/O controllers**. (Y) through the ENV command.
- Step 3. Ensure the time-of-day clock is started with the SET command (for example, SET <yyymmddhhmm>). The TIME command can confirm the correct setting and operation.
- Step 4. Obtain an IP address for the MVME177 board. Determine the assigned IP address for the cross-development host/bootp/tftp server.

- Step 5. Use the `NPING` command to confirm connectivity between the MVME177 board and the host system.
- Step 6. Use the `NIOT` (Network I/O Teach) command to configure the network driver with the correct IP addresses.
- Step 7. Move the file:
`MWOS/OS9/68060/PORTS/MVME177/CMDS/BOOTOBS/ROMBUG/rombugger` to the tftp server.
- Step 8. Initiate transfer of the RomBug image with the `NIOP` (Network I/O Physical) command. Specify a read with the `rombugger` file path to a load address of 100000. If an error occurs, repeat the transfer until successful.
- Step 9. When the previous command is complete, enter the command to program the image into flash:
- ```
177-Bug>PFLASH 10000:10000 FFA0000;B
```
- Step 10. When programming is complete, enable ROM booting by entering the `ENV` command. Specify the starting address as `FFA00400` and the ending address as `FFA00500`. You should include a delay count of three (3) seconds to give yourself time to interrupt an automatic ROM boot to regain access to the 177Bug debugger.



---

## For More Information

This is explained in more detail in the next section, [Configuring 177Bug to Boot RomBug](#).

---

## Configuring 177Bug to Boot RomBug

---

Motorola's 177Bug which resides in EPROM can be configured to automatically boot RomBug using its ROMboot feature. This allows coldstarting from the 177Bug in order to keep the diagnostics handy, then automatically transferring control to RomBug to bring up OS-9. However, this method is much slower than coldstarting directly into RomBug.

### Configuring 177Bug to Boot RomBug

---

Step 1. Verify PROM installation.

Ensure 177Bug EPROMs are installed in the primary bank, and the RomBug image has been stored in the MVME177 board's flash memory.

Step 2. Start up the system.

Reinstall the CPU module and reset the system to bring up the 177-Bug> prompt:

```
Copyright Motorola Inc. 1988 - 1993, All Rights Reserved
MVME177 Debugger/Diagnostics Release Version 1.5 - 02/14/93
COLD Start
Local Memory Found =03000000 (&50331648)
MPU Clock Speed =33Mhz
177-Bug>
```

Step 3. Enable the ROMboot feature.

Enter the ENV command at the 177-Bug> prompt entering the following values at the configuration field prompt:

```
177-Bug>ENV<return>
Bug or System environment [B/S] = B? <return>
Field Service Menu Enable [Y/N] = N? <return>
Remote Start Method Switch [G/M/B/N] = B? <return>
Probe System for Supported I/O Controllers [Y/N] = Y? N<return>
Negate VMEbus SYSFAIL* Always [Y/N] = N? <return>
Local SCSI Bus Reset on Debugger Startup [Y/N] = N? <return>
Local SCSI Bus Negotiations Type [A/S/N] = A? <return>
```

```

Ignore CFGA Block on a Hard Disk Boot [Y/N] = Y? <return>
Auto Boot Enable [Y/N] = N? <return>
Auto Boot at power-up only [Y/N] = Y? <return>
Auto Boot Controller LUN = 00? <return>
Auto Boot Device LUN = 00? <return>
Auto Boot Abort Delay = 15? <return>
Auto Boot Default String [NULL for a empty string] = ? <return>
ROM Boot Enable [Y/N] = N? Y<return>
ROM Boot at power-up only [Y/N] = Y? N<return>
ROM Boot Enable search of VMEbus [Y/N] = N? <return>
ROM Boot Abort Delay = 0? 3<return>
ROM Boot Direct Starting Address = FF800000? FFA00400<return>
ROM Boot Direct Ending Address = FFDFFFFC? FFA00500<return>
Network Auto Boot Enable [Y/N] = N? <return>
Network Auto Boot at power-up only [Y/N] = Y? <return>
Network Auto Boot Controller LUN = 00? <return>
Network Auto Boot Device LUN = 00? <return>
Network Auto Boot Abort Delay = 5? <return>
Network Auto Boot Configuration Parameters Pointer (NVRAM) = 00000000? <return>
Memory Search Starting Address = 00000000? FFE00000<return>
Memory Search Ending Address = 00010000? FFE10000<return>
Memory Search Increment Size = 00010000? <return>
Memory Search Delay Enable [Y/N] = N? <return>
Memory Search Delay Address = FFFFCE0F? <return>
Memory Size Enable [Y/N] = Y? <return>
Memory Size Starting Address = 00000000? <return>
Memory Size Ending Address = 0xx00000? <return>
Base Address of Local Memory = 00000000? <return>
Size of Local Memory Board #0 = 0xx00000? <return>
Size of Local Memory Board #1 = 0xx00000? <return>
Slave Enable #1 [Y/N] = Y? N<return>
Slave Starting Address #1 = 00000000? <return>
Slave Ending Address #1 = 0xxFFFFFF? <return>
Slave Address Translation Address #1 = 00000000? <return>
Slave Address Translation Select #1 = Fxx00000? <return>
Slave Control #1 = 03FF? <return>
Slave Enable #2 [Y/N] = N? <return>
Slave Starting Address #2 = 00000000? <return>
Slave Ending Address #2 = 00000000? <return>
Slave Address Translation Address #2 = 00000000? <return>
Slave Address Translation Select #2 = 00000000? <return>
Slave Control #2 = 0000? <return>
Master Enable #1 [Y/N] = Y? N<return>
Master Starting Address #1 = 0xx00000? <return>
Master Ending Address #1 = 00000000? <return>
Master Control #1 = 00? <return>
Master Enable #2 [Y/N] = N? <return>
Master Starting Address #2 = 00000000? <return>
Master Ending Address #2 = 00000000? <return>
Master Control #2 = 00? <return>
Master Enable #3 [Y/N] = N? <return>
Master Starting Address #3 = 00000000? <return>
Master Ending Address #3 = 00000000? <return>
Master Control #3 = 00? <return>

```

```

Master Enable #4 [Y/N] = N? <return>
Master Starting Address #4 = 00000000? <return>
Master Ending Address #4 = 00000000? <return>
Master Address Translation Address #4 = 00000000? <return>
Master Address Translation Select #4 = 00000000? <return>
Master Control #4 = 00? <return>
Short I/O (VMEbus A16) Enable [Y/N] = Y? N<return>
Short I/O (VMEbus A16) Control = 00? <return>
F-Page (VMEbus A24) Enable [Y/N] = Y? N<return>
F-Page (VMEbus A24) Control = 00? <return>
ROM Speed Bank A Code = 00? <return>
ROM Speed Bank B Code = 00? <return>
Static RAM Speed Code = 00? <return>
PCC2 Vector Base = 05? <return>
VMEC2 Vector Base #1 = 06? <return>
VMEC2 Vector Base #2 = 07? <return>
VMEC2 GCSR Group Base Address = CC? <return>
VMEC2 GCSR Board Base Address = 00? <return>
VMEbus Global Time Out Code = 01? <return>
Local Bus Time Out Code = 00? <return>
VMEbus Access Time Out Code = 02? <return>

```

Update Non-Volatile RAM (Y/N)? **Y**

Reset Local System (CPU) (Y/N)? **Y**

The system resets twice after responding to the last prompt (each time with a three second delay during which the operator may abort the ROMboot process by pressing <break>), then RomBug is entered. Resetting from RomBug brings the operator back to 177Bug debugger with another opportunity to interrupt the automatic ROMboot. Once configured, RomBug can manually be booted from 177Bug by entering the RB command at the 177-Bug> prompt.



## Note

All of the above 177Bug entries are explained in the *177Bug User's Manual* from Motorola.



---

**Note**

An `ABORT-RESET` sequence disables an automatic ROMboot sequence. As a result, it cannot be used alone to force an OS-9 RomBug reconfiguration session when 177Bug is in the primary ROM bank. To force a reconfiguration session from 177Bug, press and hold the `ABORT` switch, enter the `RB` command, wait at least as long as the ROM Boot abort delay, and then release the `ABORT` switch.

---



---

# Chapter 6: Peripheral and Transition Module Configurations

---

This chapter contains information about the MVME712 Transition Module. It includes the following topics:

- **Overview of the MVME712 Transition Module**
- **Configuring the MVME712 Transition Module**
- **Configuring the MVME050 System Controller Module for I/O Operations**



## Overview of the MVME712 Transition Module

---

The MVME712 transition module is a support module for the MVME147, MVME162 (original series models only), and MVME167 CPU modules. This transition module provides the connections for the serial ports, parallel printer port, SCSI bus, and Ethernet (for non-RF/-SRF MVME147 versions).



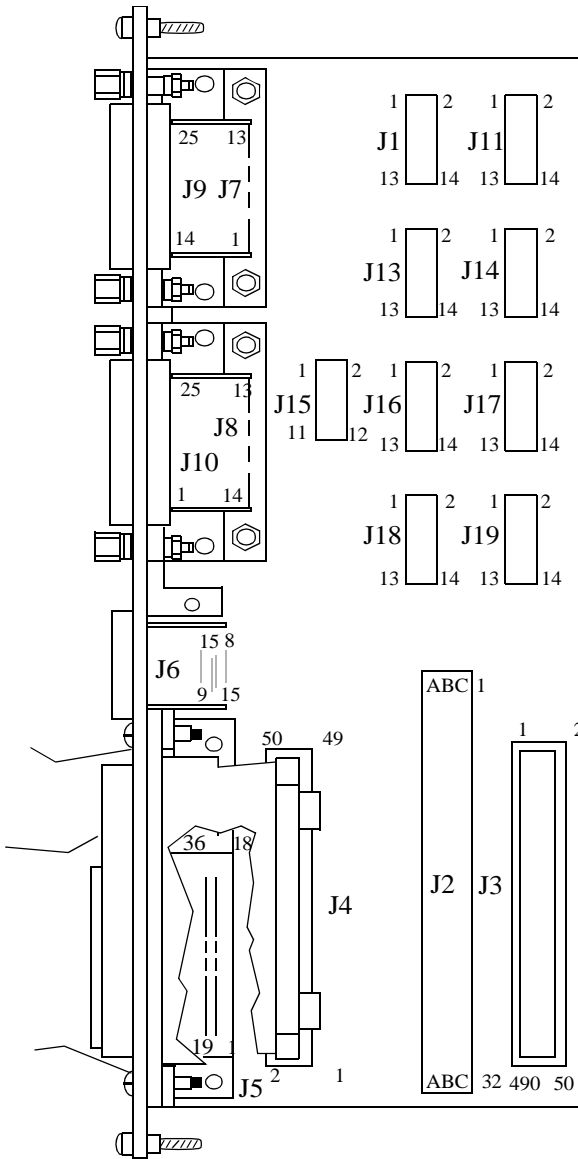
---

### Note

The *MVME147 User Manual* and the *MVME167 User Manual* contain detailed instructions for installing the MVME712 and the associated P2 adaptor module. Consult the appropriate Motorola supplied manuals to determine the correct cabling and physical mounting requirements for your desired configuration.

---

**Figure 6-1 MVME712 Jumper Block Locations**



## Configuring the MVME712 Transition Module

The following steps guide you through configuration of the MVME712 transition module.

Step 1. Remove the MVME712 module.

Remove power from the system, and eject the MVME712 module.

Step 2. Configure serial ports for DCE operation.

Serial Port 1 is controlled by jumpers J1 and J11. Serial Port 2 is controlled by jumpers J16 and J17. Serial Port 3 is controlled by jumpers J13 and J14. Serial Port 4 is controlled by jumpers J18 and J19.

### Figure 6-2 Jumper Settings

All four serial ports should be set for **To Terminal** (DCE) operation. Remove all jumpers from J11, J17, J14, and J19. Install jumpers in J1, J16, J13, and J18 as follows:

#### Jumper Block J1

|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|   |   |   |   |    |    |    |
| 1 | 3 | 5 | 7 | 9  | 11 | 13 |

#### Jumper Block J13

|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|   |   |   |   |    |    |    |
| 1 | 3 | 5 | 7 | 9  | 11 | 13 |

#### Jumper Block J16

|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|   |   |   |   |    |    |    |
| 1 | 3 | 5 | 7 | 9  | 11 | 13 |

#### Jumper Block J18

|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|   |   |   |   |    |    |    |
| 1 | 3 | 5 | 7 | 9  | 11 | 13 |

Step 3. Install P2 adaptor and SCSI cabling.

Once the serial ports are configured, install the P2 adaptor module. Run all cables from the P2 adaptor module to the MVME712 transition module, and cable the SCSI devices according to your system requirements.



---

### For More Information

The *MVME147 User Manual* and the *MVME167 User Manual* provide detailed instructions on how this is achieved.

---

Step 4. Configure and connect the disk system.



---

### For More Information

See the *OS-9 for 68K Processors BLS Reference* manual for information on OS-9 for 68K SCSI operation and disk formats.

---

Step 5. Replace the MVME712 module.

---

# Configuring the MVME050 System Controller Module for I/O Operations

---



## Note

Important notice for all Motorola VME CPU users:

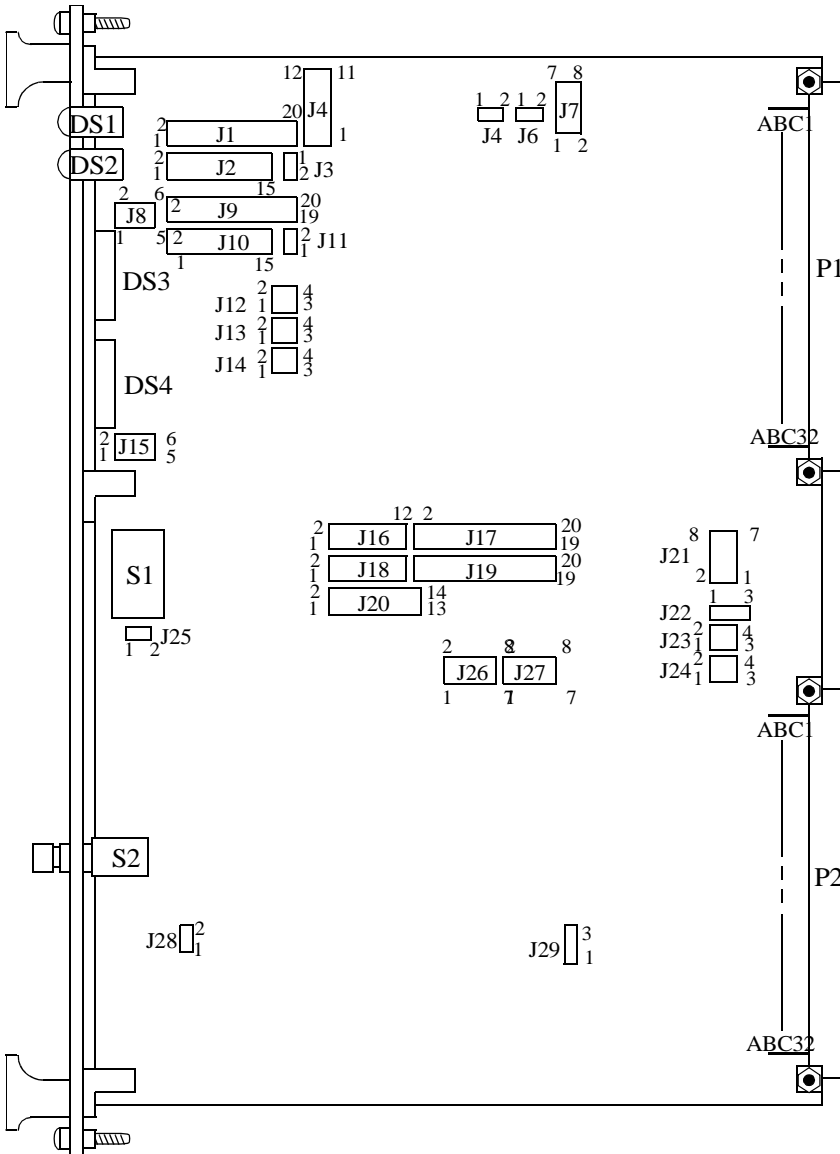
All Microware Development Packs for Motorola VME CPUs now support auto-detection of one Motorola MVME050 module with a base address of 0xffff1000. For CPUs with 24-bit addressing, the base address is 0x00ff1000.

If the MVME050 module is installed and detected, the Microware DevPak ROMs read the MVME050 front panel switches to determine the boot system configuration, overriding the boot method switches provided on the CPU. If this is not desired, you must change the base address of the MVME050 module to something other than what is specified by Microware.

If the ROM/RAM quads are not in use, disable them. Otherwise, ensure their addresses are unique within the VME bus memory map.

---

**Figure 6-3 MVME050 Jumper Block Locations**



The following steps guide you through the configuration of the MVME050 system controller for I/O operations.

- Step 1. Remove the MVME050 module.  
Remove power from the system and eject the MVME050 module.

- Step 2. Configure the jumpers on the MVME050 module.

The jumpers on the MVME050 module should remain the same as the factory configuration (or configured specifically to your needs) with the exceptions of J20, J23, and J24.

**Figure 6-4 Jumper Settings for MVME050**

**I/O Base Address Select:**

Select the base address of \$1000 (Short I/O Space) by connecting the following jumpers:

J20: 1 to 2, 3 to 4, 5 to 6, 9 to 10, 11 to 12, 13 to 14

**Jumper Block J20**

|   |   |   |   |    |    |    |
|---|---|---|---|----|----|----|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|   |   |   |   |    |    |    |
| 1 | 3 | 5 | 7 | 9  | 11 | 13 |

**Txc Signal Serial Input:**

J23 controls port #1 and J24 controls port #2. These settings select the transmit clock (Txc) signal as an input for the RS-232 ports.

**Jumper Block J23 Jumper Block J24**

|   |   |
|---|---|
| 2 | 4 |
|   |   |
| 1 | 3 |

J23: 3 to 4

|   |   |
|---|---|
| 2 | 4 |
|   |   |
| 1 | 3 |

J24: 3 to 4

- Step 3. Set the front panel switch block on the MVME050 module.

The front panel switch block of the MVME050 controls various options you can set for the system. Microware reserves switch 1 for future use. Switches 6 and 7 are not used by Microware and may be used for your applications.

Switch 2 is the ROM boot override. If closed, switches 3 and 4 select the boot method. If open, the kernel is searched for in ROM (the system does a **ROM** boot).



Switch 3 (menu select) and switch 4 (boot method) control how the system comes up. These switches may be set in the following combinations:

**Figure 6-5 Switch Combinations and Definitions**

| Switch 3 | Switch 4 | Definition                                            |
|----------|----------|-------------------------------------------------------|
| on       | on       | <code>sysboot</code> builds user select menu          |
| on       | off      | <code>sysboot</code> uses <b>AUTOSELECT</b> algorithm |
| off      | on       | Boots from the primary device                         |
| off      | off      | Boots from the secondary device                       |

Switch 5 controls the enabling of the ROM debugger. If closed, the ROM debugger is called after a system reset. If open, the normal booting action takes place after a system reset.

Switch 8 controls the front panel LED display. If this switch is off, the LED display is enabled. If this switch is on, the LED display is blanked.

Step 4. Replace the MVME050 module.

## Configuring the MVME701 I/O Transition Module

The MVME701 module is a support module for the MVME050 system controller module:

Step 1. Remove the MVME701A module.

Remove power from the system, and eject the MVME701 module.

Step 2. Configure the jumpers on the MVME701A module.

The jumpers on the MVME701A module control options for the following functions:

### **Figure 6-6 Jumper Settings for MVME701A**

#### **Serial Port 1 Direction:**

**To Terminal** operation (DCE): remove all jumpers from J6 and install all jumpers on J5 (1 to 2, 3 to 4, 5 to 6, etc.).

**To Modem** operation (DTE): remove all jumpers from J5 and install all jumpers on J6 (1 to 2, 3 to 4, 5 to 6, etc.).

#### **Serial Port 2 Direction:**

**To Terminal** operation (DCE): remove all jumpers from J10 and install all jumpers on J9 (1 to 2, 3 to 4, 5 to 6, etc.).

**To Modem** operation (DTE): remove all jumpers from J9 and install all jumpers on J10 (1 to 2, 3 to 4, 5 to 6, etc.).

#### **Serial Port 2 DSR and CTS Configuration:**

Hold DSR signal always TRUE, install J8: 1 to 2

To drive the DSR signal with your terminal, install J8: 2 to 3

Hold CTS signal always TRUE, install J12: 1 to 2

To drive the CTS signal with your terminal, install J12: 2 to 3

#### **Serial Port 1 DSR and CTS Configuration:**

Hold DSR signal always TRUE, install: J7: 1 to 2

To drive the DSR signal with your terminal, install J7: 2 to 3

Hold CTS signal always TRUE, install J11: 1 to 2

To drive the CTS signal with your terminal, install J11: 2 to 3

Step 3. Replace the MVME701A module.

---



---

# Appendix A: Downloading RomBug from a Host over a Serial Interface

---

This appendix describes the process of downloading the RomBug image from a cross-development host over a serial (RS232) interface. This information applies to the following reference boards:

- MVME162
- MVME172
- MVME177



# Downloading RomBug from a Host over a Serial Interface

---

To download the RomBug image from a cross-development host over a serial (RS232) interface, complete the following steps:



## Note

Before proceeding, review the steps for installing RomBug into flash. A version of these steps is included in each chapter of this manual; therefore, you will need to refer to the chapter respective to your reference board.

---

- Step 1. From a DOS prompt, change into the applicable **BOOTFILES** directory (<MWOS>\OS9\CPU\PORTS<TARGET>\CMDS\BOOTOBS\BOOTFILES) and execute the following command:

```
binex -s=2 rombugger > rombugger.S
```

The binex utility converts binary files to S-record files. An S-record file is a type of text file that contains records representing binary data in ASCII hexadecimal form. This Motorola-standard format is one of the formats accepted when you transfer an executable file from the host computer to the target board.

---



## For More Information

For more information on Boot and ROM customization, refer to the ***OS-9 for 68K Processors BLS Reference***, included with this documentation set.

---

- Step 2. With the target system powered off, use a serial cable to connect the target board's RS232 serial port 2 to an unused RS-232 COM port on the host PC.
- Step 3. Open a terminal emulation program, such as Hyperterminal, to establish a connection between the host PC and the target board. The host PC's port should be configured for the following settings:
- 9600 baud
  - 8 data bits
  - one stop bit
  - no parity
  - flow control: XON/XOFF enabled
- Step 4. Enter the following command to initiate reception of the `rombugger.S` file from the host PC to the MVME172 board:
- ```
1XX-Bug>LO 1 100000
```
- The `LO` command accepts serial data from the host PC and loads it into the target board's memory. Some boards allow use of the `X` option (`LO 1 100000 ; X`) to echo the `S`-records to your terminal as they are read in at the host port.
- Step 5. Use the terminal emulation program to initiate transmission of the `rombugger.S` text file from the host PC to the target board. If an error occurs, repeat the transfer, starting with step two, until it is successful.
- Step 6. Complete the installation by performing the steps for programming the image into flash. These steps are described in the **Installing RomBug into Flash** section of this manual. Because a version of this section is included in each chapter of this manual, you will need to refer to the chapter respective to your reference board.
-

